

StdAir

0.44.0

Generated by Doxygen 1.7.4

Wed Nov 2 2011 22:19:08

Contents

1	StdAir Documentation	1
1.1	Getting Started	1
1.2	StdAir at SourceForge	1
1.3	StdAir Development	1
1.4	External Libraries	2
1.5	Support StdAir	2
1.6	About StdAir	2
2	C++ Utility Class Browsing and Dumping the StdAir BOM Tree	2
3	C++ Class Building Sample StdAir BOM Trees	21
4	C++ Class Storing the StdAir Context	63
5	People	64
5.1	Project Admins (and Developers)	64
5.2	Retired Developers	65
5.3	Contributors	65
5.4	Distribution Maintainers	65
6	Coding Rules	65
6.1	Default Naming Rules for Variables	65
6.2	Default Naming Rules for Functions	65
6.3	Default Naming Rules for Classes and Structures	66
6.4	Default Naming Rules for Files	66
6.5	Default Functionality of Classes	66
7	Copyright and License	66
7.1	GNU LESSER GENERAL PUBLIC LICENSE	66
7.1.1	Version 2.1, February 1999	66
7.2	Preamble	67
7.3	TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MOD- IFICATION	68
7.3.1	NO WARRANTY	73
7.3.2	END OF TERMS AND CONDITIONS	73

7.4	How to Apply These Terms to Your New Programs	74
8	Documentation Rules	74
8.1	General Rules	74
8.2	File Header	75
8.3	Grouping Various Parts	76
9	Main features	76
9.1	Standard Airline IT Business Object Model (BOM)	76
9.2	Architecture of the StdAir library	77
10	Make a Difference	77
11	Make a new release	78
11.1	Introduction	78
11.2	Initialisation	78
11.3	Branch creation	78
11.4	Commit and publish the release branch	79
11.5	Update the change-log in the trunk as well	79
11.6	Create distribution packages	79
11.7	Generation the RPM packages	79
11.8	Update distributed change log	80
11.9	Create the binary package, including the documentation	80
11.10	Upload the files to SourceForge	80
11.11	Upload the documentation to SourceForge	80
11.12	Make a new post	81
11.13	Send an email on the announcement mailing-list	81
12	Installation	81
12.1	Table of Contents	81
12.2	Fedora/RedHat Linux distributions	81
12.3	StdAir Requirements	82
12.4	Basic Installation	82
12.5	Compilers and Options	83
12.6	Compiling For Multiple Architectures	84
12.7	Installation Names	84

12.8	Optional Features	85
12.9	Particular systems	86
12.10	Specifying the System Type	87
12.11	Sharing Defaults	87
12.12	Defining Variables	87
12.13	'cmake' Invocation	88
13	Linking with StdAir	92
13.1	Table of Contents	92
13.2	Introduction	92
13.3	Using the pkg-config command	93
13.4	Using the stdair-config script	93
13.5	M4 macro for the GNU Autotools	93
13.6	Using StdAir with dynamic linking	94
14	Test Rules	94
14.1	The Test Source Files	94
14.2	The Reference File	94
14.3	Testing StdAir Library	94
15	Users Guide	95
15.1	Table of Contents	95
15.2	Introduction	95
15.3	Get Started	95
15.3.1	Get the StdAir library	95
15.3.2	Build the StdAir project	95
15.3.3	Build and Run the Tests	96
15.3.4	Install the StdAir Project (Binaries, Documentation)	96
15.4	Exploring the Predefined BOM Tree	96
15.4.1	Airline Distribution BOM Tree	96
15.4.2	Airline Network BOM Tree	96
15.4.3	Airline Inventory BOM Tree	96
15.5	Extending the BOM Tree	96
16	Supported Systems	96

16.1 Table of Contents	96
16.2 Introduction	97
16.3 StdAir 3.10.x	97
16.3.1 Linux Systems	97
16.3.2 Windows Systems	102
16.3.3 Unix Systems	105
17 StdAir Supported Systems (Previous Releases)	105
17.1 StdAir 3.9.1	105
17.2 StdAir 3.9.0	105
17.3 StdAir 3.8.1	105
18 Tutorials	105
18.1 Table of Contents	105
18.2 Introduction	106
18.2.1 Preparing the StdAir Project for Development	106
18.3 Build a Predefined BOM Tree	106
18.3.1 Instantiate the BOM Root Object	106
18.3.2 Instantiate the (Airline) Inventory Object	107
18.3.3 Link the Inventory Object with the BOM Root	107
18.3.4 Build Another Airline Inventory	107
18.3.5 Dump The BOM Tree Content	107
18.3.6 Result of the Tutorial Program	109
18.4 Extend the Pre-Defined BOM Tree	111
18.4.1 Extend an Airline Inventory Object	111
18.4.2 Build the Specific BOM Objects	112
18.4.3 Result of the Tutorial Program	113
19 Command-Line Utility to Demonstrate Typical StdAir Usage	113
20 Specific Implementation of a BOM Root	117
21 Specific Implementation of a BOM Root	118
22 Specific Implementation of an Airline Inventory	118
23 Specific Implementation of an Airline Inventory	119

24 Command-Line Test to Demonstrate How To Extend StdAir BOM	120
25 Module Index	125
25.1 Modules	125
26 Directory Hierarchy	125
26.1 Directories	125
27 Namespace Index	126
27.1 Namespace List	126
28 Class Index	126
28.1 Class Hierarchy	126
29 Class Index	132
29.1 Class List	132
30 File Index	139
30.1 File List	139
31 Module Documentation	149
31.1 Abstract part of the Business Object Model (BOM)	149
31.2 Part of the Business Object Model (BOM) handling	149
32 Directory Documentation	149
32.1 stdair/basic/ Directory Reference	150
32.2 batches/ Directory Reference	151
32.3 stdair/bom/ Directory Reference	151
32.4 stdair/ui/cmdline/ Directory Reference	155
32.5 stdair/command/ Directory Reference	155
32.6 stdair/config/ Directory Reference	155
32.7 stdair/dbadaptor/ Directory Reference	155
32.8 stdair/factory/ Directory Reference	156
32.9 stdair/service/ Directory Reference	156
32.10test/stdair/ Directory Reference	156
32.11stdair/ Directory Reference	157
32.12test/ Directory Reference	157

32.13stdair/ui/ Directory Reference	157
33 Namespace Documentation	158
33.1 boost Namespace Reference	158
33.1.1 Detailed Description	158
33.2 boost::serialization Namespace Reference	158
33.3 bpt Namespace Reference	158
33.3.1 Typedef Documentation	158
33.4 soci Namespace Reference	158
33.5 stdair Namespace Reference	158
33.5.1 Detailed Description	178
33.5.2 Typedef Documentation	178
33.5.3 Function Documentation	208
33.5.4 Variable Documentation	222
33.6 stdair::LOG Namespace Reference	235
33.6.1 Detailed Description	236
33.6.2 Enumeration Type Documentation	236
33.6.3 Variable Documentation	236
33.7 stdair_test Namespace Reference	236
33.7.1 Detailed Description	236
33.8 swift Namespace Reference	237
33.8.1 Detailed Description	237
34 Class Documentation	237
34.1 stdair::AirlineClassList Class Reference	237
34.1.1 Detailed Description	238
34.1.2 Member Typedef Documentation	238
34.1.3 Constructor & Destructor Documentation	239
34.1.4 Member Function Documentation	239
34.1.5 Friends And Related Function Documentation	241
34.1.6 Member Data Documentation	241
34.2 stdair::AirlineClassListKey Struct Reference	242
34.2.1 Detailed Description	243
34.2.2 Constructor & Destructor Documentation	243
34.2.3 Member Function Documentation	243

34.2.4	Friends And Related Function Documentation	245
34.3	stdair::AirlineFeature Class Reference	245
34.3.1	Detailed Description	246
34.3.2	Member Typedef Documentation	246
34.3.3	Constructor & Destructor Documentation	246
34.3.4	Member Function Documentation	247
34.3.5	Friends And Related Function Documentation	248
34.3.6	Member Data Documentation	248
34.4	stdair::AirlineFeatureKey Struct Reference	249
34.4.1	Detailed Description	249
34.4.2	Constructor & Destructor Documentation	249
34.4.3	Member Function Documentation	249
34.5	stdair::AirlineStruct Struct Reference	251
34.5.1	Detailed Description	251
34.5.2	Constructor & Destructor Documentation	251
34.5.3	Member Function Documentation	252
34.6	stdair::AirportPair Class Reference	253
34.6.1	Detailed Description	254
34.6.2	Member Typedef Documentation	254
34.6.3	Constructor & Destructor Documentation	255
34.6.4	Member Function Documentation	255
34.6.5	Friends And Related Function Documentation	256
34.6.6	Member Data Documentation	257
34.7	stdair::AirportPairKey Struct Reference	257
34.7.1	Detailed Description	258
34.7.2	Constructor & Destructor Documentation	258
34.7.3	Member Function Documentation	258
34.8	stdair::BasChronometer Struct Reference	259
34.8.1	Detailed Description	260
34.8.2	Constructor & Destructor Documentation	260
34.8.3	Member Function Documentation	260
34.9	stdair::BasDBParams Struct Reference	260
34.9.1	Detailed Description	261
34.9.2	Constructor & Destructor Documentation	261

34.9.3 Member Function Documentation	262
34.10stdair::BasFileMgr Struct Reference	264
34.10.1 Detailed Description	265
34.10.2 Member Function Documentation	265
34.11stdair::BasLogParams Struct Reference	265
34.11.1 Detailed Description	266
34.11.2 Constructor & Destructor Documentation	266
34.11.3 Member Function Documentation	266
34.11.4 Friends And Related Function Documentation	268
34.12stdair::BomAbstract Class Reference	268
34.12.1 Detailed Description	269
34.12.2 Constructor & Destructor Documentation	270
34.12.3 Member Function Documentation	270
34.13stdair::BomArchive Class Reference	271
34.13.1 Detailed Description	271
34.13.2 Member Function Documentation	271
34.14stdair::BomDisplay Class Reference	272
34.14.1 Detailed Description	273
34.14.2 Member Function Documentation	273
34.15stdair::BomHolder< BOM > Class Template Reference	280
34.15.1 Detailed Description	281
34.15.2 Member Typedef Documentation	281
34.15.3 Constructor & Destructor Documentation	282
34.15.4 Member Function Documentation	282
34.15.5 Friends And Related Function Documentation	283
34.15.6 Member Data Documentation	283
34.16stdair::BomHolderKey Struct Reference	284
34.16.1 Detailed Description	284
34.16.2 Constructor & Destructor Documentation	285
34.16.3 Member Function Documentation	285
34.17stdair::BomJSONExport Class Reference	286
34.17.1 Detailed Description	286
34.17.2 Member Function Documentation	286
34.18stdair::BomJSONImport Class Reference	287

34.18.1 Detailed Description	287
34.18.2 Member Function Documentation	287
34.19stdair::BomKeyManager Class Reference	288
34.19.1 Detailed Description	288
34.19.2 Member Function Documentation	288
34.20stdair::BomManager Class Reference	290
34.20.1 Detailed Description	291
34.20.2 Member Function Documentation	291
34.20.3 Friends And Related Function Documentation	293
34.21stdair::BomRetriever Class Reference	293
34.21.1 Detailed Description	294
34.21.2 Member Function Documentation	294
34.22stdair::BomRoot Class Reference	302
34.22.1 Detailed Description	303
34.22.2 Member Typedef Documentation	303
34.22.3 Constructor & Destructor Documentation	303
34.22.4 Member Function Documentation	304
34.22.5 Friends And Related Function Documentation	306
34.22.6 Member Data Documentation	306
34.23stdair::BomRootKey Struct Reference	307
34.23.1 Detailed Description	307
34.23.2 Constructor & Destructor Documentation	307
34.23.3 Member Function Documentation	308
34.23.4 Friends And Related Function Documentation	309
34.24stdair_test::BookingClass Struct Reference	309
34.24.1 Detailed Description	309
34.24.2 Constructor & Destructor Documentation	310
34.24.3 Member Function Documentation	310
34.24.4 Member Data Documentation	310
34.25stdair::BookingClass Class Reference	310
34.25.1 Detailed Description	312
34.25.2 Member Typedef Documentation	313
34.25.3 Constructor & Destructor Documentation	313
34.25.4 Member Function Documentation	313

34.25.5 Friends And Related Function Documentation	320
34.25.6 Member Data Documentation	320
34.26stdair::BookingClassKey Struct Reference	324
34.26.1 Detailed Description	324
34.26.2 Constructor & Destructor Documentation	324
34.26.3 Member Function Documentation	325
34.27stdair::BookingRequestStruct Struct Reference	326
34.27.1 Detailed Description	327
34.27.2 Constructor & Destructor Documentation	327
34.27.3 Member Function Documentation	328
34.28stdair::Bucket Class Reference	332
34.28.1 Detailed Description	333
34.28.2 Member Typedef Documentation	333
34.28.3 Constructor & Destructor Documentation	333
34.28.4 Member Function Documentation	334
34.28.5 Friends And Related Function Documentation	336
34.28.6 Member Data Documentation	336
34.29stdair::BucketKey Struct Reference	337
34.29.1 Detailed Description	338
34.29.2 Constructor & Destructor Documentation	338
34.29.3 Member Function Documentation	339
34.29.4 Friends And Related Function Documentation	340
34.30stdair_test::Cabin Struct Reference	340
34.30.1 Detailed Description	340
34.30.2 Member Typedef Documentation	340
34.30.3 Constructor & Destructor Documentation	341
34.30.4 Member Function Documentation	341
34.30.5 Member Data Documentation	341
34.31stdair::CancellationStruct Struct Reference	341
34.31.1 Detailed Description	342
34.31.2 Constructor & Destructor Documentation	342
34.31.3 Member Function Documentation	342
34.32stdair::CmdAbstract Class Reference	344
34.32.1 Detailed Description	344

34.33stdair::CmdBomManager Class Reference	344
34.33.1 Detailed Description	344
34.33.2 Friends And Related Function Documentation	344
34.34stdair::CmdBomSerialiser Class Reference	345
34.34.1 Detailed Description	345
34.35stdair::CodeConversionException Class Reference	345
34.35.1 Detailed Description	346
34.35.2 Constructor & Destructor Documentation	346
34.35.3 Member Function Documentation	346
34.35.4 Member Data Documentation	346
34.36stdair::CodeDuplicationException Class Reference	346
34.36.1 Detailed Description	347
34.36.2 Constructor & Destructor Documentation	347
34.36.3 Member Function Documentation	347
34.36.4 Member Data Documentation	347
34.37COMMAND Struct Reference	348
34.37.1 Detailed Description	348
34.37.2 Member Data Documentation	348
34.38stdair::ContinuousAttributeLite< T > Struct Template Reference	349
34.38.1 Detailed Description	349
34.38.2 Member Typedef Documentation	349
34.38.3 Constructor & Destructor Documentation	349
34.38.4 Member Function Documentation	350
34.39stdair::date_time_element< MIN, MAX > Struct Template Reference	351
34.39.1 Detailed Description	351
34.39.2 Constructor & Destructor Documentation	352
34.39.3 Member Function Documentation	352
34.39.4 Member Data Documentation	352
34.40stdair::DatePeriod Class Reference	352
34.40.1 Detailed Description	354
34.40.2 Member Typedef Documentation	354
34.40.3 Constructor & Destructor Documentation	354
34.40.4 Member Function Documentation	354
34.40.5 Friends And Related Function Documentation	356

34.40.6 Member Data Documentation	356
34.41stdair::DatePeriodKey Struct Reference	357
34.41.1 Detailed Description	357
34.41.2 Constructor & Destructor Documentation	357
34.41.3 Member Function Documentation	358
34.42stdair::DbAbstract Class Reference	359
34.42.1 Detailed Description	359
34.42.2 Constructor & Destructor Documentation	359
34.42.3 Member Function Documentation	359
34.43stdair::DBManagerForAirlines Class Reference	360
34.43.1 Detailed Description	360
34.43.2 Member Function Documentation	360
34.44stdair::DBSessionManager Class Reference	361
34.44.1 Detailed Description	362
34.44.2 Member Function Documentation	362
34.44.3 Friends And Related Function Documentation	362
34.45stdair::DefaultDCPList Struct Reference	362
34.45.1 Detailed Description	363
34.45.2 Member Function Documentation	363
34.46stdair::DefaultDtdFratMap Struct Reference	363
34.46.1 Detailed Description	363
34.46.2 Member Function Documentation	363
34.47stdair::DefaultDtdProbMap Struct Reference	363
34.47.1 Detailed Description	364
34.47.2 Member Function Documentation	364
34.48stdair::DemandGenerationMethod Struct Reference	364
34.48.1 Detailed Description	365
34.48.2 Member Enumeration Documentation	365
34.48.3 Constructor & Destructor Documentation	365
34.48.4 Member Function Documentation	366
34.49stdair::DictionaryManager Class Reference	368
34.49.1 Detailed Description	368
34.49.2 Member Function Documentation	368
34.50stdair::DocumentNotFoundException Class Reference	369

34.50.1 Detailed Description	369
34.50.2 Constructor & Destructor Documentation	370
34.50.3 Member Function Documentation	370
34.50.4 Member Data Documentation	370
34.51stdair::DoWStruct Struct Reference	370
34.51.1 Detailed Description	371
34.51.2 Member Typedef Documentation	371
34.51.3 Constructor & Destructor Documentation	371
34.51.4 Member Function Documentation	372
34.52stdair::EventException Class Reference	374
34.52.1 Detailed Description	374
34.52.2 Constructor & Destructor Documentation	374
34.52.3 Member Function Documentation	374
34.52.4 Member Data Documentation	375
34.53stdair::EventQueue Class Reference	375
34.53.1 Detailed Description	377
34.53.2 Member Typedef Documentation	378
34.53.3 Constructor & Destructor Documentation	378
34.53.4 Member Function Documentation	378
34.53.5 Friends And Related Function Documentation	385
34.53.6 Member Data Documentation	386
34.54stdair::EventQueueException Class Reference	387
34.54.1 Detailed Description	387
34.54.2 Constructor & Destructor Documentation	387
34.54.3 Member Function Documentation	388
34.54.4 Member Data Documentation	388
34.55stdair::EventQueueKey Struct Reference	388
34.55.1 Detailed Description	389
34.55.2 Constructor & Destructor Documentation	389
34.55.3 Member Function Documentation	389
34.56stdair::EventStruct Struct Reference	390
34.56.1 Detailed Description	391
34.56.2 Constructor & Destructor Documentation	391
34.56.3 Member Function Documentation	392

34.56.4 Friends And Related Function Documentation	394
34.57stdair::EventType Struct Reference	395
34.57.1 Detailed Description	395
34.57.2 Member Enumeration Documentation	396
34.57.3 Constructor & Destructor Documentation	396
34.57.4 Member Function Documentation	396
34.58stdair::FacAbstract Class Reference	398
34.58.1 Detailed Description	399
34.58.2 Constructor & Destructor Documentation	399
34.59stdair::FacBom< BOM > Class Template Reference	399
34.59.1 Detailed Description	400
34.59.2 Constructor & Destructor Documentation	400
34.59.3 Member Function Documentation	400
34.60stdair::FacBomManager Class Reference	401
34.60.1 Detailed Description	402
34.60.2 Constructor & Destructor Documentation	402
34.60.3 Member Function Documentation	402
34.61stdair::FacServiceAbstract Class Reference	406
34.61.1 Detailed Description	406
34.61.2 Member Typedef Documentation	407
34.61.3 Constructor & Destructor Documentation	407
34.61.4 Member Function Documentation	407
34.61.5 Member Data Documentation	407
34.62stdair::FacSTDAIRServiceContext Class Reference	408
34.62.1 Detailed Description	408
34.62.2 Member Typedef Documentation	409
34.62.3 Constructor & Destructor Documentation	409
34.62.4 Member Function Documentation	409
34.62.5 Member Data Documentation	410
34.63stdair::FacSupervisor Class Reference	410
34.63.1 Detailed Description	411
34.63.2 Member Typedef Documentation	411
34.63.3 Constructor & Destructor Documentation	411
34.63.4 Member Function Documentation	412

34.64stdair::FareFamily Class Reference	413
34.64.1 Detailed Description	415
34.64.2 Member Typedef Documentation	415
34.64.3 Constructor & Destructor Documentation	415
34.64.4 Member Function Documentation	415
34.64.5 Friends And Related Function Documentation	417
34.64.6 Member Data Documentation	417
34.65stdair::FareFamilyKey Struct Reference	418
34.65.1 Detailed Description	418
34.65.2 Constructor & Destructor Documentation	418
34.65.3 Member Function Documentation	419
34.65.4 Friends And Related Function Documentation	420
34.66stdair::FareFeatures Class Reference	420
34.66.1 Detailed Description	421
34.66.2 Member Typedef Documentation	421
34.66.3 Constructor & Destructor Documentation	422
34.66.4 Member Function Documentation	422
34.66.5 Friends And Related Function Documentation	425
34.66.6 Member Data Documentation	425
34.67stdair::FareFeaturesKey Struct Reference	426
34.67.1 Detailed Description	426
34.67.2 Constructor & Destructor Documentation	426
34.67.3 Member Function Documentation	427
34.68stdair::FareOptionStruct Struct Reference	428
34.68.1 Detailed Description	429
34.68.2 Constructor & Destructor Documentation	429
34.68.3 Member Function Documentation	430
34.69stdair::FileNotFoundException Class Reference	432
34.69.1 Detailed Description	433
34.69.2 Constructor & Destructor Documentation	433
34.69.3 Member Function Documentation	433
34.69.4 Member Data Documentation	433
34.70stdair::FlagSaver Struct Reference	434
34.70.1 Detailed Description	434

34.70.2 Constructor & Destructor Documentation	434
34.71stdair::FlightDate Class Reference	434
34.71.1 Detailed Description	436
34.71.2 Member Typedef Documentation	436
34.71.3 Constructor & Destructor Documentation	436
34.71.4 Member Function Documentation	436
34.71.5 Friends And Related Function Documentation	440
34.71.6 Member Data Documentation	440
34.72stdair::FlightDateKey Struct Reference	441
34.72.1 Detailed Description	441
34.72.2 Constructor & Destructor Documentation	442
34.72.3 Member Function Documentation	442
34.72.4 Friends And Related Function Documentation	443
34.73stdair::FlightPeriod Class Reference	443
34.73.1 Detailed Description	444
34.73.2 Member Typedef Documentation	445
34.73.3 Constructor & Destructor Documentation	445
34.73.4 Member Function Documentation	445
34.73.5 Friends And Related Function Documentation	447
34.73.6 Member Data Documentation	447
34.74stdair::FlightPeriodKey Struct Reference	447
34.74.1 Detailed Description	448
34.74.2 Constructor & Destructor Documentation	448
34.74.3 Member Function Documentation	448
34.75FloatingPoint< RawType > Class Template Reference	449
34.75.1 Detailed Description	450
34.75.2 Member Typedef Documentation	450
34.75.3 Constructor & Destructor Documentation	451
34.75.4 Member Function Documentation	451
34.75.5 Member Data Documentation	452
34.76stdair::ForecastingMethod Struct Reference	453
34.76.1 Detailed Description	454
34.76.2 Member Enumeration Documentation	454
34.76.3 Constructor & Destructor Documentation	454

34.76.4 Member Function Documentation	455
34.77stdair::GuillotineBlock Class Reference	457
34.77.1 Detailed Description	459
34.77.2 Member Typedef Documentation	459
34.77.3 Constructor & Destructor Documentation	459
34.77.4 Member Function Documentation	459
34.77.5 Friends And Related Function Documentation	465
34.77.6 Member Data Documentation	465
34.78stdair::GuillotineBlockKey Struct Reference	467
34.78.1 Detailed Description	468
34.78.2 Constructor & Destructor Documentation	468
34.78.3 Member Function Documentation	468
34.78.4 Friends And Related Function Documentation	469
34.79stdair::InputFilePath Class Reference	470
34.79.1 Detailed Description	470
34.79.2 Constructor & Destructor Documentation	470
34.79.3 Member Function Documentation	470
34.79.4 Member Data Documentation	471
34.80stdair::Inventory Class Reference	471
34.80.1 Detailed Description	472
34.80.2 Member Typedef Documentation	472
34.80.3 Constructor & Destructor Documentation	472
34.80.4 Member Function Documentation	473
34.80.5 Friends And Related Function Documentation	475
34.80.6 Member Data Documentation	476
34.81stdair::InventoryKey Struct Reference	476
34.81.1 Detailed Description	477
34.81.2 Constructor & Destructor Documentation	477
34.81.3 Member Function Documentation	478
34.81.4 Friends And Related Function Documentation	479
34.82stdair::KeyAbstract Struct Reference	479
34.82.1 Detailed Description	481
34.82.2 Constructor & Destructor Documentation	481
34.82.3 Member Function Documentation	481

34.83stdair::KeyNotFoundException Class Reference	482
34.83.1 Detailed Description	483
34.83.2 Constructor & Destructor Documentation	483
34.83.3 Member Function Documentation	483
34.83.4 Member Data Documentation	483
34.84stdair::LegCabin Class Reference	484
34.84.1 Detailed Description	486
34.84.2 Member Typedef Documentation	486
34.84.3 Constructor & Destructor Documentation	486
34.84.4 Member Function Documentation	487
34.84.5 Friends And Related Function Documentation	496
34.84.6 Member Data Documentation	496
34.85stdair::LegCabinKey Struct Reference	499
34.85.1 Detailed Description	500
34.85.2 Constructor & Destructor Documentation	500
34.85.3 Member Function Documentation	501
34.85.4 Friends And Related Function Documentation	502
34.86stdair::LegDate Class Reference	502
34.86.1 Detailed Description	503
34.86.2 Member Typedef Documentation	504
34.86.3 Constructor & Destructor Documentation	504
34.86.4 Member Function Documentation	504
34.86.5 Friends And Related Function Documentation	509
34.86.6 Member Data Documentation	509
34.87stdair::LegDateKey Struct Reference	511
34.87.1 Detailed Description	511
34.87.2 Constructor & Destructor Documentation	512
34.87.3 Member Function Documentation	512
34.88stdair::Logger Class Reference	513
34.88.1 Detailed Description	513
34.88.2 Member Function Documentation	514
34.88.3 Friends And Related Function Documentation	514
34.89stdair::MemoryAllocationException Class Reference	514
34.89.1 Detailed Description	515

34.89.2 Constructor & Destructor Documentation	515
34.89.3 Member Function Documentation	515
34.89.4 Member Data Documentation	515
34.90stdair::NonInitialisedContainerException Class Reference	515
34.90.1 Detailed Description	516
34.90.2 Constructor & Destructor Documentation	516
34.90.3 Member Function Documentation	516
34.90.4 Member Data Documentation	516
34.91stdair::NonInitialisedDBSessionManagerException Class Reference	517
34.91.1 Detailed Description	517
34.91.2 Constructor & Destructor Documentation	517
34.91.3 Member Function Documentation	517
34.91.4 Member Data Documentation	518
34.92stdair::NonInitialisedLogServiceException Class Reference	518
34.92.1 Detailed Description	518
34.92.2 Constructor & Destructor Documentation	519
34.92.3 Member Function Documentation	519
34.92.4 Member Data Documentation	519
34.93stdair::NonInitialisedRelationShipException Class Reference	519
34.93.1 Detailed Description	520
34.93.2 Constructor & Destructor Documentation	520
34.93.3 Member Function Documentation	520
34.93.4 Member Data Documentation	520
34.94stdair::NonInitialisedServiceException Class Reference	520
34.94.1 Detailed Description	521
34.94.2 Constructor & Destructor Documentation	521
34.94.3 Member Function Documentation	521
34.94.4 Member Data Documentation	521
34.95stdair::ObjectCreationgDuplicationException Class Reference	522
34.95.1 Detailed Description	522
34.95.2 Constructor & Destructor Documentation	522
34.95.3 Member Function Documentation	523
34.95.4 Member Data Documentation	523
34.96stdair::ObjectLinkingException Class Reference	523

34.96.1 Detailed Description	524
34.96.2 Constructor & Destructor Documentation	524
34.96.3 Member Function Documentation	524
34.96.4 Member Data Documentation	524
34.97stdair::ObjectNotFoundException Class Reference	524
34.97.1 Detailed Description	525
34.97.2 Constructor & Destructor Documentation	525
34.97.3 Member Function Documentation	525
34.97.4 Member Data Documentation	525
34.98stdair::OnDDate Class Reference	526
34.98.1 Detailed Description	527
34.98.2 Member Typedef Documentation	527
34.98.3 Constructor & Destructor Documentation	527
34.98.4 Member Function Documentation	528
34.98.5 Friends And Related Function Documentation	531
34.98.6 Member Data Documentation	531
34.99stdair::OnDDateKey Struct Reference	532
34.99.1 Detailed Description	533
34.99.2 Constructor & Destructor Documentation	533
34.99.3 Member Function Documentation	533
34.99.4 Friends And Related Function Documentation	535
34.100stdair::OptimisationNotificationStruct Struct Reference	535
34.100.1 Detailed Description	536
34.100.2 Constructor & Destructor Documentation	536
34.100.3 Member Function Documentation	537
34.101stdair::ParsedKey Struct Reference	539
34.101.1 Detailed Description	540
34.101.2 Constructor & Destructor Documentation	540
34.101.3 Member Function Documentation	540
34.101.4 Member Data Documentation	542
34.102stdair::ParserException Class Reference	543
34.102.1 Detailed Description	543
34.102.2 Constructor & Destructor Documentation	543
34.102.3 Member Function Documentation	544

34.102.4Member Data Documentation	544
34.103stdair::ParsingFileFailedException Class Reference	544
34.103.1Detailed Description	545
34.103.2Constructor & Destructor Documentation	545
34.103.3Member Function Documentation	545
34.103.4Member Data Documentation	545
34.104stdair::PartnershipTechnique Struct Reference	545
34.104.1Detailed Description	546
34.104.2Member Enumeration Documentation	547
34.104.3Constructor & Destructor Documentation	547
34.104.4Member Function Documentation	548
34.105stdair::PassengerType Struct Reference	550
34.105.1Detailed Description	551
34.105.2Member Enumeration Documentation	551
34.105.3Constructor & Destructor Documentation	551
34.105.4Member Function Documentation	551
34.106stdair::PeriodStruct Struct Reference	553
34.106.1Detailed Description	554
34.106.2Constructor & Destructor Documentation	554
34.106.3Member Function Documentation	554
34.107stdair::PosChannel Class Reference	556
34.107.1Detailed Description	557
34.107.2Member Typedef Documentation	558
34.107.3Constructor & Destructor Documentation	558
34.107.4Member Function Documentation	558
34.107.5Friends And Related Function Documentation	560
34.107.6Member Data Documentation	560
34.108stdair::PosChannelKey Struct Reference	560
34.108.1Detailed Description	561
34.108.2Constructor & Destructor Documentation	561
34.108.3Member Function Documentation	561
34.109stdair::ProgressStatus Struct Reference	563
34.109.1Detailed Description	563
34.109.2Constructor & Destructor Documentation	564

34.109.3	Member Function Documentation	565
34.110	Stdair::ProgressStatusSet Struct Reference	567
34.110.1	Detailed Description	568
34.110.2	Constructor & Destructor Documentation	568
34.110.3	Member Function Documentation	568
34.111	Stdair::RandomGeneration Struct Reference	570
34.111.1	Detailed Description	571
34.111.2	Constructor & Destructor Documentation	571
34.111.3	Member Function Documentation	572
34.111.4	Member Data Documentation	574
34.112	Stdair::RMEventStruct Struct Reference	574
34.112.1	Detailed Description	575
34.112.2	Constructor & Destructor Documentation	575
34.112.3	Member Function Documentation	576
34.113	Stdair::RootException Class Reference	577
34.113.1	Detailed Description	578
34.113.2	Constructor & Destructor Documentation	578
34.113.3	Member Function Documentation	578
34.113.4	Member Data Documentation	578
34.114	Stdair::RootFilePath Class Reference	579
34.114.1	Detailed Description	579
34.114.2	Constructor & Destructor Documentation	579
34.114.3	Member Function Documentation	580
34.114.4	Member Data Documentation	580
34.115	Stdair::SampleType Struct Reference	580
34.115.1	Detailed Description	581
34.115.2	Member Enumeration Documentation	582
34.115.3	Constructor & Destructor Documentation	582
34.115.4	Member Function Documentation	582
34.116	Stdair::SegmentCabin Class Reference	584
34.116.1	Detailed Description	586
34.116.2	Member Typedef Documentation	586
34.116.3	Constructor & Destructor Documentation	587
34.116.4	Member Function Documentation	587

34.116.5	Friends And Related Function Documentation	592
34.116.6	Member Data Documentation	592
34.117	Stdair::SegmentCabinKey Struct Reference	595
34.117.1	Detailed Description	595
34.117.2	Constructor & Destructor Documentation	595
34.117.3	Member Function Documentation	596
34.117.4	Friends And Related Function Documentation	597
34.118	Stdair::SegmentDate Class Reference	597
34.118.1	Detailed Description	599
34.118.2	Member Typedef Documentation	599
34.118.3	Constructor & Destructor Documentation	599
34.118.4	Member Function Documentation	599
34.118.5	Friends And Related Function Documentation	604
34.118.6	Member Data Documentation	604
34.119	Stdair::SegmentDateKey Struct Reference	606
34.119.1	Detailed Description	607
34.119.2	Constructor & Destructor Documentation	607
34.119.3	Member Function Documentation	607
34.119.4	Friends And Related Function Documentation	608
34.120	Stdair::SegmentPeriod Class Reference	609
34.120.1	Detailed Description	610
34.120.2	Member Typedef Documentation	610
34.120.3	Constructor & Destructor Documentation	610
34.120.4	Member Function Documentation	611
34.120.5	Friends And Related Function Documentation	614
34.120.6	Member Data Documentation	614
34.121	Stdair::SegmentPeriodKey Struct Reference	616
34.121.1	Detailed Description	616
34.121.2	Constructor & Destructor Documentation	616
34.121.3	Member Function Documentation	617
34.122	Stdair::SerialisationException Class Reference	618
34.122.1	Detailed Description	618
34.122.2	Constructor & Destructor Documentation	619
34.122.3	Member Function Documentation	619

34.122.4Member Data Documentation	619
34.123stdair::ServiceAbstract Class Reference	619
34.123.1Detailed Description	620
34.123.2Constructor & Destructor Documentation	620
34.123.3Member Function Documentation	620
34.124stdair::ServiceInitialisationType Struct Reference	621
34.124.1Detailed Description	622
34.124.2Member Enumeration Documentation	622
34.124.3Constructor & Destructor Documentation	622
34.124.4Member Function Documentation	622
34.125swift::SKeymap Class Reference	625
34.125.1Detailed Description	625
34.125.2Constructor & Destructor Documentation	626
34.125.3Member Function Documentation	626
34.125.4Friends And Related Function Documentation	627
34.126stdair::SnapshotStruct Struct Reference	627
34.126.1Detailed Description	628
34.126.2Constructor & Destructor Documentation	628
34.126.3Member Function Documentation	628
34.127stdair::SQLDatabaseConnectionImpossibleException Class Reference	629
34.127.1Detailed Description	630
34.127.2Constructor & Destructor Documentation	630
34.127.3Member Function Documentation	630
34.127.4Member Data Documentation	630
34.128stdair::SQLDatabaseException Class Reference	631
34.128.1Detailed Description	631
34.128.2Constructor & Destructor Documentation	631
34.128.3Member Function Documentation	631
34.128.4Member Data Documentation	632
34.129swift::SReadline Class Reference	632
34.129.1Detailed Description	633
34.129.2Constructor & Destructor Documentation	633
34.129.3Member Function Documentation	634
34.130stdair::STDAIR_Service Class Reference	637

34.130.1	Detailed Description	638
34.130.2	Constructor & Destructor Documentation	639
34.130.3	Member Function Documentation	639
34.131	stdair::STDAIR_ServiceContext Class Reference	649
34.131.1	Detailed Description	649
34.131.2	Member Function Documentation	649
34.131.3	Friends And Related Function Documentation	650
34.132	stdair::StructAbstract Struct Reference	650
34.132.1	Detailed Description	651
34.132.2	Constructor & Destructor Documentation	652
34.132.3	Member Function Documentation	652
34.133	stdair::TimePeriod Class Reference	653
34.133.1	Detailed Description	654
34.133.2	Member Typedef Documentation	654
34.133.3	Constructor & Destructor Documentation	654
34.133.4	Member Function Documentation	654
34.133.5	Friends And Related Function Documentation	656
34.133.6	Member Data Documentation	657
34.134	stdair::TimePeriodKey Struct Reference	657
34.134.1	Detailed Description	658
34.134.2	Constructor & Destructor Documentation	658
34.134.3	Member Function Documentation	658
34.135	stdair::TravelSolutionStruct Struct Reference	659
34.135.1	Detailed Description	660
34.135.2	Constructor & Destructor Documentation	660
34.135.3	Member Function Documentation	661
34.136	oci::type_conversion< stdair::AirlineStruct > Struct Template Reference	664
34.136.1	Detailed Description	664
34.136.2	Member Typedef Documentation	664
34.136.3	Member Function Documentation	664
34.137	TypeWithSize< size > Class Template Reference	665
34.137.1	Detailed Description	665
34.137.2	Member Typedef Documentation	665
34.138	TypeWithSize< 4 > Class Template Reference	665

34.138.1	Detailed Description	666
34.138.2	Member Typedef Documentation	666
34.139	TypeWithSize< 8 > Class Template Reference	666
34.139.1	Detailed Description	666
34.139.2	Member Typedef Documentation	666
34.140	stdair::VirtualClassStruct Struct Reference	667
34.140.1	Detailed Description	667
34.140.2	Constructor & Destructor Documentation	667
34.140.3	Member Function Documentation	668
34.141	stdair::YieldFeatures Class Reference	670
34.141.1	Detailed Description	671
34.141.2	Member Typedef Documentation	671
34.141.3	Constructor & Destructor Documentation	671
34.141.4	Member Function Documentation	672
34.141.5	Friends And Related Function Documentation	673
34.141.6	Member Data Documentation	674
34.142	stdair::YieldFeaturesKey Struct Reference	674
34.142.1	Detailed Description	675
34.142.2	Constructor & Destructor Documentation	675
34.142.3	Member Function Documentation	675
34.143	stdair::YieldRange Class Reference	676
34.143.1	Detailed Description	677
34.143.2	Constructor & Destructor Documentation	677
34.143.3	Member Function Documentation	678
34.144	stdair::YieldStore Class Reference	679
34.144.1	Detailed Description	680
34.144.2	Member Typedef Documentation	681
34.144.3	Constructor & Destructor Documentation	681
34.144.4	Member Function Documentation	681
34.144.5	Friends And Related Function Documentation	682
34.144.6	Member Data Documentation	682
34.145	stdair::YieldStoreKey Struct Reference	683
34.145.1	Detailed Description	683
34.145.2	Constructor & Destructor Documentation	684

34.145.3Member Function Documentation	684
35 File Documentation	685
35.1 batches/stdair.cpp File Reference	685
35.2 stdair.cpp	685
35.3 doc/local/authors.doc File Reference	690
35.4 doc/local/codingrules.doc File Reference	690
35.5 doc/local/copyright.doc File Reference	690
35.6 doc/local/documentation.doc File Reference	690
35.7 doc/local/features.doc File Reference	690
35.8 doc/local/help_wanted.doc File Reference	690
35.9 doc/local/howto_release.doc File Reference	690
35.10doc/local/index.doc File Reference	690
35.11doc/local/installation.doc File Reference	690
35.12doc/local/linking.doc File Reference	690
35.13doc/local/test.doc File Reference	690
35.14doc/local/users_guide.doc File Reference	690
35.15doc/local/verification.doc File Reference	690
35.16doc/tutorial/tutorial.doc File Reference	690
35.17stdair/basic/BasChronometer.cpp File Reference	690
35.18BasChronometer.cpp	690
35.19stdair/basic/BasChronometer.hpp File Reference	691
35.20BasChronometer.hpp	692
35.21stdair/basic/BasConst.cpp File Reference	692
35.22BasConst.cpp	696
35.23stdair/basic/BasConst_BomDisplay.hpp File Reference	703
35.24BasConst_BomDisplay.hpp	704
35.25stdair/basic/BasConst_BookingClass.hpp File Reference	704
35.26BasConst_BookingClass.hpp	705
35.27stdair/basic/BasConst_DefaultObject.hpp File Reference	706
35.28BasConst_DefaultObject.hpp	707
35.29stdair/basic/BasConst_Event.hpp File Reference	708
35.30BasConst_Event.hpp	709
35.31stdair/basic/BasConst_General.hpp File Reference	709

35.32BasConst_General.hpp	710
35.33stdair/basic/BasConst_Inventory.hpp File Reference	711
35.34BasConst_Inventory.hpp	712
35.35stdair/basic/BasConst_Period_BOM.hpp File Reference	713
35.36BasConst_Period_BOM.hpp	713
35.37stdair/basic/BasConst_Request.hpp File Reference	714
35.38BasConst_Request.hpp	715
35.39stdair/basic/BasConst_TravelSolution.hpp File Reference	716
35.40BasConst_TravelSolution.hpp	716
35.41stdair/basic/BasConst_Yield.hpp File Reference	717
35.42BasConst_Yield.hpp	717
35.43stdair/basic/BasDBParams.cpp File Reference	717
35.44BasDBParams.cpp	718
35.45stdair/basic/BasDBParams.hpp File Reference	719
35.46BasDBParams.hpp	719
35.47stdair/basic/BasFileMgr.cpp File Reference	721
35.48BasFileMgr.cpp	721
35.49stdair/basic/BasFileMgr.hpp File Reference	722
35.50BasFileMgr.hpp	722
35.51stdair/basic/BasLogParams.cpp File Reference	723
35.52BasLogParams.cpp	723
35.53stdair/basic/BasLogParams.hpp File Reference	724
35.54BasLogParams.hpp	724
35.55stdair/basic/BasParserHelperTypes.hpp File Reference	726
35.56BasParserHelperTypes.hpp	727
35.57stdair/basic/BasParserTypes.hpp File Reference	728
35.58BasParserTypes.hpp	728
35.59stdair/basic/BasTypes.hpp File Reference	729
35.60BasTypes.hpp	729
35.61stdair/basic/ContinuousAttributeLite.hpp File Reference	730
35.62ContinuousAttributeLite.hpp	730
35.63stdair/basic/DemandGenerationMethod.cpp File Reference	734
35.64DemandGenerationMethod.cpp	734
35.65stdair/basic/DemandGenerationMethod.hpp File Reference	737

35.66DemandGenerationMethod.hpp	737
35.67stdair/basic/DictionaryManager.cpp File Reference	738
35.68DictionaryManager.cpp	739
35.69stdair/basic/DictionaryManager.hpp File Reference	739
35.70DictionaryManager.hpp	740
35.71stdair/basic/EventType.cpp File Reference	740
35.72EventType.cpp	740
35.73stdair/basic/EventType.hpp File Reference	742
35.74EventType.hpp	743
35.75stdair/basic/float_utils.hpp File Reference	744
35.76float_utils.hpp	744
35.77stdair/basic/float_utils_google.hpp File Reference	744
35.78float_utils_google.hpp	744
35.79stdair/basic/ForecastingMethod.cpp File Reference	749
35.80ForecastingMethod.cpp	749
35.81stdair/basic/ForecastingMethod.hpp File Reference	751
35.82ForecastingMethod.hpp	751
35.83stdair/basic/PartnershipTechnique.cpp File Reference	752
35.84PartnershipTechnique.cpp	753
35.85stdair/basic/PartnershipTechnique.hpp File Reference	755
35.86PartnershipTechnique.hpp	756
35.87stdair/basic/PassengerType.cpp File Reference	757
35.88PassengerType.cpp	757
35.89stdair/basic/PassengerType.hpp File Reference	759
35.90PassengerType.hpp	759
35.91stdair/basic/ProgressStatus.cpp File Reference	760
35.92ProgressStatus.cpp	760
35.93stdair/basic/ProgressStatus.hpp File Reference	762
35.94ProgressStatus.hpp	762
35.95stdair/basic/ProgressStatusSet.cpp File Reference	764
35.96ProgressStatusSet.cpp	764
35.97stdair/basic/ProgressStatusSet.hpp File Reference	765
35.98ProgressStatusSet.hpp	766
35.99stdair/basic/RandomGeneration.cpp File Reference	767

35.100RandomGeneration.cpp	767
35.101stdair/basic/RandomGeneration.hpp File Reference	769
35.102RandomGeneration.hpp	769
35.103stdair/basic/SampleType.cpp File Reference	770
35.104SampleType.cpp	771
35.105stdair/basic/SampleType.hpp File Reference	773
35.106SampleType.hpp	773
35.107stdair/basic/ServiceInitialisationType.cpp File Reference	774
35.108ServiceInitialisationType.cpp	774
35.109stdair/basic/ServiceInitialisationType.hpp File Reference	777
35.110ServiceInitialisationType.hpp	777
35.111stdair/basic/StructAbstract.hpp File Reference	778
35.111.1Function Documentation	779
35.112StructAbstract.hpp	779
35.113stdair/basic/YieldRange.cpp File Reference	780
35.114YieldRange.cpp	781
35.115stdair/basic/YieldRange.hpp File Reference	782
35.116YieldRange.hpp	782
35.117stdair/bom/AirlineClassList.cpp File Reference	783
35.118AirlineClassList.cpp	784
35.119stdair/bom/AirlineClassList.hpp File Reference	785
35.120AirlineClassList.hpp	785
35.121stdair/bom/AirlineClassListKey.cpp File Reference	788
35.122AirlineClassListKey.cpp	788
35.123stdair/bom/AirlineClassListKey.hpp File Reference	790
35.124AirlineClassListKey.hpp	790
35.125stdair/bom/AirlineClassListTypes.hpp File Reference	792
35.126AirlineClassListTypes.hpp	792
35.127stdair/bom/AirlineFeature.cpp File Reference	793
35.128AirlineFeature.cpp	793
35.129stdair/bom/AirlineFeature.hpp File Reference	794
35.130AirlineFeature.hpp	794
35.131stdair/bom/AirlineFeatureKey.cpp File Reference	795
35.132AirlineFeatureKey.cpp	795

35.133	stdair/bom/AirlineFeatureKey.hpp File Reference	796
35.134	AirlineFeatureKey.hpp	796
35.135	stdair/bom/AirlineFeatureTypes.hpp File Reference	797
35.136	AirlineFeatureTypes.hpp	798
35.137	stdair/bom/AirlineStruct.cpp File Reference	798
35.138	AirlineStruct.cpp	798
35.139	stdair/bom/AirlineStruct.hpp File Reference	799
35.140	AirlineStruct.hpp	800
35.141	stdair/bom/AirportPair.cpp File Reference	801
35.142	AirportPair.cpp	801
35.143	stdair/bom/AirportPair.hpp File Reference	802
35.144	AirportPair.hpp	802
35.145	stdair/bom/AirportPairKey.cpp File Reference	804
35.146	AirportPairKey.cpp	804
35.147	stdair/bom/AirportPairKey.hpp File Reference	805
35.148	AirportPairKey.hpp	805
35.149	stdair/bom/AirportPairTypes.hpp File Reference	806
35.150	AirportPairTypes.hpp	807
35.151	stdair/bom/BomAbstract.hpp File Reference	807
35.151	Function Documentation	808
35.152	BomAbstract.hpp	808
35.153	stdair/bom/BomArchive.cpp File Reference	809
35.154	BomArchive.cpp	810
35.155	stdair/bom/BomArchive.hpp File Reference	811
35.156	BomArchive.hpp	812
35.157	stdair/bom/BomDisplay.cpp File Reference	812
35.158	BomDisplay.cpp	812
35.159	stdair/bom/BomDisplay.hpp File Reference	831
35.160	BomDisplay.hpp	831
35.161	stdair/bom/BomHolder.hpp File Reference	833
35.162	BomHolder.hpp	833
35.163	stdair/bom/BomHolderKey.cpp File Reference	834
35.164	BomHolderKey.cpp	835
35.165	stdair/bom/BomHolderKey.hpp File Reference	835

35.166	BomHolderKey.hpp	836
35.167	stdair/bom/BomJSONExport.cpp File Reference	836
35.168	BomJSONExport.cpp	837
35.169	stdair/bom/BomJSONExport.hpp File Reference	845
35.170	BomJSONExport.hpp	845
35.171	stdair/bom/BomJSONImport.cpp File Reference	846
35.172	BomJSONImport.cpp	846
35.173	stdair/bom/BomJSONImport.hpp File Reference	847
35.174	BomJSONImport.hpp	848
35.175	stdair/bom/BomKeyManager.cpp File Reference	848
35.176	BomKeyManager.cpp	849
35.177	stdair/bom/BomKeyManager.hpp File Reference	851
35.178	BomKeyManager.hpp	851
35.179	stdair/bom/BomManager.hpp File Reference	852
35.180	BomManager.hpp	852
35.181	stdair/bom/BomRetriever.cpp File Reference	857
35.182	BomRetriever.cpp	858
35.183	stdair/bom/BomRetriever.hpp File Reference	866
35.184	BomRetriever.hpp	866
35.185	stdair/bom/BomRoot.cpp File Reference	868
35.186	BomRoot.cpp	868
35.187	stdair/bom/BomRoot.hpp File Reference	869
35.188	BomRoot.hpp	870
35.189	stdair/bom/BomRootKey.cpp File Reference	871
35.190	BomRootKey.cpp	872
35.191	stdair/bom/BomRootKey.hpp File Reference	873
35.192	BomRootKey.hpp	874
35.193	stdair/bom/BookingClass.cpp File Reference	875
35.194	BookingClass.cpp	875
35.195	stdair/bom/BookingClass.hpp File Reference	877
35.196	BookingClass.hpp	877
35.197	stdair/bom/BookingClassKey.cpp File Reference	881
35.198	BookingClassKey.cpp	882
35.199	stdair/bom/BookingClassKey.hpp File Reference	883

35.20 6 BookingClassKey.hpp	883
35.20 5 tdair/bom/BookingClassTypes.hpp File Reference	884
35.20 2 BookingClassTypes.hpp	884
35.20 3 tdair/bom/BookingRequestStruct.cpp File Reference	885
35.20 4 BookingRequestStruct.cpp	885
35.20 5 tdair/bom/BookingRequestStruct.hpp File Reference	889
35.20 6 BookingRequestStruct.hpp	889
35.20 7 tdair/bom/BookingRequestTypes.hpp File Reference	892
35.20 8 BookingRequestTypes.hpp	892
35.20 9 tdair/bom/Bucket.cpp File Reference	893
35.21 0 Bucket.cpp	893
35.21 1 tdair/bom/Bucket.hpp File Reference	894
35.21 2 Bucket.hpp	895
35.21 3 tdair/bom/BucketKey.cpp File Reference	897
35.21 4 BucketKey.cpp	897
35.21 5 tdair/bom/BucketKey.hpp File Reference	899
35.21 6 BucketKey.hpp	899
35.21 7 tdair/bom/BucketTypes.hpp File Reference	900
35.21 8 BucketTypes.hpp	901
35.21 9 tdair/bom/CancellationStruct.cpp File Reference	901
35.22 0 CancellationStruct.cpp	902
35.22 1 tdair/bom/CancellationStruct.hpp File Reference	903
35.22 2 CancellationStruct.hpp	903
35.22 3 tdair/bom/CancellationTypes.hpp File Reference	905
35.22 4 CancellationTypes.hpp	905
35.22 5 tdair/bom/DatePeriod.cpp File Reference	905
35.22 6 DatePeriod.cpp	906
35.22 7 tdair/bom/DatePeriod.hpp File Reference	907
35.22 8 DatePeriod.hpp	907
35.22 9 tdair/bom/DatePeriodKey.cpp File Reference	908
35.23 0 DatePeriodKey.cpp	909
35.23 1 tdair/bom/DatePeriodKey.hpp File Reference	910
35.23 2 DatePeriodKey.hpp	910
35.23 3 tdair/bom/DatePeriodTypes.hpp File Reference	911

35.234	DatePeriodTypes.hpp	911
35.235	stdair/bom/DoWStruct.cpp File Reference	912
35.236	DoWStruct.cpp	912
35.237	stdair/bom/DoWStruct.hpp File Reference	914
35.238	DoWStruct.hpp	915
35.239	stdair/bom/EventQueue.cpp File Reference	916
35.240	EventQueue.cpp	916
35.241	stdair/bom/EventQueue.hpp File Reference	922
35.242	EventQueue.hpp	922
35.243	stdair/bom/EventQueueKey.cpp File Reference	925
35.244	EventQueueKey.cpp	926
35.245	stdair/bom/EventQueueKey.hpp File Reference	926
35.246	EventQueueKey.hpp	927
35.247	stdair/bom/EventQueueTypes.hpp File Reference	927
35.248	EventQueueTypes.hpp	928
35.249	stdair/bom/EventStruct.cpp File Reference	928
35.250	EventStruct.cpp	929
35.251	stdair/bom/EventStruct.hpp File Reference	933
35.252	EventStruct.hpp	933
35.253	stdair/bom/EventTypes.hpp File Reference	935
35.254	EventTypes.hpp	936
35.255	stdair/bom/FareFamily.cpp File Reference	936
35.256	FareFamily.cpp	937
35.257	stdair/bom/FareFamily.hpp File Reference	938
35.258	FareFamily.hpp	938
35.259	stdair/bom/FareFamilyKey.cpp File Reference	940
35.260	FareFamilyKey.cpp	941
35.261	stdair/bom/FareFamilyKey.hpp File Reference	942
35.262	FareFamilyKey.hpp	942
35.263	stdair/bom/FareFamilyTypes.hpp File Reference	944
35.264	FareFamilyTypes.hpp	944
35.265	stdair/bom/FareFeatures.cpp File Reference	944
35.266	FareFeatures.cpp	945
35.267	stdair/bom/FareFeatures.hpp File Reference	947

35.26	FareFeatures.hpp	947
35.26	stdair/bom/FareFeaturesKey.cpp File Reference	949
35.27	FareFeaturesKey.cpp	949
35.27	stdair/bom/FareFeaturesKey.hpp File Reference	950
35.27	FareFeaturesKey.hpp	951
35.27	stdair/bom/FareFeaturesTypes.hpp File Reference	952
35.27	FareFeaturesTypes.hpp	953
35.27	stdair/bom/FareOptionStruct.cpp File Reference	953
35.27	FareOptionStruct.cpp	953
35.27	stdair/bom/FareOptionStruct.hpp File Reference	955
35.27	FareOptionStruct.hpp	956
35.27	stdair/bom/FareOptionTypes.hpp File Reference	957
35.28	FareOptionTypes.hpp	958
35.28	stdair/bom/FlightDate.cpp File Reference	958
35.28	FlightDate.cpp	959
35.28	stdair/bom/FlightDate.hpp File Reference	960
35.28	FlightDate.hpp	961
35.28	stdair/bom/FlightDateKey.cpp File Reference	963
35.28	FlightDateKey.cpp	963
35.28	stdair/bom/FlightDateKey.hpp File Reference	965
35.28	FlightDateKey.hpp	965
35.28	stdair/bom/FlightDateTypes.hpp File Reference	967
35.29	FlightDateTypes.hpp	967
35.29	stdair/bom/FlightPeriod.cpp File Reference	967
35.29	FlightPeriod.cpp	968
35.29	stdair/bom/FlightPeriod.hpp File Reference	968
35.29	FlightPeriod.hpp	969
35.29	stdair/bom/FlightPeriodKey.cpp File Reference	970
35.29	FlightPeriodKey.cpp	970
35.29	stdair/bom/FlightPeriodKey.hpp File Reference	971
35.29	FlightPeriodKey.hpp	971
35.29	stdair/bom/FlightPeriodTypes.hpp File Reference	972
35.30	FlightPeriodTypes.hpp	972
35.30	stdair/bom/GuillotineBlock.cpp File Reference	973

35.302	GuillotineBlock.cpp	973
35.303	stdair/bom/GuillotineBlock.hpp File Reference	979
35.304	GuillotineBlock.hpp	979
35.305	stdair/bom/GuillotineBlockKey.cpp File Reference	983
35.306	GuillotineBlockKey.cpp	983
35.307	stdair/bom/GuillotineBlockKey.hpp File Reference	985
35.308	GuillotineBlockKey.hpp	985
35.309	stdair/bom/GuillotineBlockTypes.hpp File Reference	986
35.310	GuillotineBlockTypes.hpp	987
35.311	stdair/bom/Inventory.cpp File Reference	987
35.312	Inventory.cpp	988
35.313	stdair/bom/Inventory.hpp File Reference	989
35.314	Inventory.hpp	989
35.315	stdair/bom/InventoryKey.cpp File Reference	991
35.316	InventoryKey.cpp	992
35.317	stdair/bom/InventoryKey.hpp File Reference	993
35.318	InventoryKey.hpp	994
35.319	stdair/bom/InventoryTypes.hpp File Reference	995
35.320	InventoryTypes.hpp	995
35.321	stdair/bom/key_types.hpp File Reference	996
35.322	key_types.hpp	996
35.323	stdair/bom/KeyAbstract.hpp File Reference	996
35.323	Function Documentation	997
35.324	KeyAbstract.hpp	998
35.325	stdair/bom/LegCabin.cpp File Reference	999
35.326	LegCabin.cpp	999
35.327	stdair/bom/LegCabin.hpp File Reference	1001
35.328	LegCabin.hpp	1002
35.329	stdair/bom/LegCabinKey.cpp File Reference	1007
35.330	LegCabinKey.cpp	1008
35.331	stdair/bom/LegCabinKey.hpp File Reference	1009
35.332	LegCabinKey.hpp	1009
35.333	stdair/bom/LegCabinTypes.hpp File Reference	1010
35.334	LegCabinTypes.hpp	1011

35.335tdair/bom/LegDate.cpp File Reference	1011
35.336egDate.cpp	1012
35.337tdair/bom/LegDate.hpp File Reference	1013
35.338egDate.hpp	1014
35.339tdair/bom/LegDateKey.cpp File Reference	1017
35.340egDateKey.cpp	1017
35.341tdair/bom/LegDateKey.hpp File Reference	1018
35.342egDateKey.hpp	1018
35.343tdair/bom/LegDateTypes.hpp File Reference	1019
35.344egDateTypes.hpp	1020
35.345tdair/bom/OnDDate.cpp File Reference	1020
35.346nDDate.cpp	1020
35.347tdair/bom/OnDDate.hpp File Reference	1022
35.348nDDate.hpp	1023
35.349tdair/bom/OnDDateKey.cpp File Reference	1025
35.350nDDateKey.cpp	1026
35.351tdair/bom/OnDDateKey.hpp File Reference	1028
35.352nDDateKey.hpp	1028
35.353tdair/bom/OnDDateTypes.hpp File Reference	1030
35.354nDDateTypes.hpp	1030
35.355tdair/bom/OptimisationNotificationStruct.cpp File Reference	1031
35.356optimisationNotificationStruct.cpp	1031
35.357tdair/bom/OptimisationNotificationStruct.hpp File Reference	1033
35.358optimisationNotificationStruct.hpp	1033
35.359tdair/bom/OptimisationNotificationTypes.hpp File Reference	1036
35.360optimisationNotificationTypes.hpp	1036
35.361tdair/bom/ParsedKey.cpp File Reference	1036
35.362arsedKey.cpp	1037
35.363tdair/bom/ParsedKey.hpp File Reference	1039
35.364arsedKey.hpp	1040
35.365tdair/bom/PeriodStruct.cpp File Reference	1041
35.366eriodStruct.cpp	1041
35.367tdair/bom/PeriodStruct.hpp File Reference	1042
35.368eriodStruct.hpp	1043

35.369tdair/bom/PosChannel.cpp File Reference	1044
35.370PosChannel.cpp	1044
35.371tdair/bom/PosChannel.hpp File Reference	1045
35.372PosChannel.hpp	1045
35.373tdair/bom/PosChannelKey.cpp File Reference	1047
35.374PosChannelKey.cpp	1047
35.375tdair/bom/PosChannelKey.hpp File Reference	1048
35.376PosChannelKey.hpp	1048
35.377tdair/bom/PosChannelTypes.hpp File Reference	1049
35.378PosChannelTypes.hpp	1050
35.379tdair/bom/RMEventStruct.cpp File Reference	1050
35.380RMEventStruct.cpp	1050
35.381tdair/bom/RMEventStruct.hpp File Reference	1051
35.382RMEventStruct.hpp	1052
35.383tdair/bom/RMEventTypes.hpp File Reference	1053
35.384RMEventTypes.hpp	1053
35.385tdair/bom/SegmentCabin.cpp File Reference	1054
35.386SegmentCabin.cpp	1054
35.387tdair/bom/SegmentCabin.hpp File Reference	1055
35.388SegmentCabin.hpp	1056
35.389tdair/bom/SegmentCabinKey.cpp File Reference	1060
35.390SegmentCabinKey.cpp	1060
35.391tdair/bom/SegmentCabinKey.hpp File Reference	1062
35.392SegmentCabinKey.hpp	1062
35.393tdair/bom/SegmentCabinTypes.hpp File Reference	1063
35.394SegmentCabinTypes.hpp	1064
35.395tdair/bom/SegmentDate.cpp File Reference	1064
35.396SegmentDate.cpp	1064
35.397tdair/bom/SegmentDate.hpp File Reference	1066
35.398SegmentDate.hpp	1066
35.399tdair/bom/SegmentDateKey.cpp File Reference	1069
35.400SegmentDateKey.cpp	1070
35.401tdair/bom/SegmentDateKey.hpp File Reference	1071
35.402SegmentDateKey.hpp	1072

35.403	stdair/bom/SegmentDateTypes.hpp File Reference	1073
35.404	SegmentDateTypes.hpp	1073
35.405	stdair/bom/SegmentPeriod.cpp File Reference	1074
35.406	SegmentPeriod.cpp	1074
35.407	stdair/bom/SegmentPeriod.hpp File Reference	1075
35.408	SegmentPeriod.hpp	1075
35.409	stdair/bom/SegmentPeriodKey.cpp File Reference	1077
35.410	SegmentPeriodKey.cpp	1077
35.411	stdair/bom/SegmentPeriodKey.hpp File Reference	1078
35.412	SegmentPeriodKey.hpp	1078
35.413	stdair/bom/SegmentPeriodTypes.hpp File Reference	1079
35.414	SegmentPeriodTypes.hpp	1080
35.415	stdair/bom/SnapshotStruct.cpp File Reference	1080
35.416	SnapshotStruct.cpp	1080
35.417	stdair/bom/SnapshotStruct.hpp File Reference	1081
35.418	SnapshotStruct.hpp	1082
35.419	stdair/bom/SnapshotTypes.hpp File Reference	1083
35.420	SnapshotTypes.hpp	1083
35.421	stdair/bom/TimePeriod.cpp File Reference	1083
35.422	TimePeriod.cpp	1084
35.423	stdair/bom/TimePeriod.hpp File Reference	1085
35.424	TimePeriod.hpp	1085
35.425	stdair/bom/TimePeriodKey.cpp File Reference	1087
35.426	TimePeriodKey.cpp	1087
35.427	stdair/bom/TimePeriodKey.hpp File Reference	1088
35.428	TimePeriodKey.hpp	1088
35.429	stdair/bom/TimePeriodTypes.hpp File Reference	1089
35.430	TimePeriodTypes.hpp	1090
35.431	stdair/bom/TravelSolutionStruct.cpp File Reference	1090
35.432	TravelSolutionStruct.cpp	1091
35.433	stdair/bom/TravelSolutionStruct.hpp File Reference	1093
35.434	TravelSolutionStruct.hpp	1094
35.435	stdair/bom/TravelSolutionTypes.hpp File Reference	1096
35.436	TravelSolutionTypes.hpp	1096

35.43	stdair/bom/VirtualClassStruct.cpp File Reference	1097
35.43	VirtualClassStruct.cpp	1098
35.43	stdair/bom/VirtualClassStruct.hpp File Reference	1099
35.44	VirtualClassStruct.hpp	1099
35.44	stdair/bom/VirtualClassTypes.hpp File Reference	1101
35.44	VirtualClassTypes.hpp	1101
35.44	stdair/bom/YieldFeatures.cpp File Reference	1102
35.44	YieldFeatures.cpp	1102
35.44	stdair/bom/YieldFeatures.hpp File Reference	1103
35.44	YieldFeatures.hpp	1104
35.44	stdair/bom/YieldFeaturesKey.cpp File Reference	1105
35.44	YieldFeaturesKey.cpp	1105
35.44	stdair/bom/YieldFeaturesKey.hpp File Reference	1106
35.45	YieldFeaturesKey.hpp	1107
35.45	stdair/bom/YieldFeaturesTypes.hpp File Reference	1108
35.45	YieldFeaturesTypes.hpp	1108
35.45	stdair/bom/YieldStore.cpp File Reference	1109
35.45	YieldStore.cpp	1109
35.45	stdair/bom/YieldStore.hpp File Reference	1109
35.45	YieldStore.hpp	1110
35.45	stdair/bom/YieldStoreKey.cpp File Reference	1111
35.45	YieldStoreKey.cpp	1111
35.45	stdair/bom/YieldStoreKey.hpp File Reference	1112
35.46	YieldStoreKey.hpp	1112
35.46	stdair/bom/YieldStoreTypes.hpp File Reference	1113
35.46	YieldStoreTypes.hpp	1113
35.46	stdair/command/CmdAbstract.cpp File Reference	1114
35.46	CmdAbstract.cpp	1114
35.46	stdair/command/CmdAbstract.hpp File Reference	1114
35.46	CmdAbstract.hpp	1114
35.46	stdair/command/CmdBomManager.cpp File Reference	1115
35.46	CmdBomManager.cpp	1115
35.46	stdair/command/CmdBomManager.hpp File Reference	1157
35.47	CmdBomManager.hpp	1157

35.471	stdair/command/CmdBomSerialiser.cpp File Reference	1158
35.472	CmdBomSerialiser.cpp	1160
35.473	stdair/command/CmdBomSerialiser.hpp File Reference	1163
35.474	CmdBomSerialiser.hpp	1164
35.475	stdair/command/DBManagerForAirlines.cpp File Reference	1164
35.476	DBManagerForAirlines.cpp	1165
35.477	stdair/command/DBManagerForAirlines.hpp File Reference	1167
35.478	DBManagerForAirlines.hpp	1167
35.479	stdair/config/stdair-paths.hpp File Reference	1168
35.479	Define Documentation	1169
35.480	stdair-paths.hpp	1170
35.481	stdair/dbadaptor/DbAbstract.cpp File Reference	1171
35.482	DbAbstract.cpp	1171
35.483	stdair/dbadaptor/DbAbstract.hpp File Reference	1171
35.483	Function Documentation	1172
35.484	DbAbstract.hpp	1172
35.485	stdair/dbadaptor/DbAirline.cpp File Reference	1173
35.486	DbAirline.cpp	1173
35.487	stdair/dbadaptor/DbAirline.hpp File Reference	1174
35.488	DbAirline.hpp	1175
35.489	stdair/factory/FacAbstract.cpp File Reference	1175
35.490	FacAbstract.cpp	1175
35.491	stdair/factory/FacAbstract.hpp File Reference	1176
35.492	FacAbstract.hpp	1176
35.493	stdair/factory/FacBom.hpp File Reference	1176
35.494	FacBom.hpp	1177
35.495	stdair/factory/FacBomManager.hpp File Reference	1179
35.496	FacBomManager.hpp	1179
35.497	stdair/service/DBSessionManager.cpp File Reference	1185
35.498	DBSessionManager.cpp	1185
35.499	stdair/service/DBSessionManager.hpp File Reference	1187
35.500	DBSessionManager.hpp	1187
35.501	stdair/service/FacServiceAbstract.cpp File Reference	1188
35.502	FacServiceAbstract.cpp	1188

35.503tdair/service/FacServiceAbstract.hpp File Reference	1189
35.504FacServiceAbstract.hpp	1189
35.505tdair/service/FacSTDAIRServiceContext.cpp File Reference	1190
35.506FacSTDAIRServiceContext.cpp	1190
35.507tdair/service/FacSTDAIRServiceContext.hpp File Reference	1191
35.508FacSTDAIRServiceContext.hpp	1191
35.509tdair/service/FacSupervisor.cpp File Reference	1192
35.510FacSupervisor.cpp	1192
35.511tdair/service/FacSupervisor.hpp File Reference	1194
35.512FacSupervisor.hpp	1194
35.513tdair/service/Logger.cpp File Reference	1195
35.514Logger.cpp	1195
35.515tdair/service/Logger.hpp File Reference	1197
35.515. Define Documentation	1197
35.516Logger.hpp	1198
35.517tdair/service/ServiceAbstract.cpp File Reference	1200
35.518ServiceAbstract.cpp	1200
35.519tdair/service/ServiceAbstract.hpp File Reference	1200
35.519. Function Documentation	1201
35.520ServiceAbstract.hpp	1201
35.521tdair/service/STDAIR_Service.cpp File Reference	1202
35.522STDAIR_Service.cpp	1203
35.523tdair/service/STDAIR_ServiceContext.cpp File Reference	1213
35.524STDAIR_ServiceContext.cpp	1213
35.525tdair/service/STDAIR_ServiceContext.hpp File Reference	1214
35.526STDAIR_ServiceContext.hpp	1215
35.527tdair/stdair_basic_types.hpp File Reference	1216
35.528tdair_basic_types.hpp	1218
35.529tdair/stdair_date_time_types.hpp File Reference	1219
35.530tdair_date_time_types.hpp	1220
35.531tdair/stdair_db.hpp File Reference	1220
35.532tdair_db.hpp	1221
35.533tdair/stdair_demand_types.hpp File Reference	1221
35.534tdair_demand_types.hpp	1223

35.535tdair/stdair_event_types.hpp File Reference	1224
35.536tdair_event_types.hpp	1224
35.537tdair/stdair_exceptions.hpp File Reference	1225
35.538tdair_exceptions.hpp	1226
35.539tdair/stdair_fare_types.hpp File Reference	1228
35.540tdair_fare_types.hpp	1228
35.541tdair/stdair_file.hpp File Reference	1229
35.542tdair_file.hpp	1229
35.543tdair/stdair_inventory_types.hpp File Reference	1230
35.544tdair_inventory_types.hpp	1232
35.545tdair/stdair_log.hpp File Reference	1234
35.546tdair_log.hpp	1234
35.547tdair/stdair_maths_types.hpp File Reference	1235
35.548tdair_maths_types.hpp	1236
35.549tdair/stdair_rm_types.hpp File Reference	1237
35.550tdair_rm_types.hpp	1237
35.551tdair/STDAIR_Service.hpp File Reference	1238
35.552STDAIR_Service.hpp	1238
35.553tdair/stdair_service_types.hpp File Reference	1241
35.554tdair_service_types.hpp	1241
35.555tdair/stdair_types.hpp File Reference	1241
35.556tdair_types.hpp	1242
35.557tdair/ui/cmdline/readline_autocomp.hpp File Reference	1242
35.557.1Typedef Documentation	1243
35.557.2Function Documentation	1243
35.557.3Variable Documentation	1246
35.558readline_autocomp.hpp	1247
35.559tdair/ui/cmdline/SReadline.hpp File Reference	1252
35.559.1Detailed Description	1253
35.560SReadline.hpp	1253
35.561test/stdair/MPBomRoot.cpp File Reference	1260
35.562MPBomRoot.cpp	1260
35.563test/stdair/MPBomRoot.hpp File Reference	1261
35.564MPBomRoot.hpp	1261

35.565test/stdair/MPInventory.cpp File Reference	1261
35.566MPInventory.cpp	1262
35.567test/stdair/MPInventory.hpp File Reference	1262
35.568MPInventory.hpp	1262
35.569test/stdair/StandardAirlineITTestSuite.cpp File Reference	1263
35.570StandardAirlineITTestSuite.cpp	1263
35.571test/stdair/StdairTestLib.hpp File Reference	1268
35.572StdairTestLib.hpp	1268

1 StdAir Documentation

1.1 Getting Started

- [Main features](#)
- [Installation](#)
- [Linking with StdAir](#)
- [Users Guide](#)
- [Tutorials](#)
- [Copyright and License](#)
- [Make a Difference](#)
- [Make a new release](#)
- [People](#)

1.2 StdAir at SourceForge

- [Project page](#)
- [Download StdAir](#)
- [Open a ticket for a bug or feature](#)
- [Mailing lists](#)
- [Forums](#)
 - [Discuss about Development issues](#)
 - [Ask for Help](#)
 - [Discuss StdAir](#)

1.3 StdAir Development

- [Git Repository](#) (Subversion is deprecated)
- [Coding Rules](#)
- [Documentation Rules](#)
- [Test Rules](#)

1.4 External Libraries

- [Boost](#) (C++ STL extensions)
- [ZeroMQ](#) (networking made easy)
- [Python](#)
- [MySQL client](#)
- [SOI](#) (C++ DB API)

1.5 Support StdAir

1.6 About StdAir

StdAir is a C++ library of classes and functions modeling typical airline IT business objects. For instance, it is used by the C++ Revenue Management Open Library project (<http://sourceforge.net/projects/rmol/>). StdAir mainly targets simulation purposes. [N](#)

StdAir makes an extensive use of existing open-source libraries for increased functionality, speed and accuracy. In particular [Boost](#) (*C++ STL Extensions*) library is used.

The StdAir library originates from the department of Operational Research and Innovation at [Amadeus](#), Sophia Antipolis, France. StdAir is released under the terms of the [GNU Lesser General Public License](#) (LGPLv2.1) for you to enjoy.

StdAir should work on [GNU/Linux](#), [Sun Solaris](#), Microsoft Windows (with [Cygwin](#), [MinGW/MSYS](#), or [Microsoft Visual C++ .NET](#)) and [Mac OS X](#) operating systems.

Note

(N) - The StdAir library is **NOT** intended, in any way, to be used by airlines for production systems. If you want to report issue, bug or feature request, or if you just want to give feedback, have a look on the right-hand side of this page for the preferred reporting methods. In any case, please do not contact Amadeus directly for any matter related to StdAir.

2 C++ Utility Class Browsing and Dumping the StdAir BOM Tree

```

*/
// ////////////////////////////////////////
// Import section
// ////////////////////////////////////////
// STL
#include <cassert>
#include <ostream>
// StdAir
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/EventQueue.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/AirportPair.hpp>
#include <stdair/bom/PosChannel.hpp>
#include <stdair/bom/DatePeriod.hpp>
#include <stdair/bom/TimePeriod.hpp>
#include <stdair/bom/FareFeatures.hpp>
#include <stdair/bom/YieldFeatures.hpp>
#include <stdair/bom/AirlineClassList.hpp>
#include <stdair/bom/Bucket.hpp>
#include <stdair/bom/TravelSolutionTypes.hpp>
#include <stdair/bom/TravelSolutionStruct.hpp>
#include <stdair/bom/BomDisplay.hpp>
#include <stdair/bom/OnDDate.hpp>

namespace stdair {

    struct FlagSaver {
    public:
        FlagSaver (std::ostream& oStream)
            : _oStream (oStream), _streamFlags (oStream.flags()) {
        }

        ~FlagSaver() {
            // Reset formatting flags of the given output stream
            _oStream.flags (_streamFlags);
        }

    private:
        std::ostream& _oStream;
        std::ios::fmtflags _streamFlags;
    };

// ////////////////////////////////////////
std::string BomDisplay::csvDisplay (const EventQueue& iEventQueue) {
    std::ostringstream oStream;

    oStream << std::endl;
    oStream << "=====
    << std::endl;
    oStream << "EventQueue: " << iEventQueue.describeKey() << std::endl;
    oStream << "=====

```

```

        << std::endl;

    return oStream.str();
}

// ////////////////////////////////////////
void BomDisplay::list (std::ostream& oStream, const BomRoot& iBomRoot,
                     const AirlineCode_T& iAirlineCode,
                     const FlightNumber_T& iFlightNumber) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    // Check whether there are Inventory objects
    if (BomManager::hasList<Inventory> (iBomRoot) == false) {
        return;
    }

    // Browse the inventories
    unsigned short invIdx = 1;
    const InventoryList_T& lInventoryList =
        BomManager::getList<Inventory> (iBomRoot);
    for (InventoryList_T::const_iterator itInv = lInventoryList.begin();
         itInv != lInventoryList.end(); ++itInv, ++invIdx) {
        const Inventory* lInv_ptr = *itInv;
        assert (lInv_ptr != NULL);

        // Retrieve the inventory key (airline code)
        const AirlineCode_T& lAirlineCode = lInv_ptr->getAirlineCode();

        // Display only the requested inventories
        if (iAirlineCode == "all" || iAirlineCode == lAirlineCode) {
            // Get the list of flight-dates for that inventory
            list (oStream, *lInv_ptr, invIdx, iFlightNumber);
        }
    }
}

// ////////////////////////////////////////
void BomDisplay::list (std::ostream& oStream, const Inventory& iInventory,
                     const unsigned short iInventoryIndex,
                     const FlightNumber_T& iFlightNumber) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    // Check whether there are FlightDate objects
    if (BomManager::hasMap<FlightDate> (iInventory) == false) {
        return;
    }

    //
    const AirlineCode_T& lAirlineCode = iInventory.getAirlineCode();
    oStream << iInventoryIndex << ". " << lAirlineCode << std::endl;

    // Browse the flight-dates
    unsigned short lCurrentFlightNumber = 0;
    unsigned short flightNumberIdx = 0;
    unsigned short departureDateIdx = 1;
    const FlightDateMap_T& lFlightDateList =
        BomManager::getMap<FlightDate> (iInventory);
    for (FlightDateMap_T::const_iterator itFD = lFlightDateList.begin();
         itFD != lFlightDateList.end(); ++itFD, ++departureDateIdx) {
        const FlightDate* lFD_ptr = itFD->second;

```

```

    assert (lFD_ptr != NULL);

    // Retrieve the key of the flight-date
    const FlightNumber_T& lFlightNumber = lFD_ptr->getFlightNumber();
    const Date_T& lFlightDateDate = lFD_ptr->getDepartureDate();

    // Display only the requested flight number
    if (iFlightNumber == 0 || iFlightNumber == lFlightNumber) {
        //
        if (lCurrentFlightNumber != lFlightNumber) {
            lCurrentFlightNumber = lFlightNumber;
            ++flightNumberIdx; departureDateIdx = 1;
            ostream << " " << iInventoryIndex << "." << flightNumberIdx << ". "
                << lAirlineCode << lFlightNumber << std::endl;
        }

        ostream << " " << iInventoryIndex << "." << flightNumberIdx
            << "." << departureDateIdx << ". "
            << lAirlineCode << lFlightNumber << " / " << lFlightDateDate
            << std::endl;
    }
}

// ////////////////////////////////////////
void BomDisplay::listAirportPairDateRange (std::ostream& ostream,
                                           const BomRoot& iBomRoot) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (ostream);

    // Check whether there are AirportPair objects
    if (BomManager::hasList<AirportPair> (iBomRoot) == false) {
        return;
    }

    const AirportPairList_T& lAirportPairList =
        BomManager::getList<AirportPair> (iBomRoot);
    for (AirportPairList_T::const_iterator itAir = lAirportPairList.begin();
         itAir != lAirportPairList.end(); ++itAir) {
        const AirportPair* lAir_ptr = *itAir;
        assert (lAir_ptr != NULL);

        // Check whether there are date-period objects
        assert (BomManager::hasList<DatePeriod> (*lAir_ptr) == true);

        // Browse the date-period objects
        const DatePeriodList_T& lDatePeriodList =
            BomManager::getList<DatePeriod> (*lAir_ptr);

        for (DatePeriodList_T::const_iterator itDP = lDatePeriodList.begin();
             itDP != lDatePeriodList.end(); ++itDP) {
            const DatePeriod* lDP_ptr = *itDP;
            assert (lDP_ptr != NULL);

            // Display the date-period object
            ostream << lAir_ptr->describeKey()
                << " / " << lDP_ptr->describeKey() << std::endl;
        }
    }
}

```

```

// //////////////////////////////////////
void BomDisplay::csvDisplay (std::ostream& oStream,
                           const BomRoot& iBomRoot) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    oStream << std::endl;
    oStream << "====="
              << std::endl;
    oStream << "BomRoot: " << iBomRoot.describeKey() << std::endl;
    oStream << "====="
              << std::endl;

    // Check whether there are Inventory objects
    if (BomManager::hasList<Inventory> (iBomRoot) == false) {
        return;
    }

    // Browse the inventories
    const InventoryList_T& lInventoryList =
        BomManager::getList<Inventory> (iBomRoot);
    for (InventoryList_T::const_iterator itInv = lInventoryList.begin();
         itInv != lInventoryList.end(); ++itInv) {
        const Inventory* lInv_ptr = *itInv;
        assert (lInv_ptr != NULL);

        // Display the inventory
        csvDisplay (oStream, *lInv_ptr);
    }
}

// //////////////////////////////////////
void BomDisplay::csvDisplay (std::ostream& oStream,
                           const Inventory& iInventory) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    oStream << "+++++" << std::endl;
    oStream << "Inventory: " << iInventory.describeKey() << std::endl;
    oStream << "+++++" << std::endl;

    // Check whether there are FlightDate objects
    if (BomManager::hasList<FlightDate> (iInventory) == false) {
        return;
    }

    // Browse the flight-dates
    const FlightDateList_T& lFlightDateList =
        BomManager::getList<FlightDate> (iInventory);
    for (FlightDateList_T::const_iterator itFD = lFlightDateList.begin();
         itFD != lFlightDateList.end(); ++itFD) {
        const FlightDate* lFD_ptr = *itFD;
        assert (lFD_ptr != NULL);

        // Display the flight-date
        csvDisplay (oStream, *lFD_ptr);
    }

    // Check if the inventory contains a list of partners

    if (BomManager::hasList<Inventory> (iInventory)){

```

```

// Browse the partner's inventories
const InventoryList_T& lPartnerInventoryList =
    BomManager::getList<Inventory> (iInventory);

for (InventoryList_T::const_iterator itInv = lPartnerInventoryList.begin();
     itInv != lPartnerInventoryList.end(); ++itInv) {

    oStream << "-----" << std::endl;
    oStream << "Partner inventory:" << std::endl;
    oStream << "-----" << std::endl;
    const Inventory* lInv_ptr = *itInv;
    assert (lInv_ptr != NULL);

    // Display the inventory
    csvDisplay (oStream, *lInv_ptr);
}
oStream << "*****" << std::endl;
oStream << std::endl;
}

// Check if the inventory contains a list of O&D dates
if (BomManager::hasList<OnDDate> (iInventory)){

    //Browse the O&Ds
    const OnDDateList_T& lOnDDateList =
        BomManager::getList<OnDDate> (iInventory);

    for (OnDDateList_T::const_iterator itOnD = lOnDDateList.begin();
         itOnD != lOnDDateList.end(); ++itOnD) {
        oStream << "*****" << std::endl;
        oStream << "O&D-Date:" << std::endl;
        oStream << "-----" << std::endl;
        oStream << "Airline, Date, Origin-Destination, Segments, " << std::endl;

        const OnDDate* lOnDDate_ptr = *itOnD;
        assert (lOnDDate_ptr != NULL);

        // Display the O&D date
        csvDisplay (oStream, *lOnDDate_ptr);
    }
    oStream << "*****" << std::endl;
}

// ////////////////////////////////////////
void BomDisplay::csvDisplay (std::ostream& oStream,
                             const OnDDate& iOnDDate) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    const AirlineCode_T& lAirlineCode = iOnDDate.getAirlineCode();
    const Date_T& lDate = iOnDDate.getDate();
    const AirportCode_T& lOrigin = iOnDDate.getOrigin();
    const AirportCode_T& lDestination = iOnDDate.getDestination();

    oStream << lAirlineCode << ", " << lDate << ", " << lOrigin << "-"
        << lDestination << ", " << iOnDDate.describeKey() << ", "
        << std::endl;
}

```

```

const StringDemandStructMap_T& lDemandInfoMap =
    iOnDDate.getDemandInfoMap();

// Check if the map contains information.
const bool isInfoMapEmpty = lDemandInfoMap.empty();
if (isInfoMapEmpty) {
    return;
}
assert (lDemandInfoMap.empty() ==false);

oStream << "-----" << std::endl;
oStream << "Cabin-Class path, Demand mean, Demand std dev, Yield, "
    << std::endl;

for (StringDemandStructMap_T::const_iterator itDI = lDemandInfoMap.begin();
    itDI != lDemandInfoMap.end(); ++itDI) {

    const std::string& lCabinClassPath = itDI->first;
    const YieldDemandPair_T lYieldDemandPair =
        itDI->second;
    const Yield_T lYield = lYieldDemandPair.first;
    const MeanStdDevPair_T lMeanStdDevPair =
        lYieldDemandPair.second;
    const MeanValue_T lDemandMean = lMeanStdDevPair.first;
    const StdDevValue_T lDemandStdDev = lMeanStdDevPair.second;

    oStream << lCabinClassPath << ", "
        << lDemandMean << ", "
        << lDemandStdDev << ", "
        << lYield << ", "
        << std::endl;
}

}

// ////////////////////////////////////////
void BomDisplay::csvDisplay (std::ostream& oStream,
    const FlightDate& iFlightDate) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    const AirlineCode_T& lAirlineCode = iFlightDate.getAirlineCode();
    oStream << "*****" << std::endl;
    oStream << "FlightDate: " << lAirlineCode << iFlightDate.describeKey()
        << std::endl;
    oStream << "*****" << std::endl;

    //
    csvSegmentDateDisplay (oStream, iFlightDate);
    //
    csvLegDateDisplay (oStream, iFlightDate);

    //
    csvLegCabinDisplay (oStream, iFlightDate);

    //
    csvBucketDisplay (oStream, iFlightDate);

    //
    csvFareFamilyDisplay (oStream, iFlightDate);

```

```

//
csvBookingClassDisplay (oStream, iFlightDate);
}

// ////////////////////////////////////////
void BomDisplay::csvLegDateDisplay (std::ostream& oStream,
                                   const FlightDate& iFlightDate) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    oStream << "*****" << std::endl;
    oStream << "Leg-Dates:" << std::endl
    << "-----" << std::endl;
    oStream << "Flight, Leg, BoardDate, BoardTime, "
    << "OffDate, OffTime, Date Offset, Time Offset, Elapsed, "
    << "Distance, Capacity, " << std::endl;

    // Retrieve the key of the flight-date
    const AirlineCode_T& lAirlineCode = iFlightDate.getAirlineCode();
    const FlightNumber_T& lFlightNumber = iFlightDate.getFlightNumber();
    const Date_T& lFlightDateDate = iFlightDate.getDepartureDate();

    // Check whether there are LegDate objects
    if (BomManager::hasList<LegDate> (iFlightDate) == false) {
        return;
    }

    // Browse the leg-dates
    const LegDateList_T& lLegDateList =
        BomManager::getList<LegDate> (iFlightDate);
    for (LegDateList_T::const_iterator itLD = lLegDateList.begin();
         itLD != lLegDateList.end(); ++itLD) {
        const LegDate* lLD_ptr = *itLD;
        assert (lLD_ptr != NULL);

        oStream << lAirlineCode << lFlightNumber << " "
        << lFlightDateDate << ", ";

        oStream << lLD_ptr->getBoardingPoint() << "-"
        << lLD_ptr->getOffPoint() << ", "
        << lLD_ptr->getBoardingDate() << ", "
        << lLD_ptr->getBoardingTime() << ", "
        << lLD_ptr->getOffDate() << ", "
        << lLD_ptr->getOffTime() << ", "
        << lLD_ptr->getElapsedTime() << ", "
        << lLD_ptr->getDateOffset().days() << ", "
        << lLD_ptr->getTimeOffset() << ", "
        << lLD_ptr->getDistance() << ", "
        << lLD_ptr->getCapacity() << ", " << std::endl;
    }
    oStream << "*****" << std::endl;
}

// ////////////////////////////////////////
void BomDisplay::csvSegmentDateDisplay (std::ostream& oStream,
                                        const FlightDate& iFlightDate) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    oStream << "*****" << std::endl;
    oStream << "SegmentDates:" << std::endl
    << "-----" << std::endl;

```

```

oStream << "Flight, Segment, Date"
        << std::endl;

// Retrieve the key of the flight-date
const AirlineCode_T& lAirlineCode = iFlightDate.getAirlineCode();
const FlightNumber_T& lFlightNumber = iFlightDate.getFlightNumber();
const Date_T& lFlightDateDate = iFlightDate.getDepartureDate();

// Check whether there are SegmentDate objects
if (BomManager::hasList<SegmentDate> (iFlightDate) == false) {
    return;
}

// Browse the segment-dates
const SegmentDateList_T& lSegmentDateList =
    BomManager::getList<SegmentDate> (iFlightDate);
for (SegmentDateList_T::const_iterator itSD = lSegmentDateList.begin();
     itSD != lSegmentDateList.end(); ++itSD) {
    const SegmentDate* lSD_ptr = *itSD;
    assert (lSD_ptr != NULL);

    // Retrieve the key of the segment-date, as well as its dates
    const Date_T& lSegmentDateDate = lSD_ptr->getBoardingDate();
    const AirportCode_T& lBoardPoint = lSD_ptr->getBoardingPoint();
    const AirportCode_T& lOffPoint = lSD_ptr->getOffPoint();
    oStream << lAirlineCode << lFlightNumber << " " << lFlightDateDate << ", "
            << lBoardPoint << "-" << lOffPoint << ", " << lSegmentDateDate << s
td::endl;

    // Check if the current segment has corresponding marketing segments.
    const bool isMarketingSDListEmpty = BomManager::hasList<SegmentDate>(*lSD_ptr);
    if (isMarketingSDListEmpty == false) {
        //
        const SegmentDateList_T& lMarketingSDList = BomManager::getList<SegmentDate>(*lSD_ptr);

        oStream << " *** Marketed by ";
        for (SegmentDateList_T::const_iterator itMarketingSD = lMarketingSDList.begin();
             itMarketingSD != lMarketingSDList.end(); ++itMarketingSD) {
            SegmentDate* lMarketingSD_ptr = *itMarketingSD;
            FlightDate* lMarketingFD_ptr = BomManager::getParentPtr<FlightDate>(*lMarketingSD_ptr);
            Inventory* lMarketingInv_ptr = BomManager::getParentPtr<Inventory>(*lMarketingFD_ptr);
            oStream << lMarketingInv_ptr->toString() << lMarketingFD_ptr->toString()
                << " * ";
        }

        // Check if the current segment is operated by another segment date.
        const SegmentDate* lOperatingSD_ptr = lSD_ptr->getOperatingSegmentDate ();
        if (lOperatingSD_ptr != NULL) {
            const FlightDate* lOperatingFD_ptr = BomManager::getParentPtr<FlightDate>(*lOperatingSD_ptr);
            const Inventory* lOperatingInv_ptr = BomManager::getParentPtr<Inventory>(*lOperatingFD_ptr);
            oStream << " *** Operated by " << lOperatingInv_ptr->toString()
                << lOperatingFD_ptr->toString() << std::endl;
        }
    }
}

```

```

        oStream << std::endl;
    }
}

// ////////////////////////////////////////
void BomDisplay::csvLegCabinDisplay (std::ostream& oStream,
                                     const FlightDate& iFlightDate) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    oStream << "*****" << std::endl;
    oStream << "LegCabins:" << std::endl
              << "-----" << std::endl;
    oStream << "Flight, Leg, Cabin, "
              << "OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, "
              << "CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice, "
              << std::endl;

    // Retrieve the key of the flight-date
    const AirlineCode_T& lAirlineCode = iFlightDate.getAirlineCode();
    const FlightNumber_T& lFlightNumber = iFlightDate.getFlightNumber();
    const Date_T& lFlightDateDate = iFlightDate.getDepartureDate();

    // Check whether there are LegDate objects
    if (BomManager::hasList<LegDate> (iFlightDate) == false) {
        return;
    }

    // Browse the leg-dates
    const LegDateList_T& lLegDateList =
        BomManager::getList<LegDate> (iFlightDate);
    for (LegDateList_T::const_iterator itLD = lLegDateList.begin();
         itLD != lLegDateList.end(); ++itLD) {
        const LegDate* lLD_ptr = *itLD;
        assert (lLD_ptr != NULL);

        // Retrieve the key of the leg-date, as well as its off point
        const Date_T& lLegDateDate = lLD_ptr->getBoardingDate();
        const AirportCode_T& lBoardPoint = lLD_ptr->getBoardingPoint();
        const AirportCode_T& lOffPoint = lLD_ptr->getOffPoint();

        // Browse the leg-cabins
        const LegCabinList_T& lLegCabinList =
            BomManager::getList<LegCabin> (*lLD_ptr);
        for (LegCabinList_T::const_iterator itLC = lLegCabinList.begin();
             itLC != lLegCabinList.end(); ++itLC) {
            const LegCabin* lLC_ptr = *itLC;
            assert (lLC_ptr != NULL);

            oStream << lAirlineCode << lFlightNumber << " "
                    << lFlightDateDate << ", ";

            oStream << lBoardPoint << "-" << lOffPoint
                    << " " << lLegDateDate << ", ";

            oStream << lLC_ptr->getCabinCode() << ", ";

            oStream << lLC_ptr->getOfferedCapacity() << ", "
                    << lLC_ptr->getPhysicalCapacity() << ", "
                    << lLC_ptr->getRegradeAdjustment() << ", "
                    << lLC_ptr->getAuthorizationLevel() << ", "

```

```

        << lLC_ptr->getUPR() << ", "
        << lLC_ptr->getSoldSeat() << ", "
        << lLC_ptr->getStaffNbOfSeats() << ", "
        << lLC_ptr->getWLNbOfSeats() << ", "
        << lLC_ptr->getGroupNbOfSeats() << ", "
        << lLC_ptr->getCommittedSpace() << ", "
        << lLC_ptr->getAvailabilityPool() << ", "
        << lLC_ptr->getAvailability() << ", "
        << lLC_ptr->getNetAvailability() << ", "
        << lLC_ptr->getGrossAvailability() << ", "
        << lLC_ptr->getAvgCancellationPercentage() << ", "
        << lLC_ptr->getETB() << ", "
        << lLC_ptr->getCurrentBidPrice() << ", "
        << std::endl;
    }
}
oStream << "*****" << std::endl;
}

// ////////////////////////////////////////
void BomDisplay::csvSegmentCabinDisplay (std::ostream& oStream,
                                         const FlightDate& iFlightDate) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

}

// ////////////////////////////////////////
void BomDisplay::csvFareFamilyDisplay (std::ostream& oStream,
                                       const FlightDate& iFlightDate) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    oStream << "*****" << std::endl;
    oStream << "SegmentCabins:" << std::endl
        << "-----" << std::endl;
    oStream << "Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, "
        << "CommSpace, AvPool, BP, " << std::endl;

    // Retrieve the key of the flight-date
    const AirlineCode_T& lAirlineCode = iFlightDate.getAirlineCode();
    const FlightNumber_T& lFlightNumber = iFlightDate.getFlightNumber();
    const Date_T& lFlightDateDate = iFlightDate.getDepartureDate();

    // Check whether there are SegmentDate objects
    if (BomManager::hasList<SegmentDate> (iFlightDate) == false) {
        return;
    }

    // Browse the segment-dates
    const SegmentDateList_T& lSegmentDateList =
        BomManager::getList<SegmentDate> (iFlightDate);
    for (SegmentDateList_T::const_iterator itSD = lSegmentDateList.begin();
         itSD != lSegmentDateList.end(); ++itSD) {
        const SegmentDate* lSD_ptr = *itSD;
        assert (lSD_ptr != NULL);

        // Retrieve the key of the segment-date, as well as its dates
        const Date_T& lSegmentDateDate = lSD_ptr->getBoardingDate();
        const AirportCode_T& lBoardPoint = lSD_ptr->getBoardingPoint();
        const AirportCode_T& lOffPoint = lSD_ptr->getOffPoint();

```

```

// Browse the segment-cabins
const SegmentCabinList_T& lSegmentCabinList =
    BomManager::getList<SegmentCabin> (*lSD_ptr);
for (SegmentCabinList_T::const_iterator itSC = lSegmentCabinList.begin();
     itSC != lSegmentCabinList.end(); ++itSC) {
    const SegmentCabin* lSC_ptr = *itSC;
    assert (lSC_ptr != NULL);

    // Retrieve the key of the segment-cabin
    const CabinCode_T& lCabinCode = lSC_ptr->getCabinCode();

    // Check whether there are fare family objects
    if (BomManager::hasList<FareFamily> (*lSC_ptr) == false) {
        continue;
    }

    // Browse the fare families
    const FareFamilyList_T& lFareFamilyList =
        BomManager::getList<FareFamily> (*lSC_ptr);
    for (FareFamilyList_T::const_iterator itFF = lFareFamilyList.begin();
         itFF != lFareFamilyList.end(); ++itFF) {
        const FareFamily* lFF_ptr = *itFF;
        assert (lFF_ptr != NULL);

        ostream << lAirlineCode << lFlightNumber << " "
                  << lFlightDateDate << ", ";

        ostream << lBoardPoint << "-" << lOffPoint << " "
                  << lSegmentDateDate << ", ";

        ostream << lCabinCode << ", " << lFF_ptr->getFamilyCode() << ", ";

        ostream << lSC_ptr->getBookingCounter() << ", "
                  << lSC_ptr->getMIN() << ", "
                  << lSC_ptr->getUPR() << ", "
                  << lSC_ptr->getCommittedSpace() << ", "
                  << lSC_ptr->getAvailabilityPool() << ", "
                  << lSC_ptr->getCurrentBidPrice() << ", "
                  << std::endl;
    }
}
ostream << "*****" << std::endl;
}

// //////////////////////////////////////
void BomDisplay::csvBucketDisplay (std::ostream& ostream,
                                   const FlightDate& iFlightDate) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (ostream);

    ostream << "*****" << std::endl;
    ostream << "Buckets:" << std::endl;
    ostream << "-----" << std::endl;
    ostream << "Flight, Leg, Cabin, Yield, AU/SI, SS, AV, "
              << std::endl;

    // Retrieve the key of the flight-date
    const AirlineCode_T& lAirlineCode = iFlightDate.getAirlineCode();
    const FlightNumber_T& lFlightNumber = iFlightDate.getFlightNumber();
    const Date_T& lFlightDateDate = iFlightDate.getDepartureDate();

```

```

// Check whether there are LegDate objects
if (BomManager::hasList<LegDate> (iFlightDate) == false) {
    return;
}

// Browse the leg-dates
const LegDateList_T& lLegDateList =
    BomManager::getList<LegDate> (iFlightDate);
for (LegDateList_T::const_iterator itLD = lLegDateList.begin();
     itLD != lLegDateList.end(); ++itLD) {
    const LegDate* lLD_ptr = *itLD;
    assert (lLD_ptr != NULL);

    // Retrieve the key of the leg-date, as well as its off point
    const Date_T& lLegDateDate = lLD_ptr->getBoardingDate();
    const AirportCode_T& lBoardPoint = lLD_ptr->getBoardingPoint();
    const AirportCode_T& lOffPoint = lLD_ptr->getOffPoint();

    // Browse the leg-cabins
    const LegCabinList_T& lLegCabinList =
        BomManager::getList<LegCabin> (*lLD_ptr);
    for (LegCabinList_T::const_iterator itLC = lLegCabinList.begin();
         itLC != lLegCabinList.end(); ++itLC) {
        const LegCabin* lLC_ptr = *itLC;
        assert (lLC_ptr != NULL);

        // Check whether there are bucket objects
        if (BomManager::hasList<Bucket> (*lLC_ptr) == false) {
            continue;
        }

        // Retrieve the key of the leg-cabin
        const CabinCode_T& lCabinCode = lLC_ptr->getCabinCode();

        // Browse the buckets
        const BucketList_T& lBucketList = BomManager::getList<Bucket> (*lLC_ptr);

        for (BucketList_T::const_iterator itBuck = lBucketList.begin();
             itBuck != lBucketList.end(); ++itBuck) {
            const Bucket* lBucket_ptr = *itBuck;
            assert (lBucket_ptr != NULL);

            ostream << lAirlineCode << lFlightNumber << " "
                << lFlightDateDate << ", ";

            ostream << lBoardPoint << "-" << lOffPoint << " "
                << lLegDateDate << ", " << lCabinCode << ", ";

            ostream << lBucket_ptr->getYieldRangeUpperValue() << ", "
                << lBucket_ptr->getSeatIndex() << ", "
                << lBucket_ptr->getSoldSeats() << ", "
                << lBucket_ptr->getAvailability() << ", ";
            ostream << std::endl;
        }
    }
}
ostream << "*****" << std::endl;
}

// //////////////////////////////////////
void BomDisplay::csvBookingClassDisplay (std::ostream& ostream,
                                         const BookingClass& iBookingClass,

```

```

        const std::string& iLeadingString) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    oStream << iLeadingString << iBookingClass.getClassCode();

    if (iBookingClass.getSubclassCode() == 0) {
        oStream << ", ";
    } else {
        oStream << iBookingClass.getSubclassCode() << ", ";
    }
    oStream << iBookingClass.getAuthorizationLevel() << " ("
        << iBookingClass.getProtection() << "), "
        << iBookingClass.getNegotiatedSpace() << ", "
        << iBookingClass.getNoShowPercentage() << ", "
        << iBookingClass.getCancellationPercentage() << ", "
        << iBookingClass.getNbOfBookings() << ", "
        << iBookingClass.getNbOfGroupBookings() << " ("
        << iBookingClass.getNbOfPendingGroupBookings() << "), "
        << iBookingClass.getNbOfStaffBookings() << ", "
        << iBookingClass.getNbOfWLBookings() << ", "
        << iBookingClass.getETB() << ", "
        << iBookingClass.getNetClassAvailability() << ", "
        << iBookingClass.getNetRevenueAvailability() << ", "
        << iBookingClass.getSegmentAvailability() << ", "
        << std::endl;
}

// ////////////////////////////////////////
void BomDisplay::csvBookingClassDisplay (std::ostream& oStream,
        const FlightDate& iFlightDate) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    // Headers
    oStream << "*****" << std::endl;
    oStream << "Subclasses:" << std::endl
        << "-----" << std::endl;
    oStream << "Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), "
        << "Negot, NS%, OB%, "
        << "Bkgs, GrpBks (pdg), StfBkgs, WLBkgs, ETB, "
        << "ClassAvl, RevAvl, SegAvl, "
        << std::endl;

    // Retrieve the key of the flight-date
    const AirlineCode_T& lAirlineCode = iFlightDate.getAirlineCode();
    const FlightNumber_T& lFlightNumber = iFlightDate.getFlightNumber();
    const Date_T& lFlightDateDate = iFlightDate.getDepartureDate();

    // Check whether there are SegmentDate objects
    if (BomManager::hasList<SegmentDate> (iFlightDate) == false) {
        return;
    }

    // Browse the segment-dates
    const SegmentDateList_T& lSegmentDateList =
        BomManager::getList<SegmentDate> (iFlightDate);
    for (SegmentDateList_T::const_iterator itSD = lSegmentDateList.begin();
        itSD != lSegmentDateList.end(); ++itSD) {
        const SegmentDate* lSD_ptr = *itSD;
        assert (lSD_ptr != NULL);
    }
}

```

```

// Retrieve the key of the segment-date, as well as its dates
const Date_T& lSegmentDateDate = lSD_ptr->getBoardingDate();
const AirportCode_T& lBoardPoint = lSD_ptr->getBoardingPoint();
const AirportCode_T& lOffPoint = lSD_ptr->getOffPoint();

// Browse the segment-cabins
const SegmentCabinList_T& lSegmentCabinList =
    BomManager::getList<SegmentCabin> (*lSD_ptr);
for (SegmentCabinList_T::const_iterator itSC = lSegmentCabinList.begin();
     itSC != lSegmentCabinList.end(); ++itSC) {
    const SegmentCabin* lSC_ptr = *itSC;
    assert (lSC_ptr != NULL);

    // Retrieve the key of the segment-cabin
    const CabinCode_T& lCabinCode = lSC_ptr->getCabinCode();

    // Build the leading string to be displayed
    std::ostringstream oSCLeadingStr;
    oSCLeadingStr << lAirlineCode << lFlightNumber << " "
                  << lFlightDateDate << ", "
                  << lBoardPoint << "-" << lOffPoint << " "
                  << lSegmentDateDate << ", "
                  << lCabinCode << ", ";

    // Default Fare Family code, when there are no FF
    FamilyCode_T lFamilyCode ("NoFF");

    // Check whether there are FareFamily objects
    if (BomManager::hasList<FareFamily> (*lSC_ptr) == true) {

        // Browse the fare families
        const FareFamilyList_T& lFareFamilyList =
            BomManager::getList<FareFamily> (*lSC_ptr);
        for (FareFamilyList_T::const_iterator itFF = lFareFamilyList.begin();
             itFF != lFareFamilyList.end(); ++itFF) {
            const FareFamily* lFF_ptr = *itFF;
            assert (lFF_ptr != NULL);

            // Retrieve the key of the segment-cabin
            lFamilyCode = lFF_ptr->getFamilyCode();

            // Complete the leading string to be displayed
            std::ostringstream oFFLeadingStr;
            oFFLeadingStr << oSCLeadingStr.str() << lFamilyCode << ", ";

            // Browse the booking-classes
            const BookingClassList_T& lBookingClassList =
                BomManager::getList<BookingClass> (*lFF_ptr);
            for (BookingClassList_T::const_iterator itBC =
                 lBookingClassList.begin();
                 itBC != lBookingClassList.end(); ++itBC) {
                const BookingClass* lBC_ptr = *itBC;
                assert (lBC_ptr != NULL);

                //
                csvBookingClassDisplay (oStream, *lBC_ptr, oFFLeadingStr.str());
            }
        }

        // Go on to the next segment-cabin
        continue;
    }
}

```

```

    assert (BomManager::hasList<FareFamily> (*lSC_ptr) == false);

    // The fare family code is a fake one ('NoFF'), and therefore
    // does not vary
    std::ostringstream oFFLeadingStr;
    oFFLeadingStr << oSCLeadingStr.str() << lFamilyCode << ", ";

    // Browse the booking-classes, directly from the segment-cabin object
    const BookingClassList_T& lBookingClassList =
        BomManager::getList<BookingClass> (*lSC_ptr);
    for (BookingClassList_T::const_iterator itBC =
        lBookingClassList.begin();
        itBC != lBookingClassList.end(); ++itBC) {
        const BookingClass* lBC_ptr = *itBC;
        assert (lBC_ptr != NULL);

        //
        csvBookingClassDisplay (oStream, *lBC_ptr, oFFLeadingStr.str());
    }
}
oStream << "*****" << std::endl;
}

// ////////////////////////////////////////
void BomDisplay::
csvDisplay (std::ostream& oStream,
            const TravelSolutionList_T& iTravelSolutionList) {

    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    oStream << "Travel solutions:";
    unsigned short idx = 0;
    for (TravelSolutionList_T::const_iterator itTS =
        iTravelSolutionList.begin();
        itTS != iTravelSolutionList.end(); ++itTS, ++idx) {
        const TravelSolutionStruct& lTS = *itTS;

        oStream << std::endl;
        oStream << "    [" << idx << "] " << lTS.display();
    }
}

// ////////////////////////////////////////
void BomDisplay::
csvDisplay (std::ostream& oStream,
            const DatePeriodList_T& iDatePeriodList) {

    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    // Browse the date-period objects
    for (DatePeriodList_T::const_iterator itDP = iDatePeriodList.begin();
        itDP != iDatePeriodList.end(); ++itDP) {
        const DatePeriod* lDP_ptr = *itDP;
        assert (lDP_ptr != NULL);

        // Display the date-period object
        csvDateDisplay (oStream, *lDP_ptr);
    }
}

```

```

// //////////////////////////////////////
void BomDisplay::csvSimFQTAirRACDisplay (std::ostream& oStream,
                                         const BomRoot& iBomRoot) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    oStream << std::endl;
    oStream << "=====
    << std::endl;
    oStream << "BomRoot: " << iBomRoot.describeKey() << std::endl;
    oStream << "=====
    << std::endl;

    // Check whether there are airport-pair objects
    if (BomManager::hasList<AirportPair> (iBomRoot) == false) {
        return;
    }

    // Browse the airport-pair objects
    const AirportPairList_T& lAirportPairList =
        BomManager::getList<AirportPair> (iBomRoot);
    for (AirportPairList_T::const_iterator itAir = lAirportPairList.begin();
         itAir != lAirportPairList.end(); ++itAir) {
        const AirportPair* lAir_ptr = *itAir;
        assert (lAir_ptr != NULL);

        // Display the airport pair object
        csvAirportPairDisplay (oStream, *lAir_ptr);
    }
}

// //////////////////////////////////////
void BomDisplay::csvAirportPairDisplay (std::ostream& oStream,
                                         const AirportPair& iAirportPair) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    oStream << "+++++" << std::endl;
    oStream << "AirportPair: " << iAirportPair.describeKey() << std::endl;
    oStream << "+++++" << std::endl;

    // Check whether there are date-period objects
    if (BomManager::hasList<DatePeriod> (iAirportPair) == false) {
        return;
    }

    // Browse the date-period objects
    const DatePeriodList_T& lDatePeriodList =
        BomManager::getList<DatePeriod> (iAirportPair);
    for (DatePeriodList_T::const_iterator itDP = lDatePeriodList.begin();
         itDP != lDatePeriodList.end(); ++itDP) {
        const DatePeriod* lDP_ptr = *itDP;
        assert (lDP_ptr != NULL);

        // Display the date-period object
        csvDateDisplay (oStream, *lDP_ptr);
    }
}

// //////////////////////////////////////
void BomDisplay::csvDateDisplay (std::ostream& oStream,

```

```

        const DatePeriod& iDatePeriod) {

    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    oStream << "-----" << std::endl;
    oStream << "DatePeriod: " << iDatePeriod.describeKey() << std::endl;
    oStream << "-----" << std::endl;

    // Check whether there are pos-channel objects
    if (BomManager::hasList<PosChannel> (iDatePeriod) == false) {
        return;
    }

    // Browse the pos-channel objects
    const PosChannelList_T& lPosChannelList =
        BomManager::getList<PosChannel> (iDatePeriod);
    for (PosChannelList_T::const_iterator itPC = lPosChannelList.begin();
        itPC != lPosChannelList.end(); ++itPC) {
        const PosChannel* lPC_ptr = *itPC;
        assert (lPC_ptr != NULL);

        // Display the pos-channel object
        csvPosChannelDisplay (oStream, *lPC_ptr);
    }
}

// ////////////////////////////////////////
void BomDisplay::csvPosChannelDisplay (std::ostream& oStream,
        const PosChannel& iPosChannel) {

    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    oStream << "*****" << std::endl;
    oStream << "PosChannel: " << iPosChannel.describeKey() << std::endl;
    oStream << "*****" << std::endl;

    // Check whether there are time-period objects
    if (BomManager::hasList<TimePeriod> (iPosChannel) == false) {
        return;
    }

    // Browse the time-period objects
    const TimePeriodList_T& lTimePeriodList =
        BomManager::getList<TimePeriod> (iPosChannel);
    for (TimePeriodList_T::const_iterator itTP = lTimePeriodList.begin();
        itTP != lTimePeriodList.end(); ++itTP) {
        const TimePeriod* lTP_ptr = *itTP;
        assert (lTP_ptr != NULL);

        // Display the time-period object
        csvTimeDisplay (oStream, *lTP_ptr);
    }
}

// ////////////////////////////////////////
void BomDisplay::csvTimeDisplay (std::ostream& oStream,
        const TimePeriod& iTimePeriod) {

    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

```

```

oStream << "-----" << std::endl;
oStream << "TimePeriod: " << iTimePeriod.describeKey() << std::endl;
oStream << "-----" << std::endl;

// Only one of the fare/yield feature list exists. Each of the following
// two methods will check for the existence of the list. So, only the
// existing list will be actually displayed.
csvFeatureListDisplay<FareFeatures> (oStream, iTimePeriod);
csvFeatureListDisplay<YieldFeatures> (oStream, iTimePeriod);
}

// ////////////////////////////////////////
template <typename FEATURE_TYPE>
void BomDisplay::csvFeatureListDisplay (std::ostream& oStream,
                                       const TimePeriod& iTimePeriod) {

    // Check whether there are fare/yield-feature objects
    if (BomManager::hasList<FEATURE_TYPE> (iTimePeriod) == false) {
        return;
    }

    // Browse the fare/yield-feature objects
    typedef typename BomHolder<FEATURE_TYPE>::BomList_T FeaturesList_T;
    const FeaturesList_T& lFeaturesList =
        BomManager::getList<FEATURE_TYPE> (iTimePeriod);
    for (typename FeaturesList_T::const_iterator itFF = lFeaturesList.begin();
         itFF != lFeaturesList.end(); ++itFF) {
        const FEATURE_TYPE* lFF_ptr = *itFF;
        assert (lFF_ptr != NULL);

        // Display the fare-features object
        csvFeaturesDisplay (oStream, *lFF_ptr);
    }
}

// ////////////////////////////////////////
template <typename FEATURE_TYPE>
void BomDisplay::csvFeaturesDisplay (std::ostream& oStream,
                                    const FEATURE_TYPE& iFeatures) {

    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    oStream << "-----" << std::endl;
    oStream << "Fare/yield-Features: " << iFeatures.describeKey() << std::endl;
    oStream << "-----" << std::endl;

    // Check whether there are airlineClassList objects
    if (BomManager::hasList<AirlineClassList> (iFeatures) == false) {
        return;
    }

    // Browse the airlineClassList objects
    const AirlineClassListList_T& lAirlineClassListList =
        BomManager::getList<AirlineClassList> (iFeatures);
    for (AirlineClassListList_T::const_iterator itACL =
         lAirlineClassListList.begin();
         itACL != lAirlineClassListList.end(); ++itACL) {
        const AirlineClassList* lACL_ptr = *itACL;
        assert (lACL_ptr != NULL);

        // Display the airlineClassList object
        csvAirlineClassDisplay(oStream, *lACL_ptr);
    }
}

```

```

    }
}

// ////////////////////////////////////////
void BomDisplay::
csvAirlineClassDisplay (std::ostream& oStream,
                       const AirlineClassList& iAirlineClassList) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    oStream << "-----" << std::endl;
    oStream << "AirlineClassList: "
              << iAirlineClassList.describeKey() << std::endl;
    oStream << "-----" << std::endl;
}

}

/*!

```

3 C++ Class Building Sample StdAir BOM Trees

```

*/
// ////////////////////////////////////////
// Import section
// ////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// StdAir
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/BasConst_DefaultObject.hpp>
#include <stdair/basic/BasConst_Request.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomRetriever.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/AirportPair.hpp>
#include <stdair/bom/PosChannel.hpp>
#include <stdair/bom/DatePeriod.hpp>
#include <stdair/bom/TimePeriod.hpp>
#include <stdair/bom/FareFeatures.hpp>
#include <stdair/bom/YieldFeatures.hpp>
#include <stdair/bom/AirlineClassList.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/TravelSolutionStruct.hpp>
#include <stdair/bom/BookingRequestStruct.hpp>
#include <stdair/factory/FacBomManager.hpp>
#include <stdair/factory/FacBom.hpp>
#include <stdair/command/CmdBomManager.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/OnDDDate.hpp>
#include <stdair/bom/SegmentPeriod.hpp>

```

```

#include <stdair/bom/FlightPeriod.hpp>

namespace stdair {

// ////////////////////////////////////////
void CmdBomManager::buildSampleBom (BomRoot& ioBomRoot) {

    // DEBUG
    STDAIR_LOG_DEBUG ("StdAir is building the BOM tree from built-in "
        << "specifications.");

    // ////////////////////////////////////////
    // Build the inventory (flight-dates) and the schedule (flight period) parts.

    buildSampleInventorySchedule (ioBomRoot);

    // Build the pricing (fare rules) and revenue accounting (yields) parts.
    buildSamplePricing (ioBomRoot);

    // ////////////////////////////////////////
    // Build the inventory (flight-dates) and the schedule (flight period) parts.

    buildPartnershipsSampleInventoryAndRM (ioBomRoot);

    // Build the pricing (fare rules) and revenue accounting (yields) parts.
    buildPartnershipsSamplePricing (ioBomRoot);

    // Build a dummy inventory, needed by RMOL.
    buildCompleteDummyInventory (ioBomRoot);
}

// ////////////////////////////////////////
void CmdBomManager::buildSampleInventorySchedule (BomRoot& ioBomRoot) {

    // Inventory
    // Step 0.1: Inventory level
    // Create an Inventory for BA
    const InventoryKey lBAKey ("BA");
    Inventory& lBAInv = FacBom<Inventory>::instance().create (lBAKey);
    FacBomManager::addToListAndMap (ioBomRoot, lBAInv);
    FacBomManager::linkWithParent (ioBomRoot, lBAInv);

    // Create an Inventory for AF
    const InventoryKey lAFKey ("AF");
    Inventory& lAFInv = FacBom<Inventory>::instance().create (lAFKey);
    FacBomManager::addToListAndMap (ioBomRoot, lAFInv);
    FacBomManager::linkWithParent (ioBomRoot, lAFInv);

    // BA
    // Step 0.2: Flight-date level
    // Create a FlightDate (BA9/10-JUN-2011) for BA's Inventory
    FlightNumber_T lFlightNumber = 9;
    Date_T lDate (2011, 6, 10);
    FlightDateKey lFlightDateKey (lFlightNumber, lDate);

    FlightDate& lBA9_20110610_FD =
        FacBom<FlightDate>::instance().create (lFlightDateKey);
    FacBomManager::addToListAndMap (lBAInv, lBA9_20110610_FD);
    FacBomManager::linkWithParent (lBAInv, lBA9_20110610_FD);

    // Display the flight-date
    // STDAIR_LOG_DEBUG ("FlightDate: " << lBA9_20110610_FD.toString());
}
}

```

```

// Step 0.3: Segment-date level
// Create a first SegmentDate (LHR-SYD) for BA's Inventory
// See http://www.britishairways.com/travel/flightinformation/public/fr_fr?&C
//   arrier=BA&FlightNumber=0009&from=LHR&to=SYD&depDate=100611&SellingClass=O
const AirportCode_T lLHR ("LHR");
const AirportCode_T lSYD ("SYD");
const DateOffset_T l1Day (1);
const DateOffset_T l2Days (2);
const Duration_T l2135 (21, 45, 0);
const Duration_T l10610 (6, 10, 0);
const Duration_T l2205 (22, 05, 0);
SegmentDateKey lSegmentDateKey (lLHR, lSYD);

SegmentDate& lLHRSYDSegment =
    FacBom<SegmentDate>::instance().create (lSegmentDateKey);
FacBomManager::addToListAndMap (lBA9_20110610_FD, lLHRSYDSegment);
FacBomManager::linkWithParent (lBA9_20110610_FD, lLHRSYDSegment);

// Fill the SegmentDate content
lLHRSYDSegment.setBoardingDate (lDate);
lLHRSYDSegment.setOffDate (lDate + l2Days);
lLHRSYDSegment.setBoardingTime (l2135);
lLHRSYDSegment.setOffTime (l10610);
lLHRSYDSegment.setElapsedTime (l2135);

// Display the segment-date
// STDAIR_LOG_DEBUG ("SegmentDate: " << lLHRSYDSegment);

// Create a second SegmentDate (LHR-BKK) for BA's Inventory
// See http://www.britishairways.com/travel/flightinformation/public/fr_fr?&C
//   arrier=BA&FlightNumber=0009&from=LHR&to=BKK&depDate=100611&SellingClass=O
const AirportCode_T lBKK ("BKK");
const Duration_T l1540 (15, 40, 0);
const Duration_T l1105 (11, 5, 0);
lSegmentDateKey = SegmentDateKey (lLHR, lBKK);

SegmentDate& lLHRBKKSegment =
    FacBom<SegmentDate>::instance().create (lSegmentDateKey);
FacBomManager::addToListAndMap (lBA9_20110610_FD, lLHRBKKSegment);
FacBomManager::linkWithParent (lBA9_20110610_FD, lLHRBKKSegment);

// Fill the SegmentDate content
lLHRBKKSegment.setBoardingDate (lDate);
lLHRBKKSegment.setOffDate (lDate + l1Day);
lLHRBKKSegment.setBoardingTime (l2135);
lLHRBKKSegment.setOffTime (l1540);
lLHRBKKSegment.setElapsedTime (l1105);

// Display the segment-date
// STDAIR_LOG_DEBUG ("SegmentDate: " << lLHRBKKSegment);

// Create a third SegmentDate (BKK-SYD) for BA's Inventory
// See http://www.britishairways.com/travel/flightinformation/public/fr_fr?&C
//   arrier=BA&FlightNumber=0009&from=BKK&to=SYD&depDate=110611&SellingClass=O
const Duration_T l1705 (17, 5, 0);
const Duration_T l10905 (9, 5, 0);
lSegmentDateKey = SegmentDateKey (lBKK, lSYD);

SegmentDate& lBKKSYSYDSegment =
    FacBom<SegmentDate>::instance().create (lSegmentDateKey);
FacBomManager::addToListAndMap (lBA9_20110610_FD, lBKKSYSYDSegment);

```

```

FacBomManager::linkWithParent (lBA9_20110610_FD, lBKKSYSYDSegment);

// Fill the SegmentDate content
lBKKSYSYDSegment.setBoardingDate (lDate + l1Day);
lBKKSYSYDSegment.setOffDate (lDate + l2Days);
lBKKSYSYDSegment.setBoardingTime (l1705);
lBKKSYSYDSegment.setOffTime (l1540);
lBKKSYSYDSegment.setElapsedTime (l0905);

// Display the segment-date
// STDAIR_LOG_DEBUG ("SegmentDate: " << lBKKSYSYDSegment);

// Step 0.4: Leg-date level
// Create a first LegDate (LHR) for BA's Inventory
LegDateKey lLegDateKey (lLHR);

LegDate& lLHRLeg = FacBom<LegDate>::instance().create (lLegDateKey);
FacBomManager::addToListAndMap (lBA9_20110610_FD, lLHRLeg);
FacBomManager::linkWithParent (lBA9_20110610_FD, lLHRLeg);

// Fill the LegDate content
lLHRLeg.setOffPoint (lBKK);
lLHRLeg.setBoardingDate (lDate);
lLHRLeg.setOffDate (lDate + l1Day);
lLHRLeg.setBoardingTime (l2135);
lLHRLeg.setOffTime (l1540);
lLHRLeg.setElapsedTime (l1105);

// Display the leg-date
// STDAIR_LOG_DEBUG ("LegDate: " << lLHRLeg.toString());

// Create a second LegDate (BKK)
lLegDateKey = LegDateKey (lBKK);

LegDate& lBKKLeg = FacBom<LegDate>::instance().create (lLegDateKey);
FacBomManager::addToListAndMap (lBA9_20110610_FD, lBKKLeg);
FacBomManager::linkWithParent (lBA9_20110610_FD, lBKKLeg);

// Display the leg-date
// STDAIR_LOG_DEBUG ("LegDate: " << lBKKLeg.toString());

// Fill the LegDate content
lBKKLeg.setOffPoint (lSYD);
lBKKLeg.setBoardingDate (lDate + l1Day);
lBKKLeg.setOffDate (lDate + l2Days);
lBKKLeg.setBoardingTime (l1705);
lBKKLeg.setOffTime (l1540);
lBKKLeg.setElapsedTime (l0905);

// Link the segment-dates with the leg-dates
FacBomManager::addToListAndMap (lLHRLeg, lLHRSYDSegment);
FacBomManager::addToListAndMap (lLHRLeg, lLHRBKKSegment);
FacBomManager::addToListAndMap (lBKKLeg, lLHRSYDSegment);
FacBomManager::addToListAndMap (lBKKLeg, lBKKSYSYDSegment);
FacBomManager::addToListAndMap (lLHRSYDSegment, lLHRLeg);
FacBomManager::addToListAndMap (lLHRBKKSegment, lLHRLeg);
FacBomManager::addToListAndMap (lLHRSYDSegment, lBKKLeg);
FacBomManager::addToListAndMap (lBKKSYSYDSegment, lBKKLeg);

// Step 0.5: segment-cabin level
// Create a SegmentCabin (Y) for the Segment LHR-BKK of BA's Inventory

```

```

const CabinCode_T lY ("Y");
SegmentCabinKey lYSegmentCabinKey (lY);

SegmentCabin& lLHRBKKSegmentYCabin =
    FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
FacBomManager::addToListAndMap (lLHRBKKSegment, lLHRBKKSegmentYCabin);
FacBomManager::linkWithParent (lLHRBKKSegment, lLHRBKKSegmentYCabin);

// Display the segment-cabin
// STDAIR_LOG_DEBUG ("SegmentCabin: " << lLHRBKKSegmentYCabin.toString());

// Create a SegmentCabin (Y) of the Segment BKK-SYD;
SegmentCabin& lBKKSYSYDSegmentYCabin =
    FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
FacBomManager::addToListAndMap (lBKKSYSYDSegment, lBKKSYSYDSegmentYCabin);
FacBomManager::linkWithParent (lBKKSYSYDSegment, lBKKSYSYDSegmentYCabin);

// Display the segment-cabin
// STDAIR_LOG_DEBUG ("SegmentCabin: " << lBKKSYSYDSegmentYCabin.toString());

// Create a SegmentCabin (Y) of the Segment LHR-SYD;
SegmentCabin& lLHRSYDSegmentYCabin =
    FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
FacBomManager::addToListAndMap (lLHRSYDSegment, lLHRSYDSegmentYCabin);
FacBomManager::linkWithParent (lLHRSYDSegment, lLHRSYDSegmentYCabin);

// Display the segment-cabin
// STDAIR_LOG_DEBUG ("SegmentCabin: " << lLHRSYDSegmentYCabin.toString());

// Step 0.6: leg-cabin level
// Create a LegCabin (Y) for the Leg LHR-BKK on BA's Inventory
LegCabinKey lYLegCabinKey (lY);

LegCabin& lLHRLegYCabin =
    FacBom<LegCabin>::instance().create (lYLegCabinKey);
FacBomManager::addToListAndMap (lLHRLeg, lLHRLegYCabin);
FacBomManager::linkWithParent (lLHRLeg, lLHRLegYCabin);

// Display the leg-cabin
// STDAIR_LOG_DEBUG ("LegCabin: " << lLHRLegYCabin.toString());

// Create a LegCabin (Y) for the Leg BKK-SYD
LegCabin& lBKKLegYCabin =
    FacBom<LegCabin>::instance().create (lYLegCabinKey);
FacBomManager::addToListAndMap (lBKKLeg, lBKKLegYCabin);
FacBomManager::linkWithParent (lBKKLeg, lBKKLegYCabin);
// Display the leg-cabin
// STDAIR_LOG_DEBUG ("LegCabin: " << lBKKLegYCabin.toString());

FacBomManager::addToListAndMap (lLHRLegYCabin, lLHRSYDSegmentYCabin,
                                lLHRSYDSegmentYCabin.getFullerKey());
FacBomManager::addToListAndMap (lLHRLegYCabin, lLHRBKKSegmentYCabin,
                                lLHRBKKSegmentYCabin.getFullerKey());
FacBomManager::addToListAndMap (lBKKLegYCabin, lLHRSYDSegmentYCabin,
                                lLHRSYDSegmentYCabin.getFullerKey());
FacBomManager::addToListAndMap (lBKKLegYCabin, lBKKSYSYDSegmentYCabin,
                                lBKKSYSYDSegmentYCabin.getFullerKey());

FacBomManager::addToListAndMap (lLHRSYDSegmentYCabin, lLHRLegYCabin,
                                lLHRLegYCabin.getFullerKey());
FacBomManager::addToListAndMap (lLHRBKKSegmentYCabin, lLHRLegYCabin,

```

```

                                lLHRLegYCabin.getFullerKey());
FacBomManager::addToListAndMap (lLHRSYDSegmentYCabin, lBKKLegYCabin,
                                lBKKLegYCabin.getFullerKey());
FacBomManager::addToListAndMap (lBKKSYSYDSegmentYCabin, lBKKLegYCabin,
                                lBKKLegYCabin.getFullerKey());

// Step 0.7: fare family level
// Create a FareFamily (1) for the Segment LHR-BKK, cabin Y on BA's Inv
const FamilyCode_T l1 ("EcoSaver");
FareFamilyKey l1FareFamilyKey (l1);

FareFamily& lLHRBKKSegmentYCabin1Family =
    FacBom<FareFamily>::instance().create (l1FareFamilyKey);
FacBomManager::addToListAndMap (lLHRBKKSegmentYCabin,
                                lLHRBKKSegmentYCabin1Family);
FacBomManager::linkWithParent (lLHRBKKSegmentYCabin,
                                lLHRBKKSegmentYCabin1Family);

// Display the booking class
// STDAIR_LOG_DEBUG ("FareFamily: "
//                  << lLHRBKKSegmentYCabin1Family.toString());

// Create a FareFamily (1) for the Segment BKK-SYD, cabin Y on BA's Inv
FareFamily& lBKKSYSYDSegmentYCabin1Family =
    FacBom<FareFamily>::instance().create (l1FareFamilyKey);
FacBomManager::addToListAndMap (lBKKSYSYDSegmentYCabin,
                                lBKKSYSYDSegmentYCabin1Family);
FacBomManager::linkWithParent (lBKKSYSYDSegmentYCabin,
                                lBKKSYSYDSegmentYCabin1Family);

// Display the booking class
// STDAIR_LOG_DEBUG ("FareFamily: "
//                  << lLHRBKKSegmentYCabin1Family.toString());

// Create a FareFamily (1) for the Segment LHR-SYD, cabin Y on BA's Inv
FareFamily& lLHRSYDSegmentYCabin1Family =
    FacBom<FareFamily>::instance().create (l1FareFamilyKey);
FacBomManager::addToListAndMap (lLHRSYDSegmentYCabin,
                                lLHRSYDSegmentYCabin1Family);
FacBomManager::linkWithParent (lLHRSYDSegmentYCabin,
                                lLHRSYDSegmentYCabin1Family);

// Display the booking class
// STDAIR_LOG_DEBUG ("FareFamily: "
//                  << lLHRBKKSegmentYCabin1Family.toString());

// Step 0.8: booking class level
// Create a BookingClass (Q) for the Segment LHR-BKK, cabin Y,
// fare family 1 on BA's Inv
const ClassCode_T lQ ("Q");
BookingClassKey lQBookingClassKey (lQ);

BookingClass& lLHRBKKSegmentYCabin1FamilyQClass =
    FacBom<BookingClass>::instance().create (lQBookingClassKey);
FacBomManager::addToListAndMap (lLHRBKKSegmentYCabin1Family,
                                lLHRBKKSegmentYCabin1FamilyQClass);
FacBomManager::linkWithParent (lLHRBKKSegmentYCabin1Family,
                                lLHRBKKSegmentYCabin1FamilyQClass);

FacBomManager::addToListAndMap (lLHRBKKSegmentYCabin,

```

```

                                lLHRBKKSegmentYCabin1FamilyQClass);
FacBomManager::addToListAndMap (lLHRBKKSegment,
                                lLHRBKKSegmentYCabin1FamilyQClass);

// Display the booking class
// STDAIR_LOG_DEBUG ("BookingClass: "
//                  << lLHRBKKSegmentYCabin1FamilyQClass.toString());

// Create a BookingClass (Q) for the Segment BKK-SYD, cabin Y,
// fare family 1 on BA's Inv
BookingClass& lBKKSYSYDSegmentYCabin1FamilyQClass =
    FacBom<BookingClass>::instance().create (lQBookingClassKey);
FacBomManager::addToListAndMap (lBKKSYSYDSegmentYCabin1Family,
                                lBKKSYSYDSegmentYCabin1FamilyQClass);
FacBomManager::linkWithParent (lBKKSYSYDSegmentYCabin1Family,
                                lBKKSYSYDSegmentYCabin1FamilyQClass);

FacBomManager::addToListAndMap (lBKKSYSYDSegmentYCabin,
                                lBKKSYSYDSegmentYCabin1FamilyQClass);
FacBomManager::addToListAndMap (lBKKSYSYDSegment,
                                lBKKSYSYDSegmentYCabin1FamilyQClass);

// Display the booking class
// STDAIR_LOG_DEBUG ("BookingClass: "
//                  << lLHRBKKSegmentYCabin1FamilyQClass.toString());

// Create a BookingClass (Q) for the Segment LHR-SYD, cabin Y,
// fare family 1 on BA's Inv
BookingClass& lLHRSYSYDSegmentYCabin1FamilyQClass =
    FacBom<BookingClass>::instance().create (lQBookingClassKey);
FacBomManager::addToListAndMap (lLHRSYSYDSegmentYCabin1Family,
                                lLHRSYSYDSegmentYCabin1FamilyQClass);
FacBomManager::linkWithParent (lLHRSYSYDSegmentYCabin1Family,
                                lLHRSYSYDSegmentYCabin1FamilyQClass);

FacBomManager::addToListAndMap (lLHRSYSYDSegmentYCabin,
                                lLHRSYSYDSegmentYCabin1FamilyQClass);
FacBomManager::addToListAndMap (lLHRSYSYDSegment,
                                lLHRSYSYDSegmentYCabin1FamilyQClass);

// Display the booking class
// STDAIR_LOG_DEBUG ("BookingClass: "
//                  << lLHRBKKSegmentYCabin1FamilyQClass.toString());

// ===== AF =====
// Step 0.2: Flight-date level
// Create a FlightDate (AF084/20-MAR-2011) for AF's Inventory
lFlightNumber = 84;
lDate = Date_T (2011, 3, 20);
lFlightDateKey = FlightDateKey (lFlightNumber, lDate);

FlightDate& lAF084_20110320_FD =
    FacBom<FlightDate>::instance().create (lFlightDateKey);
FacBomManager::addToListAndMap (lAFInv, lAF084_20110320_FD);
FacBomManager::linkWithParent (lAFInv, lAF084_20110320_FD);

// Display the flight-date
// STDAIR_LOG_DEBUG ("FlightDate: " << lAF084_20110320_FD.toString());

// Step 0.3: Segment-date level
// Create a SegmentDate (CDG-SFO) for AF's Inventory

```

```

const AirportCode_T lCDG ("CDG");
const AirportCode_T lSFO ("SFO");
const Duration_T l1040 (10, 40, 0);
const Duration_T l1250 (12, 50, 0);
const Duration_T l1110 (11, 10, 0);
lSegmentDateKey = SegmentDateKey (lCDG, lSFO);

SegmentDate& lCDGSFOSegment =
    FacBom<SegmentDate>::instance().create (lSegmentDateKey);
FacBomManager::addToListAndMap (lAF084_20110320_FD, lCDGSFOSegment);
FacBomManager::linkWithParent (lAF084_20110320_FD, lCDGSFOSegment);

// Display the segment-date
// STDAIR_LOG_DEBUG ("SegmentDate: " << lCDGSFOSegment.toString());

// Fill the SegmentDate content
lCDGSFOSegment.setBoardingDate (lDate);
lCDGSFOSegment.setOffDate (lDate);
lCDGSFOSegment.setBoardingTime (l1040);
lCDGSFOSegment.setOffTime (l1250);
lCDGSFOSegment.setElapsedTime (l1110);

// Step 0.4: Leg-date level
// Create a LegDate (CDG) for AF's Inventory
lLegDateKey = LegDateKey (lCDG);

LegDate& lCDGLeg = FacBom<LegDate>::instance().create (lLegDateKey);
FacBomManager::addToListAndMap (lAF084_20110320_FD, lCDGLeg);
FacBomManager::linkWithParent (lAF084_20110320_FD, lCDGLeg);

// Fill the LegDate content
lCDGLeg.setOffPoint (lSFO);
lCDGLeg.setBoardingDate (lDate);
lCDGLeg.setOffDate (lDate);
lCDGLeg.setBoardingTime (l1040);
lCDGLeg.setOffTime (l1250);
lCDGLeg.setElapsedTime (l1110);

// Display the leg-date
// STDAIR_LOG_DEBUG ("LegDate: " << lCDGLeg.toString());

// Link the segment-dates with the leg-dates
FacBomManager::addToListAndMap (lCDGLeg, lCDGSFOSegment);
FacBomManager::addToListAndMap (lCDGSFOSegment, lCDGLeg);

// Step 0.5: segment-cabin level
// Create a SegmentCabin (Y) for the Segment CDG-SFO of AF's Inventory
SegmentCabin& lCDGSFOSegmentYCabin =
    FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
FacBomManager::addToListAndMap (lCDGSFOSegment, lCDGSFOSegmentYCabin);
FacBomManager::linkWithParent (lCDGSFOSegment, lCDGSFOSegmentYCabin);

// Display the segment-cabin
// STDAIR_LOG_DEBUG ("SegmentCabin: " << lCDGSFOSegmentYCabin.toString());

// Step 0.6: leg-cabin level
// Create a LegCabin (Y) for the Leg CDG-SFO on AF's Inventory
LegCabin& lCDGLegYCabin =
    FacBom<LegCabin>::instance().create (lYLegCabinKey);
FacBomManager::addToListAndMap (lCDGLeg, lCDGLegYCabin);
FacBomManager::linkWithParent (lCDGLeg, lCDGLegYCabin);

```

```

// Display the leg-cabin
// STDAIR_LOG_DEBUG ("LegCabin: " << lLHRLegYCabin.toString());

// Link the segment-dates with the leg-dates
FacBomManager::addToListAndMap (lCDGLegYCabin, lCDGSFOSegmentYCabin,
                                lCDGSFOSegmentYCabin.getFullerKey());
FacBomManager::addToListAndMap (lCDGSFOSegmentYCabin, lCDGLegYCabin,
                                lCDGLegYCabin.getFullerKey());

// Step 0.7: fare family level
// Create a fareFamily (1) for the Segment CDG-SFO, cabin Y on AF's Inv
FareFamily& lCDGSFOSegmentYCabin1Family =
    FacBom<FareFamily>::instance().create (1lFareFamilyKey);
FacBomManager::addToListAndMap (lCDGSFOSegmentYCabin,
                                lCDGSFOSegmentYCabin1Family);
FacBomManager::linkWithParent (lCDGSFOSegmentYCabin,
                                lCDGSFOSegmentYCabin1Family);

// Display the fare family
// STDAIR_LOG_DEBUG ("fareFamily: "
//
//                    << lCDGSFOSegmentYCabin1Family.toString());

// Step 0.8: booking class level Create a BookingClass (Q) for the
// Segment CDG-SFO, cabin Y, fare family 1 on AF's Inv
BookingClass& lCDGSFOSegmentYCabin1FamilyQClass =
    FacBom<BookingClass>::instance().create (1QBookingClassKey);
FacBomManager::addToListAndMap (lCDGSFOSegmentYCabin1Family,
                                lCDGSFOSegmentYCabin1FamilyQClass);
FacBomManager::linkWithParent (lCDGSFOSegmentYCabin1Family,
                                lCDGSFOSegmentYCabin1FamilyQClass);

FacBomManager::addToListAndMap (lCDGSFOSegmentYCabin,
                                lCDGSFOSegmentYCabin1FamilyQClass);
FacBomManager::addToListAndMap (lCDGSFOSegment,
                                lCDGSFOSegmentYCabin1FamilyQClass);

// Display the booking class
// STDAIR_LOG_DEBUG ("BookingClass: "
//                    << lCDGSFOSegmentYCabin1FamilyQClass.toString());

/*=====
=====
=====
=====*/
// Schedule:
// BA:
// Step 1: flight period level
// Create a flight period for BA9:
const DoWStruct lDoWSrtuct ("1111111");
const Date_T lBA9DateRangeStart (2010, boost::gregorian::Jun, 6);
const Date_T lBA9DateRangeEnd (2010, boost::gregorian::Jun, 7);
const DatePeriod_T lBA9DatePeriod (lBA9DateRangeStart, lBA9DateRangeEnd);
const PeriodStruct lBA9PeriodStruct (lBA9DatePeriod, lDoWSrtuct);

lFlightNumber = FlightNumber_T (9);

FlightPeriodKey lBA9FlightPeriodKey (lFlightNumber, lBA9PeriodStruct);

```

```

FlightPeriod& lBA9FlightPeriod =
    FacBom<FlightPeriod>::instance().create (lBA9FlightPeriodKey);
FacBomManager::addToListAndMap (lBAInv, lBA9FlightPeriod);
FacBomManager::linkWithParent (lBAInv, lBA9FlightPeriod);

// Step 2: segment period level
// Create a segment period for SIN-BKK:

SegmentPeriodKey lLHRSYDSegmentPeriodKey (lLHR, lSYD);

SegmentPeriod& lLHRSYDSegmentPeriod =
    FacBom<SegmentPeriod>::instance().create (lLHRSYDSegmentPeriodKey);
FacBomManager::addToListAndMap (lBA9FlightPeriod, lLHRSYDSegmentPeriod);
FacBomManager::linkWithParent (lBA9FlightPeriod, lLHRSYDSegmentPeriod);

lLHRSYDSegmentPeriod.setBoardingTime (l2135);
lLHRSYDSegmentPeriod.setOffTime (l1540);
lLHRSYDSegmentPeriod.setElapsedTime (l1105);
ClassList_String_T lYM ("YM");
lLHRSYDSegmentPeriod.addCabinBookingClassList (lY,lYM);

// AF:
// Step 1: flight period level
// Create a flight period for AF84:
const Date_T lAF84DateRangeStart (2011, boost::gregorian::Mar, 20);
const Date_T lAF84DateRangeEnd (2011, boost::gregorian::Mar, 21);
const DatePeriod_T lAF84DatePeriod (lAF84DateRangeStart, lAF84DateRangeEnd);
const PeriodStruct lAF84PeriodStruct (lAF84DatePeriod, lDoWSrtuct);

lFlightNumber = FlightNumber_T (84);

FlightPeriodKey lAF84FlightPeriodKey (lFlightNumber, lAF84PeriodStruct);

FlightPeriod& lAF84FlightPeriod =
    FacBom<FlightPeriod>::instance().create (lAF84FlightPeriodKey);
FacBomManager::addToListAndMap (lAFInv, lAF84FlightPeriod);
FacBomManager::linkWithParent (lAFInv, lAF84FlightPeriod);

// Step 2: segment period level
// Create a segment period for SIN-BKK:

SegmentPeriodKey lCDGSFOSegmentPeriodKey (lCDG, lSFO);

SegmentPeriod& lCDGSFOSegmentPeriod =
    FacBom<SegmentPeriod>::instance().create (lCDGSFOSegmentPeriodKey);
FacBomManager::addToListAndMap (lAF84FlightPeriod, lCDGSFOSegmentPeriod);
FacBomManager::linkWithParent (lAF84FlightPeriod, lCDGSFOSegmentPeriod);

lCDGSFOSegmentPeriod.setBoardingTime (l1040);
lCDGSFOSegmentPeriod.setOffTime (l1250);
lCDGSFOSegmentPeriod.setElapsedTime (l1110);
lCDGSFOSegmentPeriod.addCabinBookingClassList (lY,lYM);

/*=====
=====
=====
=====*/
// O&D
// Create an O&D Date (BA;9,2010-Jun-06;LHR,SYD) for BA's Inventory

```

```

OnDString_T lBALHRSYDOnDStr = "BA;9,2010-Jun-06;LHR,SYD";
OnDStringList_T lBAOnDStrList;
lBAOnDStrList.push_back (lBALHRSYDOnDStr);

OnDDateKey lBAOnDDateKey (lBAOnDStrList);
OnDDate& lBA_LHRSYD_OnDDate =
    FacBom<OnDDate>::instance().create (lBAOnDDateKey);
// Link to the inventory
FacBomManager::addToListAndMap (lBAInv, lBA_LHRSYD_OnDDate);
FacBomManager::linkWithParent (lBAInv, lBA_LHRSYD_OnDDate);

// Add the segment
FacBomManager::addToListAndMap (lBA_LHRSYD_OnDDate, lLHRSYDSegment);

// Add total forecast info for cabin Y.
const MeanStdDevPair_T lMean60StdDev6 (60.0, 6.0);
const WTP_T lWTP750 = 750.0;
const WTPDemandPair_T lWTP750Mean60StdDev6 (lWTP750, lMean60StdDev6);
lBA_LHRSYD_OnDDate.setTotalForecast (lY, lWTP750Mean60StdDev6);

// Create an O&D Date (AF;84,2011-Mar-21;CDG,SFO) for AF's Inventory
OnDString_T lAFLHRSYDOnDStr = "AF;9,2011-Mar-20;CDG,SFO";
OnDStringList_T lAFOnDStrList;
lAFOnDStrList.push_back (lAFLHRSYDOnDStr);

OnDDateKey lAFOnDDateKey (lAFOnDStrList);
OnDDate& lAF_LHRSYD_OnDDate =
    FacBom<OnDDate>::instance().create (lAFOnDDateKey);
// Link to the inventory
FacBomManager::addToListAndMap (lAFInv, lAF_LHRSYD_OnDDate);
FacBomManager::linkWithParent (lAFInv, lAF_LHRSYD_OnDDate);

// Add the segment
FacBomManager::addToListAndMap (lAF_LHRSYD_OnDDate, lLHRSYDSegment);

// Add total forecast info for cabin Y.
lAF_LHRSYD_OnDDate.setTotalForecast (lY, lWTP750Mean60StdDev6);
}
// //////////////////////////////////////
void CmdBomManager::buildCompleteDummyInventory (BomRoot& ioBomRoot) {

    // Build a dummy inventory, containing a dummy flight-date with a
    // single segment-cabin and a single leg-cabin.
    const CabinCapacity_T lCapacity = DEFAULT_CABIN_CAPACITY;
    buildDummyInventory (ioBomRoot, lCapacity);

    // Retrieve the (sample) segment-cabin.
    SegmentCabin& lDummySegmentCabin =
        BomRetriever::retrieveDummySegmentCabin (ioBomRoot);

    // Retrieve the (sample) leg-cabin.
    LegCabin& lDummyLegCabin =
        BomRetriever::retrieveDummyLegCabin (ioBomRoot);

    // Add some booking classes to the dummy segment-cabin and some
    // virtual ones to the dummy leg-cabin.
    // First booking class yield and demand information.
    Yield_T lYield = 100;
    MeanValue_T lMean = 20;
    StdDevValue_T lStdDev = 9;
    BookingClassKey lBCKey (DEFAULT_CLASS_CODE);

```

```

BookingClass& lDummyBookingClass =
    FacBom<BookingClass>::instance().create (lBCKey);
lDummyBookingClass.setYield (lYield);
lDummyBookingClass.setMean (lMean);
lDummyBookingClass.setStdDev (lStdDev);
// Add a booking class to the segment-cabin.
FacBomManager::addToList (lDummySegmentCabin, lDummyBookingClass);

VirtualClassStruct lDummyVirtualClass (lDummyBookingClass);
lDummyVirtualClass.setYield (lYield);
lDummyVirtualClass.setMean (lMean);
lDummyVirtualClass.setStdDev (lStdDev);
// Add the corresponding virtual class to the leg-cabin.
lDummyLegCabin.addVirtualClass (lDummyVirtualClass);

// Second booking class yield and demand information.
lYield = 70;
lMean = 45;
lStdDev= 12;
lDummyBookingClass.setYield (lYield);
lDummyBookingClass.setMean (lMean);
lDummyBookingClass.setStdDev (lStdDev);
// Add a booking class to the segment-cabin.
FacBomManager::addToList (lDummySegmentCabin, lDummyBookingClass);

lDummyVirtualClass.setYield (lYield);
lDummyVirtualClass.setMean (lMean);
lDummyVirtualClass.setStdDev (lStdDev);
// Add the corresponding virtual class to the leg-cabin.
lDummyLegCabin.addVirtualClass (lDummyVirtualClass);

// Third booking class yield and demand information.
lYield = 42;
lMean = 80;
lStdDev= 16;
lDummyBookingClass.setYield (lYield);
lDummyBookingClass.setMean (lMean);
lDummyBookingClass.setStdDev (lStdDev);
// Add a booking class to the segment-cabin.
FacBomManager::addToList (lDummySegmentCabin, lDummyBookingClass);

lDummyVirtualClass.setYield (lYield);
lDummyVirtualClass.setMean (lMean);
lDummyVirtualClass.setStdDev (lStdDev);
// Add the corresponding virtual class to the leg-cabin.
lDummyLegCabin.addVirtualClass (lDummyVirtualClass);
}

// ////////////////////////////////////////
void CmdBomManager::buildDummyInventory (BomRoot& ioBomRoot,
                                         const CabinCapacity_T& iCapacity) {
    // Inventory
    const InventoryKey lInventoryKey (DEFAULT_AIRLINE_CODE);
    Inventory& lInv = FacBom<Inventory>::instance().create (lInventoryKey);
    FacBomManager::addToListAndMap (ioBomRoot, lInv);
    FacBomManager::linkWithParent (ioBomRoot, lInv);

    // Flight-date
    FlightDateKey lFlightDateKey (DEFAULT_FLIGHT_NUMBER, DEFAULT_DEPARTURE_DATE);
    FlightDate& lFlightDate =

```

```

    FacBom<FlightDate>::instance().create (lFlightDateKey);
    FacBomManager::addToListAndMap (lInv, lFlightDate);
    FacBomManager::linkWithParent (lInv, lFlightDate);

    // Leg-date
    LegDateKey lLegDateKey (DEFAULT_ORIGIN);
    LegDate& lLeg = FacBom<LegDate>::instance().create (lLegDateKey);
    FacBomManager::addToListAndMap (lFlightDate, lLeg);
    FacBomManager::linkWithParent (lFlightDate, lLeg);

    // Fill the LegDate content
    lLeg.setOffPoint (DEFAULT_DESTINATION);
    lLeg.setBoardingDate (DEFAULT_DEPARTURE_DATE);
    lLeg.setOffDate (DEFAULT_DEPARTURE_DATE);
    lLeg.setBoardingTime (Duration_T (14, 0, 0));
    lLeg.setOffTime (Duration_T (16, 0, 0));
    lLeg.setElapsedTime (Duration_T (8, 0, 0));

    // Leg-cabin
    LegCabinKey lLegCabinKey (DEFAULT_CABIN_CODE);
    LegCabin& lLegCabin = FacBom<LegCabin>::instance().create (lLegCabinKey);
    FacBomManager::addToListAndMap (lLeg, lLegCabin);
    FacBomManager::linkWithParent (lLeg, lLegCabin);

    lLegCabin.setCapacities (iCapacity);
    lLegCabin.setAvailabilityPool (iCapacity);

    // Segment-date
    SegmentDateKey lSegmentDateKey (DEFAULT_ORIGIN, DEFAULT_DESTINATION);
    SegmentDate& lSegment =
        FacBom<SegmentDate>::instance().create (lSegmentDateKey);
    FacBomManager::addToListAndMap (lFlightDate, lSegment);
    FacBomManager::linkWithParent (lFlightDate, lSegment);

    // Links between the segment-date and the leg-date
    FacBomManager::addToListAndMap (lLeg, lSegment);
    FacBomManager::addToListAndMap (lSegment, lLeg);

    // Fill the SegmentDate content
    lSegment.setBoardingDate (DEFAULT_DEPARTURE_DATE);
    lSegment.setOffDate (DEFAULT_DEPARTURE_DATE);
    lSegment.setBoardingTime (Duration_T (14, 0, 0));
    lSegment.setOffTime (Duration_T (16, 0, 0));
    lSegment.setElapsedTime (Duration_T (8, 0, 0));

    // Segment-cabin
    SegmentCabinKey lSegmentCabinKey (DEFAULT_CABIN_CODE);
    SegmentCabin& lSegmentCabin =
        FacBom<SegmentCabin>::instance().create (lSegmentCabinKey);
    FacBomManager::addToListAndMap (lSegment, lSegmentCabin);
    FacBomManager::linkWithParent (lSegment, lSegmentCabin);

    // Links between the segment-cabin and the leg-cabin
    FacBomManager::addToListAndMap (lLegCabin, lSegmentCabin,
                                    lSegmentCabin.getFullerKey());
    FacBomManager::addToListAndMap (lSegmentCabin, lLegCabin,
                                    lLegCabin.getFullerKey());

    // Create a FareFamily (1) for the Segment LHR-BKK, cabin Y on BA's Inv
    const FamilyCode_T l1 ("EcoSaver");
    FareFamilyKey l1FareFamilyKey (l1);

```

```

FareFamily& lSegmentYCabinlFamily =
    FacBom<FareFamily>::instance().create (lFareFamilyKey);
FacBomManager::addToListAndMap (lSegmentCabin, lSegmentYCabinlFamily);
FacBomManager::linkWithParent (lSegmentCabin, lSegmentYCabinlFamily);

// Create a booking-class
const ClassCode_T lQ ("Q");
BookingClassKey lQBookingClassKey (lQ);

BookingClass& lSegmentYCabinlFamilyQClass =
    FacBom<BookingClass>::instance().create (lQBookingClassKey);
FacBomManager::addToListAndMap (lSegmentYCabinlFamily,
                                lSegmentYCabinlFamilyQClass);
FacBomManager::linkWithParent (lSegmentYCabinlFamily,
                                lSegmentYCabinlFamilyQClass);

FacBomManager::addToListAndMap (lSegmentCabin, lSegmentYCabinlFamilyQClass);
FacBomManager::addToListAndMap (lSegment, lSegmentYCabinlFamilyQClass);
}

// ////////////////////////////////////////
void CmdBomManager::buildSamplePricing (BomRoot& ioBomRoot) {

    // Set the airport-pair primary key.
    const AirportPairKey lAirportPairKey (AIRPORT_LHR, AIRPORT_SYD);

    // Create the AirportPairKey object and link it to the BOM tree root.
    AirportPair& lAirportPair =
        FacBom<AirportPair>::instance().create (lAirportPairKey);
    FacBomManager::addToListAndMap (ioBomRoot, lAirportPair);
    FacBomManager::linkWithParent (ioBomRoot, lAirportPair);

    // Set the fare date-period primary key.
    const Date_T lDateRangeStart (2011, boost::gregorian::Jan, 15);
    const Date_T lDateRangeEnd (2011, boost::gregorian::Dec, 31);
    const DatePeriod_T lDateRange (lDateRangeStart, lDateRangeEnd);
    const DatePeriodKey lDatePeriodKey (lDateRange);

    // Create the DatePeriodKey object and link it to the PosChannel object.
    DatePeriod& lDatePeriod =
        FacBom<DatePeriod>::instance().create (lDatePeriodKey);
    FacBomManager::addToListAndMap (lAirportPair, lDatePeriod);
    FacBomManager::linkWithParent (lAirportPair, lDatePeriod);

    // Set the point-of-sale-channel primary key.
    const PosChannelKey lPosChannelKey (POS_LHR, CHANNEL_DN);

    // Create the PositionKey object and link it to the AirportPair object.
    PosChannel& lPosChannel =
        FacBom<PosChannel>::instance().create (lPosChannelKey);
    FacBomManager::addToListAndMap (lDatePeriod, lPosChannel);
    FacBomManager::linkWithParent (lDatePeriod, lPosChannel);

    // Set the fare time-period primary key.
    const Time_T lTimeRangeStart (0, 0, 0);
    const Time_T lTimeRangeEnd (23, 0, 0);
    const TimePeriodKey lTimePeriodKey (lTimeRangeStart, lTimeRangeEnd);

    // Create the TimePeriodKey and link it to the DatePeriod object.
    TimePeriod& lTimePeriod =
        FacBom<TimePeriod>::instance().create (lTimePeriodKey);

```

```

FacBomManager::addToListAndMap (lPosChannel, lTimePeriod);
FacBomManager::linkWithParent (lPosChannel, lTimePeriod);

// Pricing -- Generate the FareRule
const FareFeaturesKey lFareFeaturesKey (TRIP_TYPE_ROUND_TRIP,
                                         NO_ADVANCE_PURCHASE,
                                         SATURDAY_STAY,
                                         CHANGE_FEES,
                                         NON_REFUNDABLE,
                                         NO_STAY_DURATION);

// Create the FareFeaturesKey and link it to the TimePeriod object.
FareFeatures& lFareFeatures =
    FacBom<FareFeatures>::instance().create (lFareFeaturesKey);
FacBomManager::addToListAndMap (lTimePeriod, lFareFeatures);
FacBomManager::linkWithParent (lTimePeriod, lFareFeatures);

// Revenue Accounting -- Generate the YieldRule
const YieldFeaturesKey lYieldFeaturesKey (TRIP_TYPE_ROUND_TRIP,
                                          CABIN_Y);

// Create the YieldFeaturesKey and link it to the TimePeriod object.
YieldFeatures& lYieldFeatures =
    FacBom<YieldFeatures>::instance().create (lYieldFeaturesKey);
FacBomManager::addToListAndMap (lTimePeriod, lYieldFeatures);
FacBomManager::linkWithParent (lTimePeriod, lYieldFeatures);

// Generate Segment Features and link them to their respective
// fare and yield rules.
AirlineCodeList_T lAirlineCodeList;
lAirlineCodeList.push_back (AIRLINE_CODE_BA);
ClassList_StringList_T lClassCodeList;
lClassCodeList.push_back (CLASS_CODE_Y);
const AirlineClassListKey lAirlineClassListKey (lAirlineCodeList,
                                                lClassCodeList);

// Create the AirlineClassList
AirlineClassList& lAirlineClassList =
    FacBom<AirlineClassList>::instance().create (lAirlineClassListKey);
// Link the AirlineClassList to the FareFeatures object
lAirlineClassList.setFare (900);
FacBomManager::addToListAndMap (lFareFeatures, lAirlineClassList);
FacBomManager::linkWithParent (lFareFeatures, lAirlineClassList);

// Link the AirlineClassList to the YieldFeatures object
lAirlineClassList.setYield (900);
FacBomManager::addToListAndMap (lYieldFeatures, lAirlineClassList);
// \todo (gsabatier): the following calls overrides the parent for
// lAirlineClassList. Check that it is what is actually wanted.
FacBomManager::linkWithParent (lYieldFeatures, lAirlineClassList);
}

// //////////////////////////////////////
void CmdBomManager::
buildSampleTravelSolutionForPricing (TravelSolutionList_T& ioTravelSolutionList
) {

    // Clean the list
    ioTravelSolutionList.clear();

    //
    const std::string lBA9_SegmentDateKey ("BA, 9, 2011-06-10, LHR, SYD, 21:45");

```

```

// Add the segment date key to the travel solution
TravelSolutionStruct lTS;
lTS.addSegment (lBA9_SegmentDateKey);

// Add the travel solution to the list
ioTravelSolutionList.push_back (lTS);
}

// ////////////////////////////////////////
void CmdBomManager::
buildSampleTravelSolutions (TravelSolutionList_T& ioTravelSolutionList) {

    // Clean the list
    ioTravelSolutionList.clear();

    //
    const std::string lBA9_SegmentDateKey ("BA, 9, 2011-06-10, LHR, SYD, 21:45");

    // Add the segment date key to the travel solution
    TravelSolutionStruct lTS;
    lTS.addSegment (lBA9_SegmentDateKey);

    // Fare option
    const ClassCode_T lClassPath (CLASS_CODE_Q);
    const Fare_T lFare (900);
    const ChangeFees_T lChangeFee (CHANGE_FEES);
    const NonRefundable_T isNonRefundable (NON_REFUNDABLE);
    const SaturdayStay_T lSaturdayStay (SATURDAY_STAY);
    const FareOptionStruct lFareOption (lClassPath, lFare, lChangeFee,
                                        isNonRefundable, lSaturdayStay);

    // Add (a copy of) the fare option
    lTS.addFareOption (lFareOption);

    // Map of class availabilities: set the availability for the Q
    // booking class (the one corresponding to the fare option) to 8.
    ClassAvailabilityMap_T lClassAvailabilityMap;
    const Availability_T lAvl (8);
    const bool hasInsertBeenSuccessful = lClassAvailabilityMap.
        insert (ClassAvailabilityMap_T::value_type (lClassPath, lAvl)).second;
    assert (hasInsertBeenSuccessful == true);
    // Add the map to the dedicated list held by the travel solution
    lTS.addClassAvailabilityMap (lClassAvailabilityMap);

    // Add the travel solution to the list
    ioTravelSolutionList.push_back (lTS);
}

// ////////////////////////////////////////
BookingRequestStruct CmdBomManager::buildSampleBookingRequest() {
    // Origin
    const AirportCode_T lOrigin (AIRPORT_LHR);

    // Destination
    const AirportCode_T lDestination (AIRPORT_SYD);

    // Point of Sale (POS)
    const CityCode_T lPOS (POS_LHR);
}

```

```

// Preferred departure date (10-JUN-2011)
const Date_T lPreferredDepartureDate (2011, boost::gregorian::Jun, 10);

// Preferred departure time (08:00)
const Duration_T lPreferredDepartureTime (8, 0, 0);

// Date of the request (15-MAY-2011)
const Date_T lRequestDate (2011, boost::gregorian::May, 15);

// Time of the request (10:00)
const Duration_T lRequestTime (10, 0, 0);

// Date-time of the request (made of the date and time above)
const DateTime_T lRequestDateTime (lRequestDate, lRequestTime);

// Preferred cabin (also named class of service sometimes)
const CabinCode_T lPreferredCabin (CABIN_ECO);

// Number of persons in the party
const PartySize_T lPartySize (3);

// Channel (direct/indirect, on-line/off-line)
const ChannelLabel_T lChannel (CHANNEL_DN);

// Type of the trip (one-way, inbound/outbound of a return trip)
const TripType_T lTripType (TRIP_TYPE_INBOUND);

// Duration of the stay (expressed as a number of days)
const DayDuration_T lStayDuration (DEFAULT_STAY_DURATION);

// Frequent flyer tier (member, silver, gold, platinum, senator, etc)
const FrequentFlyer_T lFrequentFlyerType (FREQUENT_FLYER_MEMBER);

// Maximum willing-to-pay (WTP, expressed in monetary unit, e.g., EUR)
const WTP_T lWTP (DEFAULT_WTP);

// Value of time, for the customer (expressed in monetary unit per
// unit of time, e.g., EUR/hour)
const PriceValue_T lValueOfTime (DEFAULT_VALUE_OF_TIME);

// Creation of the booking request structure
BookingRequestStruct oBookingRequest (lOrigin, lDestination, lPOS,
                                       lPreferredDepartureDate,
                                       lRequestDateTime,
                                       lPreferredCabin,
                                       lPartySize, lChannel,
                                       lTripType, lStayDuration,
                                       lFrequentFlyerType,
                                       lPreferredDepartureTime,
                                       lWTP, lValueOfTime);

return oBookingRequest;
}

// ////////////////////////////////////////
BookingRequestStruct CmdBomManager::buildSampleBookingRequestForCRS() {
    // Origin
    const AirportCode_T lOrigin (AIRPORT_SIN);

    // Destination
    const AirportCode_T lDestination (AIRPORT_BKK);

```

```

// Point of Sale (POS)
const CityCode_T lPOS (POS_SIN);

// Preferred departure date (30-JAN-2010)
const Date_T lPreferredDepartureDate (2010, boost::gregorian::Jan, 30);

// Preferred departure time (10:00)
const Duration_T lPreferredDepartureTime (10, 0, 0);

// Date of the request (22-JAN-2010)
const Date_T lRequestDate (2010, boost::gregorian::Jan, 22);

// Time of the request (10:00)
const Duration_T lRequestTime (10, 0, 0);

// Date-time of the request (made of the date and time above)
const DateTime_T lRequestDateTime (lRequestDate, lRequestTime);

// Preferred cabin (also named class of service sometimes)
const CabinCode_T lPreferredCabin (CABIN_ECO);

// Number of persons in the party
const PartySize_T lPartySize (3);

// Channel (direct/indirect, on-line/off-line)
const ChannelLabel_T lChannel (CHANNEL_IN);

// Type of the trip (one-way, inbound/outbound of a return trip)
const TripType_T lTripType (TRIP_TYPE_INBOUND);

// Duration of the stay (expressed as a number of days)
const DayDuration_T lStayDuration (DEFAULT_STAY_DURATION);

// Frequent flyer tier (member, silver, gold, platinum, senator, etc)
const FrequentFlyer_T lFrequentFlyerType (FREQUENT_FLYER_MEMBER);

// Maximum willing-to-pay (WTP, expressed in monetary unit, e.g., EUR)
const WTP_T lWTP (DEFAULT_WTP);

// Value of time, for the customer (expressed in monetary unit per
// unit of time, e.g., EUR/hour)
const PriceValue_T lValueOfTime (DEFAULT_VALUE_OF_TIME);

// Creation of the booking request structure
BookingRequestStruct oBookingRequest (lOrigin,
                                       lDestination,
                                       lPOS,
                                       lPreferredDepartureDate,
                                       lRequestDateTime,
                                       lPreferredCabin,
                                       lPartySize, lChannel,
                                       lTripType, lStayDuration,
                                       lFrequentFlyerType,
                                       lPreferredDepartureTime,
                                       lWTP, lValueOfTime);

return oBookingRequest;
}

// //////////////////////////////////////
void CmdBomManager::buildPartnershipsSampleInventoryAndRM (BomRoot& ioBomRoot)
{

```

```

// Step 0.1: Inventory level
// Create an Inventory for SQ
const InventoryKey lSQKey ("SQ");
Inventory& lSQInv = FacBom<Inventory>::instance().create (lSQKey);
FacBomManager::addToListAndMap (ioBomRoot, lSQInv);
FacBomManager::linkWithParent (ioBomRoot, lSQInv);

// Create an Inventory for CX
const InventoryKey lCXKey ("CX");
Inventory& lCXInv = FacBom<Inventory>::instance().create (lCXKey);
FacBomManager::addToListAndMap (ioBomRoot, lCXInv);
FacBomManager::linkWithParent (ioBomRoot, lCXInv);

// ===== SQ =====
// Step 0.2: Flight-date level
// Create a FlightDate (SQ11/08-FEB-2010) for SQ's Inventory
FlightNumber_T lFlightNumber = 11;
Date_T lDate (2010, 2, 8);
FlightDateKey lFlightDateKey (lFlightNumber, lDate);

FlightDate& lSQ11_20100208_FD =
    FacBom<FlightDate>::instance().create (lFlightDateKey);
FacBomManager::addToListAndMap (lSQInv, lSQ11_20100208_FD);
FacBomManager::linkWithParent (lSQInv, lSQ11_20100208_FD);

// Create a (mkt) FlightDate (SQ1200/08-FEB-2010) for SQ's Inventory
FlightNumber_T lMktFlightNumber = 1200;
//lDate = Date_T (2010, 2, 8);
FlightDateKey lMktFlightDateKey (lMktFlightNumber, lDate);

FlightDate& lSQ1200_20100208_FD =
    FacBom<FlightDate>::instance().create (lMktFlightDateKey);
FacBomManager::addToListAndMap (lSQInv, lSQ1200_20100208_FD);
FacBomManager::linkWithParent (lSQInv, lSQ1200_20100208_FD);

// Display the flight-date
// STDAIR_LOG_DEBUG ("FlightDate: " << lBA9_20110610_FD.toString());

// Step 0.3: Segment-date level
// Create a first SegmentDate (SIN-BKK) for SQ's Inventory
const AirportCode_T lSIN ("SIN");
const AirportCode_T lBKK ("BKK");
const DateOffset_T l1Day (1);
const DateOffset_T l2Days (2);
const Duration_T l0820 (8, 20, 0);
const Duration_T l1100 (11, 0, 0);
const Duration_T l10340 (3, 40, 0);
SegmentDateKey lSegmentDateKey (lSIN, lBKK);

SegmentDate& lSINBKKSegment =
    FacBom<SegmentDate>::instance().create (lSegmentDateKey);
FacBomManager::addToListAndMap (lSQ11_20100208_FD, lSINBKKSegment);
FacBomManager::linkWithParent (lSQ11_20100208_FD, lSINBKKSegment);

// Fill the SegmentDate content
lSINBKKSegment.setBoardingDate (lDate);
lSINBKKSegment.setOffDate (lDate);
lSINBKKSegment.setBoardingTime (l0820);
lSINBKKSegment.setOffTime (l1100);
lSINBKKSegment.setElapsedTime (l10340);

```

```

// Create a second (mkt) SegmentDate (BKK-HKG) for SQ's Inventory
const AirportCode_T lHKG ("HKG");
const Duration_T l1200 (12, 0, 0);
const Duration_T l1540 (15, 40, 0);
const Duration_T l0240 (2, 40, 0);
SegmentDateKey lMktSegmentDateKey (lBKK, lHKG);

SegmentDate& lMktBKKHKGSegment =
    FacBom<SegmentDate>::instance().create (lMktSegmentDateKey);
FacBomManager::addToListAndMap (lSQ1200_20100208_FD, lMktBKKHKGSegment);
FacBomManager::linkWithParent (lSQ1200_20100208_FD, lMktBKKHKGSegment);

// Fill the (mkt) SegmentDate content
lMktBKKHKGSegment.setBoardingDate (lDate);
lMktBKKHKGSegment.setOffDate (lDate);
lMktBKKHKGSegment.setBoardingTime (l1200);
lMktBKKHKGSegment.setOffTime (l1540);
lMktBKKHKGSegment.setElapsedTime (l0240);

// Step 0.4: Leg-date level
// Create a first LegDate (SIN) for SQ's Inventory
LegDateKey lLegDateKey (lSIN);

LegDate& lSINLeg = FacBom<LegDate>::instance().create (lLegDateKey);
FacBomManager::addToListAndMap (lSQ11_20100208_FD, lSINLeg);
FacBomManager::linkWithParent (lSQ11_20100208_FD, lSINLeg);

// Fill the LegDate content
lSINLeg.setOffPoint (lBKK);
lSINLeg.setBoardingDate (lDate);
lSINLeg.setOffDate (lDate);
lSINLeg.setBoardingTime (l0820);
lSINLeg.setOffTime (l1100);
lSINLeg.setElapsedTime (l0340);

// Link the segment-dates with the leg-dates
FacBomManager::addToListAndMap (lSINLeg, lSINBKKSegment);
FacBomManager::addToListAndMap (lSINBKKSegment, lSINLeg);

// Step 0.5: segment-cabin level
// Create a SegmentCabin (Y) for the Segment SIN-BKK of SQ's Inventory
const CabinCode_T lY ("Y");
SegmentCabinKey lYSegmentCabinKey (lY);

SegmentCabin& lSINBKKSegmentYCabin =
    FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
FacBomManager::addToListAndMap (lSINBKKSegment, lSINBKKSegmentYCabin);
FacBomManager::linkWithParent (lSINBKKSegment, lSINBKKSegmentYCabin);

// Create a SegmentCabin (Y) for the (mkt) Segment BKK-HKG of SQ's Inventory
SegmentCabin& lMktBKKHKGSegmentYCabin =
    FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
FacBomManager::addToListAndMap (lMktBKKHKGSegment, lMktBKKHKGSegmentYCabin);
FacBomManager::linkWithParent (lMktBKKHKGSegment, lMktBKKHKGSegmentYCabin);

// Step 0.6: leg-cabin level
// Create a LegCabin (Y) for the Leg SIN-BKK on SQ's Inventory
LegCabinKey lYLegCabinKey (lY);

```

```

LegCabin& lSINLegYCabin =
    FacBom<LegCabin>::instance().create (lYLegCabinKey);
FacBomManager::addToListAndMap (lSINLeg, lSINLegYCabin);
FacBomManager::linkWithParent (lSINLeg, lSINLegYCabin);

CabinCapacity_T lCapacity (100);
lSINLegYCabin.setCapacities (lCapacity);
lSINLegYCabin.setAvailabilityPool (lCapacity);

    FacBomManager::addToListAndMap (lSINLegYCabin, lSINBKKSegmentYCabin,
                                    lSINBKKSegmentYCabin.getFullerKey());

    FacBomManager::addToListAndMap (lSINBKKSegmentYCabin, lSINLegYCabin,
                                    lSINLegYCabin.getFullerKey());

// Step 0.7: fare family level
// Create a FareFamily (1) for the Segment SIN-BKK, cabin Y on SQ's Inv
const FamilyCode_T l1 ("EcoSaver");
FareFamilyKey l1FareFamilyKey (l1);

FareFamily& lSINBKKSegmentYCabin1Family =
    FacBom<FareFamily>::instance().create (l1FareFamilyKey);
FacBomManager::addToListAndMap (lSINBKKSegmentYCabin,
                                lSINBKKSegmentYCabin1Family);
FacBomManager::linkWithParent (lSINBKKSegmentYCabin,
                                lSINBKKSegmentYCabin1Family);

// Create a FareFamily (1) for the (mkt) Segment BKK-HKG, cabin Y on SQ's Inv

FareFamily& lMktBKKHKGSegmentYCabin1Family =
    FacBom<FareFamily>::instance().create (l1FareFamilyKey);
FacBomManager::addToListAndMap (lMktBKKHKGSegmentYCabin,
                                lMktBKKHKGSegmentYCabin1Family);
FacBomManager::linkWithParent (lMktBKKHKGSegmentYCabin,
                                lMktBKKHKGSegmentYCabin1Family);

// Step 0.8: booking class level
// Create a BookingClass (Y) for the Segment SIN-BKK, cabin Y,
// fare family 1 on SQ's Inv
BookingClassKey lYBookingClassKey (lY);

BookingClass& lSINBKKSegmentYCabin1FamilyYClass =
    FacBom<BookingClass>::instance().create (lYBookingClassKey);
FacBomManager::addToListAndMap (lSINBKKSegmentYCabin1Family,
                                lSINBKKSegmentYCabin1FamilyYClass);
FacBomManager::linkWithParent (lSINBKKSegmentYCabin1Family,
                                lSINBKKSegmentYCabin1FamilyYClass);

FacBomManager::addToListAndMap (lSINBKKSegmentYCabin,
                                lSINBKKSegmentYCabin1FamilyYClass);
FacBomManager::addToListAndMap (lSINBKKSegment,
                                lSINBKKSegmentYCabin1FamilyYClass);

lSINBKKSegmentYCabin1FamilyYClass.setYield(700);

// Create a BookingClass (Y) for the (mkt) Segment BKK-HKG, cabin Y,
// fare family 1 on SQ's Inv
BookingClass& lMktBKKHKGSegmentYCabin1FamilyYClass =
    FacBom<BookingClass>::instance().create (lYBookingClassKey);
FacBomManager::addToListAndMap (lMktBKKHKGSegmentYCabin1Family,
                                lMktBKKHKGSegmentYCabin1FamilyYClass);

```

```

FacBomManager::linkWithParent (lMktBKKHKGSegmentYCabinlFamily,
                                lMktBKKHKGSegmentYCabinlFamilyYClass);

FacBomManager::addToListAndMap (lMktBKKHKGSegmentYCabin,
                                lMktBKKHKGSegmentYCabinlFamilyYClass);
FacBomManager::addToListAndMap (lMktBKKHKGSegment,
                                lMktBKKHKGSegmentYCabinlFamilyYClass);

lMktBKKHKGSegmentYCabinlFamilyYClass.setYield(700);

// Create a BookingClass (M) for the Segment SIN-BKK, cabin Y,
// fare family 1 on SQ's Inv
const ClassCode_T lM ("M");
BookingClassKey lMBookingClassKey (lM);

BookingClass& lSINBKKSegmentYCabinlFamilyMClass =
    FacBom<BookingClass>::instance().create (lMBookingClassKey);
FacBomManager::addToListAndMap (lSINBKKSegmentYCabinlFamily,
                                lSINBKKSegmentYCabinlFamilyMClass);
FacBomManager::linkWithParent (lSINBKKSegmentYCabinlFamily,
                                lSINBKKSegmentYCabinlFamilyMClass);

FacBomManager::addToListAndMap (lSINBKKSegmentYCabin,
                                lSINBKKSegmentYCabinlFamilyMClass);
FacBomManager::addToListAndMap (lSINBKKSegment,
                                lSINBKKSegmentYCabinlFamilyMClass);

lSINBKKSegmentYCabinlFamilyMClass.setYield(500);

// Create a BookingClass (M) for the (mkt) Segment BKK-HKG, cabin Y,
// fare family 1 on SQ's Inv
BookingClass& lMktBKKHKGSegmentYCabinlFamilyMClass =
    FacBom<BookingClass>::instance().create (lMBookingClassKey);
FacBomManager::addToListAndMap (lMktBKKHKGSegmentYCabinlFamily,
                                lMktBKKHKGSegmentYCabinlFamilyMClass);
FacBomManager::linkWithParent (lMktBKKHKGSegmentYCabinlFamily,
                                lMktBKKHKGSegmentYCabinlFamilyMClass);

FacBomManager::addToListAndMap (lMktBKKHKGSegmentYCabin,
                                lMktBKKHKGSegmentYCabinlFamilyMClass);
FacBomManager::addToListAndMap (lMktBKKHKGSegment,
                                lMktBKKHKGSegmentYCabinlFamilyMClass);

lMktBKKHKGSegmentYCabinlFamilyMClass.setYield(500);

/* ===== */
// Step 0.9: Partner Inventory
// Create a partner Inventory CX for SQ
const InventoryKey lPartnerCXKey ("CX");
Inventory& lPartnerCXInv = FacBom<Inventory>::instance().create (lPartnerCXKey);
FacBomManager::addToListAndMap (lSQInv, lPartnerCXInv);
FacBomManager::linkWithParent (lSQInv, lPartnerCXInv);

// Step 0.9.2 : Flight-date level
lFlightNumber = 12;
lFlightDateKey = FlightDateKey (lFlightNumber, lDate);

FlightDate& lPartnerCX12_20100208_FD =
    FacBom<FlightDate>::instance().create (lFlightDateKey);

```

```

FacBomManager::addToListAndMap (lPartnerCXInv, lPartnerCX12_20100208_FD);
FacBomManager::linkWithParent (lPartnerCXInv, lPartnerCX12_20100208_FD);

lFlightNumber = 1100;
lMktFlightDateKey = FlightDateKey (lFlightNumber, lDate);

FlightDate& lPartnerCX1100_20100208_FD =
    FacBom<FlightDate>::instance().create (lMktFlightDateKey);
FacBomManager::addToListAndMap (lPartnerCXInv, lPartnerCX1100_20100208_FD);
FacBomManager::linkWithParent (lPartnerCXInv, lPartnerCX1100_20100208_FD);

// Step 0.9.3: Segment-date level
lSegmentDateKey = SegmentDateKey (lBKK, lHKG);

SegmentDate& lPartnerBKKHKGSegment =
    FacBom<SegmentDate>::instance().create (lSegmentDateKey);
FacBomManager::addToListAndMap (lPartnerCX12_20100208_FD, lPartnerBKKHKGSegment);
FacBomManager::linkWithParent (lPartnerCX12_20100208_FD, lPartnerBKKHKGSegment);

lPartnerBKKHKGSegment.setBoardingDate (lDate);
lPartnerBKKHKGSegment.setOffDate (lDate);
lPartnerBKKHKGSegment.setBoardingTime (11200);
lPartnerBKKHKGSegment.setOffTime (11540);
lPartnerBKKHKGSegment.setElapsedTime (10240);

lMktSegmentDateKey = SegmentDateKey (lSIN, lBKK);

SegmentDate& lPartnerMktSINBKKSegment =
    FacBom<SegmentDate>::instance().create (lMktSegmentDateKey);
FacBomManager::addToListAndMap (lPartnerCX1100_20100208_FD, lPartnerMktSINBKKSegment);
FacBomManager::linkWithParent (lPartnerCX1100_20100208_FD, lPartnerMktSINBKKSegment);

lPartnerMktSINBKKSegment.setBoardingDate (lDate);
lPartnerMktSINBKKSegment.setOffDate (lDate);
lPartnerMktSINBKKSegment.setBoardingTime (10820);
lPartnerMktSINBKKSegment.setOffTime (11100);
lPartnerMktSINBKKSegment.setElapsedTime (10340);

// Step 0.9.4: Leg-date level
lLegDateKey = LegDateKey (lBKK);

LegDate& lPartnerBKKLeg = FacBom<LegDate>::instance().create (lLegDateKey);
FacBomManager::addToListAndMap (lPartnerCX12_20100208_FD, lPartnerBKKLeg);
FacBomManager::linkWithParent (lPartnerCX12_20100208_FD, lPartnerBKKLeg);

lPartnerBKKLeg.setOffPoint (lHKG);
lPartnerBKKLeg.setBoardingDate (lDate);
lPartnerBKKLeg.setOffDate (lDate);
lPartnerBKKLeg.setBoardingTime (11200);
lPartnerBKKLeg.setOffTime (11540);
lPartnerBKKLeg.setElapsedTime (10240);

FacBomManager::addToListAndMap (lPartnerBKKLeg, lPartnerBKKHKGSegment);
FacBomManager::addToListAndMap (lPartnerBKKHKGSegment, lPartnerBKKLeg);

// Step 9.0.5: segment-cabin level

SegmentCabin& lPartnerBKKHKGSegmentYCabin =

```

```

    FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
    FacBomManager::addToListAndMap (lPartnerBKKHKGSegment, lPartnerBKKHKGSegmentY
    Cabin);
    FacBomManager::linkWithParent (lPartnerBKKHKGSegment, lPartnerBKKHKGSegmentYC
    abin);

    SegmentCabin& lPartnerMktSINBKKSegmentYCabin =
        FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
    FacBomManager::addToListAndMap (lPartnerMktSINBKKSegment, lPartnerMktSINBKKSe
    gmentYCabin);
    FacBomManager::linkWithParent (lPartnerMktSINBKKSegment, lPartnerMktSINBKKSeg
    mentYCabin);

    // Step 9.0.6: leg-cabin level

    LegCabin& lPartnerBKKLegYCabin =
        FacBom<LegCabin>::instance().create (lYLegCabinKey);
    FacBomManager::addToListAndMap (lPartnerBKKLeg, lPartnerBKKLegYCabin);
    FacBomManager::linkWithParent (lPartnerBKKLeg, lPartnerBKKLegYCabin);

    lCapacity = CabinCapacity_T(999);
    lPartnerBKKLegYCabin.setCapacities (lCapacity);
    lPartnerBKKLegYCabin.setAvailabilityPool (lCapacity);

    FacBomManager::addToListAndMap (lPartnerBKKLegYCabin, lPartnerBKKHKGSegmentYC
    abin,
                                   lPartnerBKKHKGSegmentYCabin.getFullerKey());
    FacBomManager::addToListAndMap (lPartnerBKKHKGSegmentYCabin, lPartnerBKKLegYC
    abin,
                                   lPartnerBKKLegYCabin.getFullerKey());

    // Step 9.0.7: fare family level

    FareFamily& lPartnerBKKHKGSegmentYCabinlFamily =
        FacBom<FareFamily>::instance().create (lFareFamilyKey);
    FacBomManager::addToListAndMap (lPartnerBKKHKGSegmentYCabin,
                                   lPartnerBKKHKGSegmentYCabinlFamily);
    FacBomManager::linkWithParent (lPartnerBKKHKGSegmentYCabin,
                                   lPartnerBKKHKGSegmentYCabinlFamily);

    FareFamily& lPartnerMktSINBKKSegmentYCabinlFamily =
        FacBom<FareFamily>::instance().create (lFareFamilyKey);
    FacBomManager::addToListAndMap (lPartnerMktSINBKKSegmentYCabin,
                                   lPartnerMktSINBKKSegmentYCabinlFamily);
    FacBomManager::linkWithParent (lPartnerMktSINBKKSegmentYCabin,
                                   lPartnerMktSINBKKSegmentYCabinlFamily);

    // Step 9.0.8: booking class level

    BookingClass& lPartnerBKKHKGSegmentYCabinlFamilyYClass =
        FacBom<BookingClass>::instance().create (lYBookingClassKey);
    FacBomManager::addToListAndMap (lPartnerBKKHKGSegmentYCabinlFamily,
                                   lPartnerBKKHKGSegmentYCabinlFamilyYClass);
    FacBomManager::linkWithParent (lPartnerBKKHKGSegmentYCabinlFamily,
                                   lPartnerBKKHKGSegmentYCabinlFamilyYClass);

    FacBomManager::addToListAndMap (lPartnerBKKHKGSegmentYCabin,
                                   lPartnerBKKHKGSegmentYCabinlFamilyYClass);
    FacBomManager::addToListAndMap (lPartnerBKKHKGSegment,
                                   lPartnerBKKHKGSegmentYCabinlFamilyYClass);

    BookingClass& lPartnerMktSINBKKSegmentYCabinlFamilyYClass =

```

```

    FacBom<BookingClass>::instance().create (lYBookingClassKey);
    FacBomManager::addToListAndMap (lPartnerMktSINBKKSegmentYCabinlFamily,
                                     lPartnerMktSINBKKSegmentYCabinlFamilyYClass);

    FacBomManager::linkWithParent (lPartnerMktSINBKKSegmentYCabinlFamily,
                                    lPartnerMktSINBKKSegmentYCabinlFamilyYClass);

    FacBomManager::addToListAndMap (lPartnerMktSINBKKSegmentYCabin,
                                     lPartnerMktSINBKKSegmentYCabinlFamilyYClass);

    FacBomManager::addToListAndMap (lPartnerMktSINBKKSegment,
                                     lPartnerMktSINBKKSegmentYCabinlFamilyYClass);

    BookingClass& lPartnerBKKHKGSegmentYCabinlFamilyMClass =
        FacBom<BookingClass>::instance().create (lMBookingClassKey);
    FacBomManager::addToListAndMap (lPartnerBKKHKGSegmentYCabinlFamily,
                                     lPartnerBKKHKGSegmentYCabinlFamilyMClass);
    FacBomManager::linkWithParent (lPartnerBKKHKGSegmentYCabinlFamily,
                                    lPartnerBKKHKGSegmentYCabinlFamilyMClass);

    FacBomManager::addToListAndMap (lPartnerBKKHKGSegmentYCabin,
                                     lPartnerBKKHKGSegmentYCabinlFamilyMClass);
    FacBomManager::addToListAndMap (lPartnerBKKHKGSegment,
                                     lPartnerBKKHKGSegmentYCabinlFamilyMClass);

    BookingClass& lPartnerMktSINBKKSegmentYCabinlFamilyMClass =
        FacBom<BookingClass>::instance().create (lMBookingClassKey);
    FacBomManager::addToListAndMap (lPartnerMktSINBKKSegmentYCabinlFamily,
                                     lPartnerMktSINBKKSegmentYCabinlFamilyMClass);

    FacBomManager::linkWithParent (lPartnerMktSINBKKSegmentYCabinlFamily,
                                    lPartnerMktSINBKKSegmentYCabinlFamilyMClass);

    FacBomManager::addToListAndMap (lPartnerMktSINBKKSegmentYCabin,
                                     lPartnerMktSINBKKSegmentYCabinlFamilyMClass);

    FacBomManager::addToListAndMap (lPartnerMktSINBKKSegment,
                                     lPartnerMktSINBKKSegmentYCabinlFamilyMClass);

    // Step 9.0.9: link SQ inventory objects to Partner CX inventory objects

    FacBomManager::addToList (lSINBKKSegment, lPartnerMktSINBKKSegment);

    lMktBKKHKGSegment.linkWithOperating (lPartnerBKKHKGSegment);

    /* =====
       ===== */

    // Step 1.0: O&D level
    // Create an O&D Date (SQ11/08-FEB-2010/SIN-BKK-SQ1200/08-FEB-2010/BKK-HKG) f
    // or SQ's Inventory
    OnDString_T lSQSINBKKOnDStr = "SQ;11,2010-Feb-08;SIN,BKK";
    OnDString_T lMktSQBKKHKGOnDStr = "SQ;1200,2010-Feb-08;BKK,HKG";
    OnDStringList_T lOnDStringList;
    lOnDStringList.push_back (lSQSINBKKOnDStr);
    lOnDStringList.push_back (lMktSQBKKHKGOnDStr);

    OnDDateKey lOnDDateKey (lOnDStringList);
    OnDDate& lSQ_SINHKG_OnDDate =
        FacBom<OnDDate>::instance().create (lOnDDateKey);

```

```

// Link to the inventory
FacBomManager::addToListAndMap (lSQInv, lSQ_SINHKG_OnDDate);
FacBomManager::linkWithParent (lSQInv, lSQ_SINHKG_OnDDate);

// Add the segments
FacBomManager::addToListAndMap (lSQ_SINHKG_OnDDate, lSINBKKSegment);
FacBomManager::addToListAndMap (lSQ_SINHKG_OnDDate, lMktBKKHKGSegment);

// Add total forecast info for cabin Y.
const MeanStdDevPair_T lMean60StdDev6 (60.0, 6.0);
const WTP_T lWTP750 = 750.0;
const WTPDemandPair_T lWTP750Mean60StdDev6 (lWTP750, lMean60StdDev6);
lSQ_SINHKG_OnDDate.setTotalForecast (lY, lWTP750Mean60StdDev6);

// Add demand info (optional).
// 2 legs here, so 2 CabinClassPair to add in the list.
// First leg: cabin Y, class M.
CabinClassPair_T lCC_YM1 (lY, lM);
// Second leg: cabin Y, class M too.
CabinClassPair_T lCC_YM2 (lY, lM);
CabinClassPairList_T lCabinClassPairList;
lCabinClassPairList.push_back(lCC_YM1);
lCabinClassPairList.push_back(lCC_YM2);
const MeanStdDevPair_T lMean20StdDev2 (20.0, 2.0);
const Yield_T lYield850 = 850.0;
const YieldDemandPair_T lYield850Mean20StdDev2 (lYield850, lMean20StdDev2);
lSQ_SINHKG_OnDDate.setDemandInformation (lCabinClassPairList, lYield850Mean20
    StdDev2);

CabinClassPair_T lCC_YY1 (lY, lY);
CabinClassPair_T lCC_YY2 (lY, lY);
lCabinClassPairList.clear();
lCabinClassPairList.push_back(lCC_YY1);
lCabinClassPairList.push_back(lCC_YY2);
const MeanStdDevPair_T lMean10StdDev1 (10.0, 1.0);
const Yield_T lYield1200 = 1200.0;
const YieldDemandPair_T lYield1200Mean10StdDev1 (lYield1200, lMean10StdDev1);

lSQ_SINHKG_OnDDate.setDemandInformation (lCabinClassPairList, lYield1200Mean1
    0StdDev1);

// Create an O&D Date (SQ11/08-FEB-2010/SIN-BKK) for SQ's Inventory
lOnDStringList.clear();
lOnDStringList.push_back (lSQSINBKKOnDStr);

lOnDDateKey = OnDDateKey(lOnDStringList);
OnDDate& lSQ_SINBKK_OnDDate =
    FacBom<OnDDate>::instance().create (lOnDDateKey);
// Link to the inventory
FacBomManager::addToListAndMap (lSQInv, lSQ_SINBKK_OnDDate);
FacBomManager::linkWithParent (lSQInv, lSQ_SINBKK_OnDDate);

// Add the segments
FacBomManager::addToListAndMap (lSQ_SINBKK_OnDDate, lSINBKKSegment);

// Add total forecast info for cabin Y.
const WTP_T lWTP400 = 400.0;
const WTPDemandPair_T lWTP400Mean60StdDev6 (lWTP400, lMean60StdDev6);
lSQ_SINBKK_OnDDate.setTotalForecast (lY, lWTP400Mean60StdDev6);

// Add demand info (optional).
lCabinClassPairList.clear();

```

```

lCabinClassPairList.push_back(lCC_YM1);
const MeanStdDevPair_T lMean20StdDev1 (20.0, 1.0);
const Yield_T lYield500 = 500.0;
const YieldDemandPair_T lYield500Mean20StdDev1 (lYield500, lMean20StdDev1);
lSQ_SINBKK_OnDDate.setDemandInformation (lCabinClassPairList, lYield500Mean20
    StdDev1);

lCabinClassPairList.clear();
lCabinClassPairList.push_back(lCC_YY1);
const Yield_T lYield700 = 700.0;
const YieldDemandPair_T lYield700Mean20StdDev1 (lYield700, lMean10StdDev1 );
lSQ_SINBKK_OnDDate.setDemandInformation (lCabinClassPairList, lYield700Mean20
    StdDev1);

/*****
***
// Create an O&D Date (SQ1200/08-FEB-2010/BKK-HKG) for SQ's Inventory
lFullKeyList.clear();
lFullKeyList.push_back (lMktSQBKKHKGFullKeyStr);

lOnDDateKey = OnDDateKey(lFullKeyList);
OnDDate& lMktSQ_BKKHKG_OnDDate =
    FacBom<OnDDate>::instance().create (lOnDDateKey);
// Link to the inventory
FacBomManager::addToListAndMap (lSQInv, lMktSQ_BKKHKG_OnDDate);
FacBomManager::linkWithParent (lSQInv, lMktSQ_BKKHKG_OnDDate);

// Add the segments
FacBomManager::addToListAndMap (lMktSQ_BKKHKG_OnDDate, lMktBKKHKGSegment);

// Demand info is not added for purely marketed O&Ds
// Add demand info
// lCabinClassPairList.clear();
// lCabinClassPairList.push_back(lCC_YM2);
// lMktSQ_BKKHKG_OnDDate.setDemandInformation (lCabinClassPairList, 500.0, 20
    .0, 1.0);
*****/

// ///// CX /////
// Step 0.2: Flight-date level
// Create a FlightDate (CX12/08-FEB-2010) for CX's Inventory
lFlightNumber = 12;
//lDate = Date_T (2010, 2, 8);
lFlightDateKey = FlightDateKey (lFlightNumber, lDate);

FlightDate& lCX12_20100208_FD =
    FacBom<FlightDate>::instance().create (lFlightDateKey);
FacBomManager::addToListAndMap (lCXInv, lCX12_20100208_FD);
FacBomManager::linkWithParent (lCXInv, lCX12_20100208_FD);

// Create a (mkt) FlightDate (CX1100/08-FEB-2010) for CX's Inventory
lFlightNumber = 1100;
//lDate = Date_T (2010, 2, 8);
lMktFlightDateKey = FlightDateKey (lFlightNumber, lDate);

FlightDate& lCX1100_20100208_FD =
    FacBom<FlightDate>::instance().create (lMktFlightDateKey);
FacBomManager::addToListAndMap (lCXInv, lCX1100_20100208_FD);
FacBomManager::linkWithParent (lCXInv, lCX1100_20100208_FD);

```

```

// Display the flight-date
// STDAIR_LOG_DEBUG ("FlightDate: " << lAF084_20110320_FD.toString());

// Step 0.3: Segment-date level
// Create a SegmentDate BKK-HKG for CX's Inventory

lSegmentDateKey = SegmentDateKey (lBKK, lHKG);

SegmentDate& lBKKHKGSegment =
    FacBom<SegmentDate>::instance().create (lSegmentDateKey);
FacBomManager::addToListAndMap (lCX12_20100208_FD, lBKKHKGSegment);
FacBomManager::linkWithParent (lCX12_20100208_FD, lBKKHKGSegment);

// Fill the SegmentDate content
lBKKHKGSegment.setBoardingDate (lDate);
lBKKHKGSegment.setOffDate (lDate);
lBKKHKGSegment.setBoardingTime (l1200);
lBKKHKGSegment.setOffTime (l1540);
lBKKHKGSegment.setElapsedTime (l0240);

// Create a second (mkt) SegmentDate (SIN-BKK) for CX's Inventory
lMktSegmentDateKey = SegmentDateKey (lSIN, lBKK);

SegmentDate& lMktSINBKKSegment =
    FacBom<SegmentDate>::instance().create (lMktSegmentDateKey);
FacBomManager::addToListAndMap (lCX1100_20100208_FD, lMktSINBKKSegment);
FacBomManager::linkWithParent (lCX1100_20100208_FD, lMktSINBKKSegment);

// Fill the (mkt) SegmentDate content
lMktSINBKKSegment.setBoardingDate (lDate);
lMktSINBKKSegment.setOffDate (lDate);
lMktSINBKKSegment.setBoardingTime (l0820);
lMktSINBKKSegment.setOffTime (l1100);
lMktSINBKKSegment.setElapsedTime (l0340);

// Step 0.4: Leg-date level
// Create a LegDate (BKK) for CX's Inventory
lLegDateKey = LegDateKey (lBKK);

LegDate& lBKKLeg = FacBom<LegDate>::instance().create (lLegDateKey);
FacBomManager::addToListAndMap (lCX12_20100208_FD, lBKKLeg);
FacBomManager::linkWithParent (lCX12_20100208_FD, lBKKLeg);

// Fill the LegDate content
lBKKLeg.setOffPoint (lHKG);
lBKKLeg.setBoardingDate (lDate);
lBKKLeg.setOffDate (lDate);
lBKKLeg.setBoardingTime (l1200);
lBKKLeg.setOffTime (l1540);
lBKKLeg.setElapsedTime (l0240);

// Display the leg-date
// STDAIR_LOG_DEBUG ("LegDate: " << lCDGLeg.toString());

// Link the segment-dates with the leg-dates
FacBomManager::addToListAndMap (lBKKLeg, lBKKHKGSegment);
FacBomManager::addToListAndMap (lBKKHKGSegment, lBKKLeg);

// Step 0.5: segment-cabin level
// Create a SegmentCabin (Y) for the Segment BKK-HKG of CX's Inventory
SegmentCabin& lBKKHKGSegmentYCabin =
    FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);

```

```

FacBomManager::addToListAndMap (lBKKHKGSegment, lBKKHKGSegmentYCabin);
FacBomManager::linkWithParent (lBKKHKGSegment, lBKKHKGSegmentYCabin);

// Create a SegmentCabin (Y) for the (mkt) Segment SIN-BKK of CX's Inventory
SegmentCabin& lMktSINBKKSegmentYCabin =
    FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
FacBomManager::addToListAndMap (lMktSINBKKSegment, lMktSINBKKSegmentYCabin);
FacBomManager::linkWithParent (lMktSINBKKSegment, lMktSINBKKSegmentYCabin);

// Step 0.6: leg-cabin level
// Create a LegCabin (Y) for the Leg BKK-HKG on CX's Inventory
LegCabin& lBKKLegYCabin =
    FacBom<LegCabin>::instance().create (lYLegCabinKey);
FacBomManager::addToListAndMap (lBKKLeg, lBKKLegYCabin);
FacBomManager::linkWithParent (lBKKLeg, lBKKLegYCabin);

lCapacity = CabinCapacity_T(100);
lBKKLegYCabin.setCapacities (lCapacity);
lBKKLegYCabin.setAvailabilityPool (lCapacity);

// Link the segment-dates with the leg-dates
FacBomManager::addToListAndMap (lBKKLegYCabin, lBKKHKGSegmentYCabin,
                                lBKKHKGSegmentYCabin.getFullerKey());
FacBomManager::addToListAndMap (lBKKHKGSegmentYCabin, lBKKLegYCabin,
                                lBKKLegYCabin.getFullerKey());

// Step 0.7: fare family level
// Create a fareFamily (1) for the Segment BKK-HKG, cabin Y on CX's Inv
FareFamily& lBKKHKGSegmentYCabin1Family =
    FacBom<FareFamily>::instance().create (l1FareFamilyKey);
FacBomManager::addToListAndMap (lBKKHKGSegmentYCabin,
                                lBKKHKGSegmentYCabin1Family);
FacBomManager::linkWithParent (lBKKHKGSegmentYCabin,
                                lBKKHKGSegmentYCabin1Family);

// Create a FareFamily (1) for the (mkt) Segment SIN-BKK, cabin Y on CX's Inv
FareFamily& lMktSINBKKSegmentYCabin1Family =
    FacBom<FareFamily>::instance().create (l1FareFamilyKey);
FacBomManager::addToListAndMap (lMktSINBKKSegmentYCabin,
                                lMktSINBKKSegmentYCabin1Family);
FacBomManager::linkWithParent (lMktSINBKKSegmentYCabin,
                                lMktSINBKKSegmentYCabin1Family);

// Step 0.8: booking class level
// Create a BookingClass (Y) for the
// Segment BKK-HKG, cabin Y, fare family 1 on CX's Inv
BookingClass& lBKKHKGSegmentYCabin1FamilyYClass =
    FacBom<BookingClass>::instance().create (lYBookingClassKey);
FacBomManager::addToListAndMap (lBKKHKGSegmentYCabin1Family,
                                lBKKHKGSegmentYCabin1FamilyYClass);
FacBomManager::linkWithParent (lBKKHKGSegmentYCabin1Family,
                                lBKKHKGSegmentYCabin1FamilyYClass);

FacBomManager::addToListAndMap (lBKKHKGSegmentYCabin,
                                lBKKHKGSegmentYCabin1FamilyYClass);
FacBomManager::addToListAndMap (lBKKHKGSegment,
                                lBKKHKGSegmentYCabin1FamilyYClass);

lBKKHKGSegmentYCabin1FamilyYClass.setYield(700);

```

```

// Create a BookingClass (Y) for the (mkt) Segment SIN-BKK, cabin Y,
// fare family 1 on CX's Inv
BookingClass& lMktSINBKKSegmentYCabinlFamilyYClass =
    FacBom<BookingClass>::instance().create (lYBookingClassKey);
FacBomManager::addToListAndMap (lMktSINBKKSegmentYCabinlFamily,
                                lMktSINBKKSegmentYCabinlFamilyYClass);
FacBomManager::linkWithParent (lMktSINBKKSegmentYCabinlFamily,
                               lMktSINBKKSegmentYCabinlFamilyYClass);

FacBomManager::addToListAndMap (lMktSINBKKSegmentYCabin,
                                lMktSINBKKSegmentYCabinlFamilyYClass);
FacBomManager::addToListAndMap (lMktSINBKKSegment,
                                lMktSINBKKSegmentYCabinlFamilyYClass);

lMktSINBKKSegmentYCabinlFamilyYClass.setYield(700);

//Create a BookingClass (M) for the
// Segment BKK-HKG, cabin Y, fare family 1 on CX's Inv
BookingClass& lBKKHKGSegmentYCabinlFamilyMClass =
    FacBom<BookingClass>::instance().create (lMBookingClassKey);
FacBomManager::addToListAndMap (lBKKHKGSegmentYCabinlFamily,
                                lBKKHKGSegmentYCabinlFamilyMClass);
FacBomManager::linkWithParent (lBKKHKGSegmentYCabinlFamily,
                               lBKKHKGSegmentYCabinlFamilyMClass);

FacBomManager::addToListAndMap (lBKKHKGSegmentYCabin,
                                lBKKHKGSegmentYCabinlFamilyMClass);
FacBomManager::addToListAndMap (lBKKHKGSegment,
                                lBKKHKGSegmentYCabinlFamilyMClass);

lBKKHKGSegmentYCabinlFamilyMClass.setYield(500);

// Create a BookingClass (M) for the (mkt) Segment SIN-BKK, cabin Y,
// fare family 1 on CX's Inv
BookingClass& lMktSINBKKSegmentYCabinlFamilyMClass =
    FacBom<BookingClass>::instance().create (lMBookingClassKey);
FacBomManager::addToListAndMap (lMktSINBKKSegmentYCabinlFamily,
                                lMktSINBKKSegmentYCabinlFamilyMClass);
FacBomManager::linkWithParent (lMktSINBKKSegmentYCabinlFamily,
                               lMktSINBKKSegmentYCabinlFamilyMClass);

FacBomManager::addToListAndMap (lMktSINBKKSegmentYCabin,
                                lMktSINBKKSegmentYCabinlFamilyMClass);
FacBomManager::addToListAndMap (lMktSINBKKSegment,
                                lMktSINBKKSegmentYCabinlFamilyMClass);

lMktSINBKKSegmentYCabinlFamilyMClass.setYield(500);

/* =====
===== */
// Step 0.9: Partner Inventory
// Create a partner Inventory SQ for CX
const InventoryKey lPartnerSQKey ("SQ");
Inventory& lPartnerSQInv = FacBom<Inventory>::instance().create (lPartnerSQKey);
FacBomManager::addToListAndMap (lCXInv, lPartnerSQInv);
FacBomManager::linkWithParent (lCXInv, lPartnerSQInv);

// Step 0.9.2 : Flight-date level
lFlightNumber = 11;
lFlightDateKey = FlightDateKey (lFlightNumber, lDate);

```

```

FlightDate& lPartnerSQ11_20100208_FD =
    FacBom<FlightDate>::instance().create (lFlightDateKey);
FacBomManager::addToListAndMap (lPartnerSQInv, lPartnerSQ11_20100208_FD);
FacBomManager::linkWithParent (lPartnerSQInv, lPartnerSQ11_20100208_FD);

lFlightNumber = 1200;
lMktFlightDateKey = FlightDateKey (lFlightNumber, lDate);

FlightDate& lPartnerSQ1200_20100208_FD =
    FacBom<FlightDate>::instance().create (lMktFlightDateKey);
FacBomManager::addToListAndMap (lPartnerSQInv, lPartnerSQ1200_20100208_FD);
FacBomManager::linkWithParent (lPartnerSQInv, lPartnerSQ1200_20100208_FD);

// Step 0.9.3: Segment-date level
lSegmentDateKey = SegmentDateKey (lSIN, lBKK);

SegmentDate& lPartnerSINBKKSegment =
    FacBom<SegmentDate>::instance().create (lSegmentDateKey);
FacBomManager::addToListAndMap (lPartnerSQ11_20100208_FD, lPartnerSINBKKSegment);
FacBomManager::linkWithParent (lPartnerSQ11_20100208_FD, lPartnerSINBKKSegment);

lPartnerSINBKKSegment.setBoardingDate (lDate);
lPartnerSINBKKSegment.setOffDate (lDate);
lPartnerSINBKKSegment.setBoardingTime (l0820);
lPartnerSINBKKSegment.setOffTime (l1100);
lPartnerSINBKKSegment.setElapsedTime (l0340);

lMktSegmentDateKey = SegmentDateKey (lBKK, lHKG);

SegmentDate& lPartnerMktBKKHKGSegment =
    FacBom<SegmentDate>::instance().create (lMktSegmentDateKey);
FacBomManager::addToListAndMap (lPartnerSQ1200_20100208_FD, lPartnerMktBKKHKGSegment);
FacBomManager::linkWithParent (lPartnerSQ1200_20100208_FD, lPartnerMktBKKHKGSegment);

lPartnerMktBKKHKGSegment.setBoardingDate (lDate);
lPartnerMktBKKHKGSegment.setOffDate (lDate);
lPartnerMktBKKHKGSegment.setBoardingTime (l1200);
lPartnerMktBKKHKGSegment.setOffTime (l1540);
lPartnerMktBKKHKGSegment.setElapsedTime (l0240);

// Step 0.9.4: Leg-date level
lLegDateKey = LegDateKey (lSIN);

LegDate& lPartnerSINLeg = FacBom<LegDate>::instance().create (lLegDateKey);
FacBomManager::addToListAndMap (lPartnerSQ11_20100208_FD, lPartnerSINLeg);
FacBomManager::linkWithParent (lPartnerSQ11_20100208_FD, lPartnerSINLeg);

lPartnerSINLeg.setOffPoint (lBKK);
lPartnerSINLeg.setBoardingDate (lDate);
lPartnerSINLeg.setOffDate (lDate);
lPartnerSINLeg.setBoardingTime (l0820);
lPartnerSINLeg.setOffTime (l1100);
lPartnerSINLeg.setElapsedTime (l0340);

FacBomManager::addToListAndMap (lPartnerSINLeg, lPartnerSINBKKSegment);
FacBomManager::addToListAndMap (lPartnerSINBKKSegment, lPartnerSINLeg);

// Step 9.0.5: segment-cabin level

```

```

SegmentCabin& lPartnerSINBKKSegmentYCabin =
    FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
FacBomManager::addToListAndMap (lPartnerSINBKKSegment, lPartnerSINBKKSegmentYCabin);
FacBomManager::linkWithParent (lPartnerSINBKKSegment, lPartnerSINBKKSegmentYCabin);

SegmentCabin& lPartnerMktBKKHKGSegmentYCabin =
    FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
FacBomManager::addToListAndMap (lPartnerMktBKKHKGSegment, lPartnerMktBKKHKGSegmentYCabin);
FacBomManager::linkWithParent (lPartnerMktBKKHKGSegment, lPartnerMktBKKHKGSegmentYCabin);

// Step 9.0.6: leg-cabin level

LegCabin& lPartnerSINLegYCabin =
    FacBom<LegCabin>::instance().create (lYLegCabinKey);
FacBomManager::addToListAndMap (lPartnerSINLeg, lPartnerSINLegYCabin);
FacBomManager::linkWithParent (lPartnerSINLeg, lPartnerSINLegYCabin);

lCapacity = CabinCapacity_T(999);
lPartnerSINLegYCabin.setCapacities (lCapacity);
lPartnerSINLegYCabin.setAvailabilityPool (lCapacity);

FacBomManager::addToListAndMap (lPartnerSINLegYCabin, lPartnerSINBKKSegmentYCabin,
                                lPartnerSINBKKSegmentYCabin.getFullerKey());
FacBomManager::addToListAndMap (lPartnerSINBKKSegmentYCabin, lPartnerSINLegYCabin,
                                lPartnerSINLegYCabin.getFullerKey());

// Step 9.0.7: fare family level

FareFamily& lPartnerSINBKKSegmentYCabinlFamily =
    FacBom<FareFamily>::instance().create (lFareFamilyKey);
FacBomManager::addToListAndMap (lPartnerSINBKKSegmentYCabin,
                                lPartnerSINBKKSegmentYCabinlFamily);
FacBomManager::linkWithParent (lPartnerSINBKKSegmentYCabin,
                                lPartnerSINBKKSegmentYCabinlFamily);

FareFamily& lPartnerMktBKKHKGSegmentYCabinlFamily =
    FacBom<FareFamily>::instance().create (lFareFamilyKey);
FacBomManager::addToListAndMap (lPartnerMktBKKHKGSegmentYCabin,
                                lPartnerMktBKKHKGSegmentYCabinlFamily);
FacBomManager::linkWithParent (lPartnerMktBKKHKGSegmentYCabin,
                                lPartnerMktBKKHKGSegmentYCabinlFamily);

// Step 9.0.8: booking class level

BookingClass& lPartnerSINBKKSegmentYCabinlFamilyYClass =
    FacBom<BookingClass>::instance().create (lYBookingClassKey);
FacBomManager::addToListAndMap (lPartnerSINBKKSegmentYCabinlFamily,
                                lPartnerSINBKKSegmentYCabinlFamilyYClass);
FacBomManager::linkWithParent (lPartnerSINBKKSegmentYCabinlFamily,
                                lPartnerSINBKKSegmentYCabinlFamilyYClass);

FacBomManager::addToListAndMap (lPartnerSINBKKSegmentYCabin,
                                lPartnerSINBKKSegmentYCabinlFamilyYClass);
FacBomManager::addToListAndMap (lPartnerSINBKKSegment,

```

```

lPartnerSINBKKSegmentYCabinlFamilyYClass);

BookingClass& lPartnerMktBKKHKGSegmentYCabinlFamilyYClass =
    FacBom<BookingClass>::instance().create (lYBookingClassKey);
FacBomManager::addToListAndMap (lPartnerMktBKKHKGSegmentYCabinlFamily,
                                lPartnerMktBKKHKGSegmentYCabinlFamilyYClass);

FacBomManager::linkWithParent (lPartnerMktBKKHKGSegmentYCabinlFamily,
                                lPartnerMktBKKHKGSegmentYCabinlFamilyYClass);

FacBomManager::addToListAndMap (lPartnerMktBKKHKGSegmentYCabin,
                                lPartnerMktBKKHKGSegmentYCabinlFamilyYClass);

FacBomManager::addToListAndMap (lPartnerMktBKKHKGSegment,
                                lPartnerMktBKKHKGSegmentYCabinlFamilyYClass);

BookingClass& lPartnerSINBKKSegmentYCabinlFamilyMClass =
    FacBom<BookingClass>::instance().create (lMBookingClassKey);
FacBomManager::addToListAndMap (lPartnerSINBKKSegmentYCabinlFamily,
                                lPartnerSINBKKSegmentYCabinlFamilyMClass);
FacBomManager::linkWithParent (lPartnerSINBKKSegmentYCabinlFamily,
                                lPartnerSINBKKSegmentYCabinlFamilyMClass);

FacBomManager::addToListAndMap (lPartnerSINBKKSegmentYCabin,
                                lPartnerSINBKKSegmentYCabinlFamilyMClass);
FacBomManager::addToListAndMap (lPartnerSINBKKSegment,
                                lPartnerSINBKKSegmentYCabinlFamilyMClass);

BookingClass& lPartnerMktBKKHKGSegmentYCabinlFamilyMClass =
    FacBom<BookingClass>::instance().create (lMBookingClassKey);
FacBomManager::addToListAndMap (lPartnerMktBKKHKGSegmentYCabinlFamily,
                                lPartnerMktBKKHKGSegmentYCabinlFamilyMClass);

FacBomManager::linkWithParent (lPartnerMktBKKHKGSegmentYCabinlFamily,
                                lPartnerMktBKKHKGSegmentYCabinlFamilyMClass);

FacBomManager::addToListAndMap (lPartnerMktBKKHKGSegmentYCabin,
                                lPartnerMktBKKHKGSegmentYCabinlFamilyMClass);

FacBomManager::addToListAndMap (lPartnerMktBKKHKGSegment,
                                lPartnerMktBKKHKGSegmentYCabinlFamilyMClass);

// Step 9.0.9: link CX inventory objects to Partner SQ inventory objects
FacBomManager::addToList (lBKKHKGSegment, lPartnerMktBKKHKGSegment);

lMktSINBKKSegment.linkWithOperating (lPartnerSINBKKSegment);

/* =====
===== */

// Step 1.0: O&D level
// Create an O&D Date (CX1100/08-FEB-2010/SIN-BKK-CX12/08-FEB-2010/BKK-HKG) f
or CX's Inventory
OnDString_T lMktCXsINBKKOnDStr = "CX;1100,2010-Feb-08;SIN,BKK";
OnDString_T lCXBKKHKGOnDStr = "CX;12,2010-Feb-08;BKK,HKG";
lOnDStringList.clear();
lOnDStringList.push_back (lMktCXsINBKKOnDStr);
lOnDStringList.push_back (lCXBKKHKGOnDStr);

lOnDDateKey = OnDDateKey(lOnDStringList);

```

```

OnDDate& lCX_SINHKG_OnDDate =
    FacBom<OnDDate>::instance().create (lOnDDateKey);
// Link to the inventory
FacBomManager::addToListAndMap (lCXInv, lCX_SINHKG_OnDDate);
FacBomManager::linkWithParent (lCXInv, lCX_SINHKG_OnDDate);

// Add the segments
FacBomManager::addToListAndMap (lCX_SINHKG_OnDDate, lMktSINBKKSegment);
FacBomManager::addToListAndMap (lCX_SINHKG_OnDDate, lBKKHKGSegment);

// Add total forecast info for cabin Y.
lCX_SINHKG_OnDDate.setTotalForecast (lY, lWTP750Mean60StdDev6);

// Add demand info
lCabinClassPairList.clear();
lCabinClassPairList.push_back(lCC_YM1);
lCabinClassPairList.push_back(lCC_YM2);
lCX_SINHKG_OnDDate.setDemandInformation (lCabinClassPairList, lYield850Mean20
    StdDev2);

lCabinClassPairList.clear();
lCabinClassPairList.push_back(lCC_YY1);
lCabinClassPairList.push_back(lCC_YY2);
lCX_SINHKG_OnDDate.setDemandInformation (lCabinClassPairList, lYield1200Mean1
    0StdDev1);

/*****
    *****
// Create an O&D Date (CX1100/08-FEB-2010/SIN-BKK) for CX's Inventory
lFullKeyList.clear();
lFullKeyList.push_back (lMktCX_SINBKKFullKeyStr);

lOnDDateKey = OnDDateKey(lFullKeyList);
OnDDate& lMktCX_SINBKK_OnDDate =
    FacBom<OnDDate>::instance().create (lOnDDateKey);
// Link to the inventory
FacBomManager::addToListAndMap (lCXInv, lMktCX_SINBKK_OnDDate);
FacBomManager::linkWithParent (lCXInv, lMktCX_SINBKK_OnDDate);

// Add the segments
FacBomManager::addToListAndMap (lMktCX_SINBKK_OnDDate, lMktSINBKKSegment);

// Demand info is not added for purely marketed O&Ds
// Add demand info
// lCabinClassPairList.clear();
// lCabinClassPairList.push_back(lCC_YM1);
// lMktCX_SINBKK_OnDDate.setDemandInformation (lCabinClassPairList, 500.0, 20
    .0, 1.0);
*****/

// Create an O&D Date (CX12/08-FEB-2010/BKK-HKG) for CX's Inventory
lOnDStringList.clear();
lOnDStringList.push_back (lCXBKKHKGOnDStr);

lOnDDateKey = OnDDateKey(lOnDStringList);
OnDDate& lCX_BKKHKG_OnDDate =
    FacBom<OnDDate>::instance().create (lOnDDateKey);
// Link to the inventory
FacBomManager::addToListAndMap (lCXInv, lCX_BKKHKG_OnDDate);
FacBomManager::linkWithParent (lCXInv, lCX_BKKHKG_OnDDate);

```

```

// Add the segments
FacBomManager::addToListAndMap (lCX_BKKHKG_OnDDate, lBKKHKGSegment);

// Add total forecast info for cabin Y.
lCX_BKKHKG_OnDDate.setTotalForecast (lY, lWTP400Mean60StdDev6);

// Add demand info
lCabinClassPairList.clear();
lCabinClassPairList.push_back(lCC_YM2);
lCX_BKKHKG_OnDDate.setDemandInformation (lCabinClassPairList, lYield500Mean20
    StdDev1);

lCabinClassPairList.clear();
lCabinClassPairList.push_back(lCC_YM2);
const YieldDemandPair_T lYield700Mean10StdDev1 (lYield700, lMean10StdDev1 );
lCX_BKKHKG_OnDDate.setDemandInformation (lCabinClassPairList, lYield700Mean10
    StdDev1);

/*=====
=====
=====
=====*/
// Schedule:
// SQ:
// Step 1: flight period level
// Create a flight period for SQ11:
const DoWSrtuct lDoWSrtuct ("1111111");
const Date_T lDateRangeStart (2010, boost::gregorian::Feb, 8);
const Date_T lDateRangeEnd (2010, boost::gregorian::Feb, 9);
const DatePeriod_T lDatePeriod (lDateRangeStart, lDateRangeEnd);
const PeriodStruct lPeriodStruct (lDatePeriod, lDoWSrtuct);

lFlightNumber = FlightNumber_T (11);

FlightPeriodKey lFlightPeriodKey (lFlightNumber, lPeriodStruct);

FlightPeriod& lSQ11FlightPeriod =
    FacBom<FlightPeriod>::instance().create (lFlightPeriodKey);
FacBomManager::addToListAndMap (lSQInv, lSQ11FlightPeriod);
FacBomManager::linkWithParent (lSQInv, lSQ11FlightPeriod);

// Step 2: segment period level
// Create a segment period for SIN-BKK:

SegmentPeriodKey lSegmentPeriodKey (lSIN, lBKK);

SegmentPeriod& lSINBKKSegmentPeriod =
    FacBom<SegmentPeriod>::instance().create (lSegmentPeriodKey);
FacBomManager::addToListAndMap (lSQ11FlightPeriod, lSINBKKSegmentPeriod);
FacBomManager::linkWithParent (lSQ11FlightPeriod, lSINBKKSegmentPeriod);

lSINBKKSegmentPeriod.setBoardingTime (10820);
lSINBKKSegmentPeriod.setOffTime (11100);
lSINBKKSegmentPeriod.setElapsedTime (10340);
ClassList_String_T lYM ("YM");
lSINBKKSegmentPeriod.addCabinBookingClassList (lY, lYM);

// CX:
// Step 1: flight period level
// Create a flight period for CX12:

```

```

lFlightNumber = FlightNumber_T (12);

lFlightPeriodKey = FlightPeriodKey(lFlightNumber, lPeriodStruct);

FlightPeriod& lCX12FlightPeriod =
    FacBom<FlightPeriod>::instance().create (lFlightPeriodKey);
FacBomManager::addToListAndMap (lCXInv, lCX12FlightPeriod);
FacBomManager::linkWithParent (lCXInv, lCX12FlightPeriod);

// Step 2: segment period level
// Create a segment period for BKK-HKG:

lSegmentPeriodKey = SegmentPeriodKey (lBKK, lHKG);

SegmentPeriod& lBKKHKGSegmentPeriod =
    FacBom<SegmentPeriod>::instance().create (lSegmentPeriodKey);
FacBomManager::addToListAndMap (lCX12FlightPeriod, lBKKHKGSegmentPeriod);
FacBomManager::linkWithParent (lCX12FlightPeriod, lBKKHKGSegmentPeriod);

lBKKHKGSegmentPeriod.setBoardingTime (l1200);
lBKKHKGSegmentPeriod.setOffTime (l1540);
lBKKHKGSegmentPeriod.setElapsedTime (l0240);
lBKKHKGSegmentPeriod.addCabinBookingClassList (lY,lYM);
}

// ////////////////////////////////////////
void CmdBomManager::buildPartnershipsSamplePricing (BomRoot& ioBomRoot) {

    /*=====
    =====*/
    // First airport pair SIN-BKK.
    // Set the airport-pair primary key.
    AirportPairKey lAirportPairKey ("SIN", "BKK");

    // Create the AirportPairKey object and link it to the ioBomRoot object.
    AirportPair& lSINBKKAirportPair =
        FacBom<AirportPair>::instance().create (lAirportPairKey);
    FacBomManager::addToListAndMap (ioBomRoot, lSINBKKAirportPair);
    FacBomManager::linkWithParent (ioBomRoot, lSINBKKAirportPair);

    // Set the fare date-period primary key.
    const Date_T lDateRangeStart (2010, boost::gregorian::Jan, 15);
    const Date_T lDateRangeEnd (2010, boost::gregorian::Dec, 31);
    const DatePeriod_T lDateRange (lDateRangeStart, lDateRangeEnd);
    const DatePeriodKey lDatePeriodKey (lDateRange);

    // Create the DatePeriodKey object and link it to the PosChannel object.
    DatePeriod& lSINBKKDatePeriod =
        FacBom<DatePeriod>::instance().create (lDatePeriodKey);
    FacBomManager::addToListAndMap (lSINBKKAirportPair, lSINBKKDatePeriod);
    FacBomManager::linkWithParent (lSINBKKAirportPair, lSINBKKDatePeriod);

    // Set the point-of-sale-channel primary key.
    PosChannelKey lPosChannelKey ("SIN","IN");

    // Create the PositionKey object and link it to the AirportPair object.
    PosChannel& lSINPosChannel =
        FacBom<PosChannel>::instance().create (lPosChannelKey);
    FacBomManager::addToListAndMap (lSINBKKDatePeriod, lSINPosChannel);
    FacBomManager::linkWithParent (lSINBKKDatePeriod, lSINPosChannel);

```

```

// Set the fare time-period primary key.
const Time_T lTimeRangeStart (0, 0, 0);
const Time_T lTimeRangeEnd (23, 0, 0);
const TimePeriodKey lFareTimePeriodKey (lTimeRangeStart,
                                         lTimeRangeEnd);

// Create the TimePeriodKey and link it to the DatePeriod object.
TimePeriod& lSINBKKFareTimePeriod =
    FacBom<TimePeriod>::instance().create (lFareTimePeriodKey);
FacBomManager::addToListAndMap (lSINPosChannel, lSINBKKFareTimePeriod);
FacBomManager::linkWithParent (lSINPosChannel, lSINBKKFareTimePeriod);

// Generate the FareRule
const FareFeaturesKey lFareFeaturesKey (TRIP_TYPE_ONE_WAY,
                                         NO_ADVANCE_PURCHASE,
                                         SATURDAY_STAY,
                                         CHANGE_FEES,
                                         NON_REFUNDABLE,
                                         NO_STAY_DURATION);

// Create the FareFeaturesKey and link it to the TimePeriod object.
FareFeatures& lSINBKKFareFeatures =
    FacBom<FareFeatures>::instance().create (lFareFeaturesKey);
FacBomManager::addToListAndMap (lSINBKKFareTimePeriod, lSINBKKFareFeatures);
FacBomManager::linkWithParent (lSINBKKFareTimePeriod, lSINBKKFareFeatures);

// Generate Segment Features and link them to their FareRule.
AirlineCodeList_T lSQAirlineCodeList;
lSQAirlineCodeList.push_back ("SQ");

ClassList_StringList_T lYClassCodeList;
lYClassCodeList.push_back ("Y");
const AirlineClassListKey lSQAirlineYClassListKey (lSQAirlineCodeList,
                                                    lYClassCodeList);

ClassList_StringList_T lMClassCodeList;
lMClassCodeList.push_back ("M");
const AirlineClassListKey lSQAirlineMClassListKey (lSQAirlineCodeList,
                                                    lMClassCodeList);

// Create the AirlineClassListKey and link it to the FareFeatures object.
AirlineClassList& lSQAirlineYClassList =
    FacBom<AirlineClassList>::instance().create (lSQAirlineYClassListKey);
lSQAirlineYClassList.setFare(700);
FacBomManager::addToListAndMap (lSINBKKFareFeatures, lSQAirlineYClassList);
FacBomManager::linkWithParent (lSINBKKFareFeatures, lSQAirlineYClassList);

AirlineClassList& lSQAirlineMClassList =
    FacBom<AirlineClassList>::instance().create (lSQAirlineMClassListKey);
lSQAirlineMClassList.setFare(500);
FacBomManager::addToListAndMap (lSINBKKFareFeatures, lSQAirlineMClassList);
FacBomManager::linkWithParent (lSINBKKFareFeatures, lSQAirlineMClassList);

/*=====*/
// Second airport pair BKK-HKG.
// Set the airport-pair primary key.
lAirportPairKey = AirportPairKey ("BKK", "HKG");

```

```

// Create the AirportPairKey object and link it to the ioBomRoot object.
AirportPair& lBKKHKGAirportPair =
    FacBom<AirportPair>::instance().create (lAirportPairKey);
FacBomManager::addToListAndMap (ioBomRoot, lBKKHKGAirportPair);
FacBomManager::linkWithParent (ioBomRoot, lBKKHKGAirportPair);

// Set the fare date-period primary key.
// Use the same as previously.

// Create the DatePeriodKey object and link it to the PosChannel object.
DatePeriod& lBKKHKGDatePeriod =
    FacBom<DatePeriod>::instance().create (lDatePeriodKey);
FacBomManager::addToListAndMap (lBKKHKGAirportPair, lBKKHKGDatePeriod);
FacBomManager::linkWithParent (lBKKHKGAirportPair, lBKKHKGDatePeriod);

// Set the point-of-sale-channel primary key.
lPosChannelKey = PosChannelKey("BKK","IN");

// Create the PositionKey object and link it to the AirportPair object.
PosChannel& lBKKPosChannel =
    FacBom<PosChannel>::instance().create (lPosChannelKey);
FacBomManager::addToListAndMap (lBKKHKGDatePeriod, lBKKPosChannel);
FacBomManager::linkWithParent (lBKKHKGDatePeriod, lBKKPosChannel);

// Set the fare time-period primary key.
// Use the same as previously.

// Create the TimePeriodKey and link it to the DatePeriod object.
TimePeriod& lBKKHKGTimePeriod =
    FacBom<TimePeriod>::instance().create (lFareTimePeriodKey);
FacBomManager::addToListAndMap (lBKKPosChannel, lBKKHKGTimePeriod);
FacBomManager::linkWithParent (lBKKPosChannel, lBKKHKGTimePeriod);

// Generate the FareRule
// Use the same key as previously.

// Create the FareFeaturesKey and link it to the TimePeriod object.
FareFeatures& lBKKHKGTimePeriodFeatures =
    FacBom<FareFeatures>::instance().create (lFareFeaturesKey);
FacBomManager::addToListAndMap (lBKKHKGTimePeriod, lBKKHKGTimePeriodFeatures);
FacBomManager::linkWithParent (lBKKHKGTimePeriod, lBKKHKGTimePeriodFeatures);

// Generate Segment Features and link them to their FareRule.
AirlineCodeList_T lCXAirlineCodeList;
lCXAirlineCodeList.push_back ("CX");

const AirlineClassListKey lCXAirlineYClassListKey (lCXAirlineCodeList,
                                                    lYClassCodeList);

const AirlineClassListKey lCXAirlineMClassListKey (lCXAirlineCodeList,
                                                    lMClassCodeList);

// Create the AirlineClassListKey and link it to the FareFeatures object.
AirlineClassList& lCXAirlineYClassList =
    FacBom<AirlineClassList>::instance().create (lCXAirlineYClassListKey);
lCXAirlineYClassList.setFare(700);
FacBomManager::addToListAndMap (lBKKHKGTimePeriodFeatures, lCXAirlineYClassList);
FacBomManager::linkWithParent (lBKKHKGTimePeriodFeatures, lCXAirlineYClassList);

AirlineClassList& lCXAirlineMClassList =

```

```

    FacBom<AirlineClassList>::instance().create (lCXAirlineMClassListKey);
    lCXAirlineMClassList.setFare(500);
    FacBomManager::addToListAndMap (lBKHKHKGfareFeatures, lCXAirlineMClassList);
    FacBomManager::linkWithParent (lBKHKHKGfareFeatures, lCXAirlineMClassList);

    /*=====
    =====*/
    // Third airport pair SIN-HKG.
    // Set the airport-pair primary key.
    lAirportPairKey = AirportPairKey ("SIN", "HKG");

    // Create the AirportPairKey object and link it to the ioBomRoot object.
    AirportPair& lSINHKGAirportPair =
        FacBom<AirportPair>::instance().create (lAirportPairKey);
    FacBomManager::addToListAndMap (ioBomRoot, lSINHKGAirportPair);
    FacBomManager::linkWithParent (ioBomRoot, lSINHKGAirportPair);

    // Set the fare date-period primary key.
    // Use the same as previously.

    // Create the DatePeriodKey object and link it to the PosChannel object.
    DatePeriod& lSINHKGDatePeriod =
        FacBom<DatePeriod>::instance().create (lDatePeriodKey);
    FacBomManager::addToListAndMap (lSINHKGAirportPair, lSINHKGDatePeriod);
    FacBomManager::linkWithParent (lSINHKGAirportPair, lSINHKGDatePeriod);

    // Set the point-of-sale-channel primary key.
    lPosChannelKey = PosChannelKey("SIN","IN");

    // Create the PositionKey object and link it to the AirportPair object.
    PosChannel& lOnDSINPosChannel =
        FacBom<PosChannel>::instance().create (lPosChannelKey);
    FacBomManager::addToListAndMap (lSINHKGDatePeriod, lOnDSINPosChannel);
    FacBomManager::linkWithParent (lSINHKGDatePeriod, lOnDSINPosChannel);

    // Set the fare time-period primary key.
    // Use the same as previously.

    // Create the TimePeriodKey and link it to the DatePeriod object.
    TimePeriod& lSINHKGfareTimePeriod =
        FacBom<TimePeriod>::instance().create (lFareTimePeriodKey);
    FacBomManager::addToListAndMap (lOnDSINPosChannel, lSINHKGfareTimePeriod);
    FacBomManager::linkWithParent (lOnDSINPosChannel, lSINHKGfareTimePeriod);

    // Generate the FareRule
    // Use the same key as previously.

    // Create the FareFeaturesKey and link it to the TimePeriod object.
    FareFeatures& lSINHKGfareFeatures =
        FacBom<FareFeatures>::instance().create (lFareFeaturesKey);
    FacBomManager::addToListAndMap (lSINHKGfareTimePeriod, lSINHKGfareFeatures);
    FacBomManager::linkWithParent (lSINHKGfareTimePeriod, lSINHKGfareFeatures);

    // Generate Segment Features and link them to their FareRule.
    AirlineCodeList_T lSQ_CXAirlineCodeList;
    lSQ_CXAirlineCodeList.push_back ("SQ");
    lSQ_CXAirlineCodeList.push_back ("CX");

    ClassList_StringList_T lY_YClassCodeList;
    lY_YClassCodeList.push_back ("Y");

```

```

lY_YClassCodeList.push_back ("Y");
const AirlineClassListKey lSQ_CXAirlineYClassListKey (lSQ_CXAirlineCodeList,
                                                       lY_YClassCodeList);

ClassList_StringList_T lM_MClassCodeList;
lM_MClassCodeList.push_back ("M");
lM_MClassCodeList.push_back ("M");
const AirlineClassListKey lSQ_CXAirlineMClassListKey (lSQ_CXAirlineCodeList,
                                                       lM_MClassCodeList);

// Create the AirlineClassListKey and link it to the FareFeatures object.
AirlineClassList& lSQ_CXAirlineYClassList =
    FacBom<AirlineClassList>::instance().create (lSQ_CXAirlineYClassListKey);
lSQ_CXAirlineYClassList.setFare(1200);
FacBomManager::addToListAndMap (lSINHKGFAreFeatures, lSQ_CXAirlineYClassList)
;
FacBomManager::linkWithParent (lSINHKGFAreFeatures, lSQ_CXAirlineYClassList);

AirlineClassList& lSQ_CXAirlineMClassList =
    FacBom<AirlineClassList>::instance().create (lSQ_CXAirlineMClassListKey);
lSQ_CXAirlineMClassList.setFare(850);
FacBomManager::addToListAndMap (lSINHKGFAreFeatures, lSQ_CXAirlineMClassList)
;
FacBomManager::linkWithParent (lSINHKGFAreFeatures, lSQ_CXAirlineMClassList);

/*=====
=====*/

// Use the same airport pair, and date period for adding SQ SIN-BKK yields.

// Set the point-of-sale-channel primary key.
lPosChannelKey = PosChannelKey(DEFAULT_POS, DEFAULT_CHANNEL);

// Create the PositionKey object and link it to the AirportPair object.
PosChannel& lRAC_SINBKKPosChannel =
    FacBom<PosChannel>::instance().create (lPosChannelKey);
FacBomManager::addToListAndMap (lSINBKKDatePeriod, lRAC_SINBKKPosChannel);
FacBomManager::linkWithParent (lSINBKKDatePeriod, lRAC_SINBKKPosChannel);

// Set the yield time-period primary key.
const TimePeriodKey lYieldTimePeriodKey (lTimeRangeStart,
                                           lTimeRangeEnd);

// Create the TimePeriodKey and link it to the DatePeriod object.
TimePeriod& lSINBKKYieldTimePeriod =
    FacBom<TimePeriod>::instance().create (lYieldTimePeriodKey);
FacBomManager::addToListAndMap (lRAC_SINBKKPosChannel, lSINBKKYieldTimePeriod
);
FacBomManager::linkWithParent (lRAC_SINBKKPosChannel, lSINBKKYieldTimePeriod)
;

// Generate the YieldRule
const YieldFeaturesKey lYieldFeaturesKey (TRIP_TYPE_ONE_WAY,
                                           CABIN_Y);

// Create the YieldFeaturesKey and link it to the TimePeriod object.
YieldFeatures& lSINBKKYieldFeatures =
    FacBom<YieldFeatures>::instance().create (lYieldFeaturesKey);

```

```

FacBomManager::addToListAndMap (lSINBKKYieldTimePeriod, lSINBKKYieldFeatures)
;
FacBomManager::linkWithParent (lSINBKKYieldTimePeriod, lSINBKKYieldFeatures);

// Generate Segment Features and link them to their YieldRule.
// Use the same key as previously.

// Create the AirlineClassListKey and link it to the YieldFeatures object.
AirlineClassList& lRAC_SQAirlineYClassList =
    FacBom<AirlineClassList>::instance().create (lSQAirlineYClassListKey);
lRAC_SQAirlineYClassList.setYield(700);
FacBomManager::addToListAndMap (lSINBKKYieldFeatures, lRAC_SQAirlineYClassList);
FacBomManager::linkWithParent (lSINBKKYieldFeatures, lRAC_SQAirlineYClassList);

AirlineClassList& lRAC_SQAirlineMClassList =
    FacBom<AirlineClassList>::instance().create (lSQAirlineMClassListKey);
lRAC_SQAirlineMClassList.setYield(500);
FacBomManager::addToListAndMap (lSINBKKYieldFeatures, lRAC_SQAirlineMClassList);
FacBomManager::linkWithParent (lSINBKKYieldFeatures, lRAC_SQAirlineMClassList);

/*=====
=====*/

// Use the same airport pair, and date period for adding CX BKK-HKG yields.

// Set the point-of-sale-channel primary key.
// Use the same as previously.

// Create the PositionKey object and link it to the AirportPair object.
PosChannel& lRAC_BKKHKGPosChannel =
    FacBom<PosChannel>::instance().create (lPosChannelKey);
FacBomManager::addToListAndMap (lBKKHKGDatePeriod, lRAC_BKKHKGPosChannel);
FacBomManager::linkWithParent (lBKKHKGDatePeriod, lRAC_BKKHKGPosChannel);

// Set the yield time-period primary key.
// Use the same as previously.

// Create the TimePeriodKey and link it to the DatePeriod object.
TimePeriod& lBKKHKGYieldTimePeriod =
    FacBom<TimePeriod>::instance().create (lYieldTimePeriodKey);
FacBomManager::addToListAndMap (lRAC_BKKHKGPosChannel, lBKKHKGYieldTimePeriod);
FacBomManager::linkWithParent (lRAC_BKKHKGPosChannel, lBKKHKGYieldTimePeriod);

// Generate the YieldRule
// Use the same key as previously.

// Create the YieldFeaturesKey and link it to the TimePeriod object.
YieldFeatures& lBKKHKGYieldFeatures =
    FacBom<YieldFeatures>::instance().create (lYieldFeaturesKey);
FacBomManager::addToListAndMap (lBKKHKGYieldTimePeriod, lBKKHKGYieldFeatures);
FacBomManager::linkWithParent (lBKKHKGYieldTimePeriod, lBKKHKGYieldFeatures);

// Generate Segment Features and link them to their YieldRule.

```

```

// Use the same key as previously.

// Create the AirlineClassListKey and link it to the YieldFeatures object.
AirlineClassList& lRAC_CXAirlineYClassList =
    FacBom<AirlineClassList>::instance().create (lCXAirlineYClassListKey);
lRAC_CXAirlineYClassList.setYield(700);
FacBomManager::addToListAndMap (lBKHKHGYieldFeatures, lRAC_CXAirlineYClassList);
FacBomManager::linkWithParent (lBKHKHGYieldFeatures, lRAC_CXAirlineYClassList);

AirlineClassList& lRAC_CXAirlineMClassList =
    FacBom<AirlineClassList>::instance().create (lCXAirlineMClassListKey);
lRAC_CXAirlineMClassList.setYield(500);
FacBomManager::addToListAndMap (lBKHKHGYieldFeatures, lRAC_CXAirlineMClassList);
FacBomManager::linkWithParent (lBKHKHGYieldFeatures, lRAC_CXAirlineMClassList);

/*=====
=====*/

// Use the same airport pair, and date period for SQ-CX SIN-HKG

// Set the point-of-sale-channel primary key.
// Use the same as previously.

// Create the PositionKey object and link it to the AirportPair object.
PosChannel& lRAC_SINHKGChannel =
    FacBom<PosChannel>::instance().create (lPosChannelKey);
FacBomManager::addToListAndMap (lSINHKGDatePeriod, lRAC_SINHKGChannel);
FacBomManager::linkWithParent (lSINHKGDatePeriod, lRAC_SINHKGChannel);

// Set the yield time-period primary key.
// Use the same as previously.

// Create the TimePeriodKey and link it to the DatePeriod object.
TimePeriod& lSINHKGYieldTimePeriod =
    FacBom<TimePeriod>::instance().create (lYieldTimePeriodKey);
FacBomManager::addToListAndMap (lRAC_SINHKGChannel, lSINHKGYieldTimePeriod);
FacBomManager::linkWithParent (lRAC_SINHKGChannel, lSINHKGYieldTimePeriod);

// Generate the YieldRule
// Use the same key as previously.

// Create the YieldFeaturesKey and link it to the TimePeriod object.
YieldFeatures& lSINHKGYieldFeatures =
    FacBom<YieldFeatures>::instance().create (lYieldFeaturesKey);
FacBomManager::addToListAndMap (lSINHKGYieldTimePeriod, lSINHKGYieldFeatures);
FacBomManager::linkWithParent (lSINHKGYieldTimePeriod, lSINHKGYieldFeatures);

// Generate Segment Features and link them to their YieldRule.
// Use the same key as previously

// Create the AirlineClassListKey and link it to the YieldFeatures object.
AirlineClassList& lRAC_SQ_CXAirlineYClassList =
    FacBom<AirlineClassList>::instance().create (lSQ_CXAirlineYClassListKey);
lRAC_SQ_CXAirlineYClassList.setYield(1200);
FacBomManager::addToListAndMap (lSINHKGYieldFeatures, lRAC_SQ_CXAirlineYClassList);

```

```

        List);
    FacBomManager::linkWithParent (lSINHKGyieldFeatures, lRAC_SQ_CXAirlineYClassL
        ist);

    AirlineClassList& lRAC_SQ_CXAirlineMClassList =
        FacBom<AirlineClassList>::instance().create (lSQ_CXAirlineMClassListKey);
    lRAC_SQ_CXAirlineMClassList.setYield(850);
    FacBomManager::addToListAndMap (lSINHKGyieldFeatures, lRAC_SQ_CXAirlineMClass
        List);
    FacBomManager::linkWithParent (lSINHKGyieldFeatures, lRAC_SQ_CXAirlineMClassL
        ist);

    }

}

/*!

```

4 C++ Class Storing the StdAir Context

```

    */
    // //////////////////////////////////////
    // Import section
    // //////////////////////////////////////
    // STL
    #include <cassert>
    #include <sstream>
    // StdAir
    #include <stdair/basic/BasConst_General.hpp>
    #include <stdair/bom/BomRoot.hpp>
    #include <stdair/bom/EventQueue.hpp>
    #include <stdair/factory/FacBom.hpp>
    #include <stdair/service/STDAIR_ServiceContext.hpp>

    namespace stdair {

        // //////////////////////////////////////
        STDAIR_ServiceContext::STDAIR_ServiceContext ()
            : _bomRoot (NULL), _eventQueue (NULL),
              _initType (ServiceInitialisationType::NOT_YET_INITIALISED) {
            // Build the BomRoot object
            init();
        }

        // //////////////////////////////////////
        STDAIR_ServiceContext::
        STDAIR_ServiceContext (const STDAIR_ServiceContext& iServiceContext)
            : _bomRoot (iServiceContext._bomRoot),
              _eventQueue (iServiceContext._eventQueue),
              _initType (ServiceInitialisationType::NOT_YET_INITIALISED) {
            assert (false);
        }

        // //////////////////////////////////////
        STDAIR_ServiceContext::~STDAIR_ServiceContext () {
        }

        // //////////////////////////////////////
        void STDAIR_ServiceContext::init () {
            //

```

```

        initBomRoot();

        //
        initEventQueue();
    }

    // //////////////////////////////////////
    void STDAIR_ServiceContext::initBomRoot() {
        _bomRoot = &FacBom<BomRoot>::instance().create();
    }

    // //////////////////////////////////////
    void STDAIR_ServiceContext::initEventQueue() {

        // The event queue key is just a string. For now, it is not used.
        const EventQueueKey lKey ("EQ01");

        // Create an EventQueue object instance
        EventQueue& lEventQueue = FacBom<EventQueue>::instance().create (lKey);

        // Store the event queue object
        _eventQueue = &lEventQueue;
    }

    // //////////////////////////////////////
    const std::string STDAIR_ServiceContext::shortDisplay() const {
        std::ostringstream oStr;
        oStr << "STDAIR_ServiceContext -- " << _initType
            << " -- DB: " << _dbParams;
        if (_eventQueue != NULL) {
            oStr << " -- Queue: " << _eventQueue->toString();
        }
        return oStr.str();
    }

    // //////////////////////////////////////
    const std::string STDAIR_ServiceContext::display() const {
        std::ostringstream oStr;
        oStr << shortDisplay();
        return oStr.str();
    }

    // //////////////////////////////////////
    const std::string STDAIR_ServiceContext::describe() const {
        return shortDisplay();
    }

    // //////////////////////////////////////
    BomRoot& STDAIR_ServiceContext::getBomRoot() const {
        assert (_bomRoot != NULL);
        return *_bomRoot;
    }

    // //////////////////////////////////////
    EventQueue& STDAIR_ServiceContext::getEventQueue() const {
        assert (_eventQueue != NULL);
        return *_eventQueue;
    }
}

/*!

```

5 People

5.1 Project Admins (and Developers)

- Denis Arnaud <denis_arnaud@users.sourceforge.net> (N)
- Anh Quan Nguyen <quannaus@users.sourceforge.net> (N)
- Gabrielle Sabatier <gsabatier@users.sourceforge.net> (N)

5.2 Retired Developers

- Mehdi Ayouni <mehdi.ayouni@gmail.com>
- Son Nguyen Kim <snguyenkim@users.sourceforge.net> (N)

5.3 Contributors

- Emmanuel Bastien <ebastien@users.sourceforge.net> (N)

5.4 Distribution Maintainers

- **Fedora/RedHat**: Denis Arnaud <denis_arnaud@users.sourceforge.net> (N)
- **Debian**: Emmanuel Bastien <ebastien@users.sourceforge.net> (N)

Note

(N) - **Amadeus** employees.

6 Coding Rules

In the following sections we describe the naming conventions which are used for files, classes, structures, local variables, and global variables.

6.1 Default Naming Rules for Variables

Variables names follow Java naming conventions. Examples:

- `lNumberOfPassengers`
- `lSeatAvailability`

6.2 Default Naming Rules for Functions

Function names follow Java naming conventions. Example:

- `int myFunctionName (const int& a, int b)`

6.3 Default Naming Rules for Classes and Structures

Each new word in a class or structure name should always start with a capital letter and the words should be separated with an under-score. Abbreviations are written with capital letters. Examples:

- `MyClassName`
- `MyStructName`

6.4 Default Naming Rules for Files

Files are named after the C++ class names.

Source files are named using `.cpp` suffix, whereas header files end with `.hpp` extension. Examples:

- `FlightDate.hpp`
- `SegmentDate.cpp`

6.5 Default Functionality of Classes

All classes that are configured by input parameters should include:

- default empty constructor
- one or more additional constructor(s) that takes input parameters and initializes the class instance
- setup function, preferably named `'setup'` or `'set_parameters'`

Explicit destructor functions are not required, unless they are needed. It shall not be possible to use any of the other member functions unless the class has been properly initiated with the input parameters.

7 Copyright and License

7.1 GNU LESSER GENERAL PUBLIC LICENSE

7.1.1 Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc.
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL. It also counts
as the successor of the GNU Library Public License, version 2, hence
the version number 2.1.]

7.2 Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that

any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

7.3 TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and

data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

- a) The modified work must itself be a software library.
- b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.
- c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.
- d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent

and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

- a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)
- b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.
- c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.
- d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.
- e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

7.3.1 NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LI-

ABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7.3.2 END OF TERMS AND CONDITIONS

7.4 How to Apply These Terms to Your New Programs

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the library's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>
```

```
This library is free software; you can redistribute it and/or
modify it under the terms of the GNU Lesser General Public
License as published by the Free Software Foundation; either
version 2.1 of the License, or (at your option) any later version.
```

```
This library is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
Lesser General Public License for more details.
```

```
You should have received a copy of the GNU Lesser General Public
License along with this library; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
```

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the
library 'Frob' (a library for tweaking knobs) written by James Random Hacker.
```

```
<signature of Ty Coon>, 1 April 1990
Ty Coon, President of Vice
```

That's all there is to it!

Source

8 Documentation Rules

8.1 General Rules

All classes in StdAir should be properly documented with Doxygen comments in include (.hpp) files. Source (.cpp) files should be documented according to a normal standard for well documented C++ code.

An example of how the interface of a class shall be documented in StdAir is shown here:

```

/*!
 * \brief Brief description of MyClass here
 *
 * Detailed description of MyClass here. With example code if needed.
 */
class MyClass {
public:
    ///! Default constructor
    MyClass(void) { setup_done = false; }

    /*!
     * \brief Constructor that initializes the class with parameters
     *
     * Detailed description of the constructor here if needed
     *
     * \param[in] param1 Description of \a param1 here
     * \param[in] param2 Description of \a param2 here
     */
    MyClass(TYPE1 param1, TYPE2 param2) { setup(param1, param2); }

    /*!
     * \brief Setup function for MyClass
     *
     * Detailed description of the setup function here if needed
     *
     * \param[in] param1 Description of \a param1 here
     * \param[in] param2 Description of \a param2 here
     */
    void setup(TYPE1 param1, TYPE2 param2);

    /*!
     * \brief Brief description of memberFunction1
     *
     * Detailed description of memberFunction1 here if needed
     *
     * \param[in] param1 Description of \a param1 here
     * \param[in] param2 Description of \a param2 here
     * \param[in,out] param3 Description of \a param3 here
     * \return Description of the return value here
     */
    TYPE4 memberFunction1(TYPE1 param1, TYPE2 param2, TYPE3 &param3);

private:
    bool _setupDone;          /*!< Variable that checks if the class is properly
                               initialized with parameters */
    TYPE1 _privateVariable1; /*!< Short description of _privateVariable1 here
    TYPE2 _privateVariable2; /*!< Short description of _privateVariable2 here
};

```

8.2 File Header

All files should start with the following header, which include Doxygen's `\file`, `\brief` and `\author` tags, `$Date$` and `$Revisions$` CVS tags, and a common copyright note:

```

/*!
 * \file
 * \brief Brief description of the file here
 * \author Names of the authors who contributed to this code
 * \date Date
 *
 * Detailed description of the file here if needed.
 *
 * -----
 *
 * StdAir - C++ Standard Airline IT Object Library
 *
 * Copyright (C) 2009-2010 (\see authors file for a list of contributors)
 *
 * \see copyright file for license information
 *
 * -----
 */

```

8.3 Grouping Various Parts

All functions must be added to a Doxygen group in order to appear in the documentation. The following code example defines the group `'my_group'`:

```

/*!
 * \defgroup my_group Brief description of the group here
 *
 * Detailed description of the group here
 */

```

The following example shows how to document the function `myFunction` and how to add it to the group `my_group`:

```

/*!
 * \brief Brief description of myFunction here
 * \ingroup my_group
 *
 * Detailed description of myFunction here
 *
 * \param[in] param1 Description of \a param1 here
 * \param[in] param2 Description of \a param2 here
 * \return Description of the return value here
 */
TYPE3 myFunction(TYPE1 param1, TYPE2 &param2);

```

9 Main features

A short list of the main features of StdAir is given below sorted in different categories. Many more features and functions exist and for these we refer to the reference documentation.

9.1 Standard Airline IT Business Object Model (BOM)

- (Airline) Network-related classes:
 - Network, ReachableUniverse
- (Air) Travel-related classes:
 - TravelSolution, OriginDestination,
- (Airline) Inventory-related classes:
 - Inventory, FlightDate, SegmentDate, SegmentCabin, BookingClass, LegDate, LegCabin, Bucket
- (Airline) Schedule-related classes:
 - FlightPeriod, SegmentPeriod, LegPeriod
- (Simulated) Passenger-related demand classes:
 - DemandStream, BookingRequest
- (Air) Price-related classes:
 - YieldStore

9.2 Architecture of the StdAir library

- Separate structure and content classes
- `Boost.Fusion`

10 Make a Difference

Do not ask what StdAir can do for you. Ask what you can do for StdAir.

You can help us to develop the StdAir library. There are always a lot of things you can do:

- Start using StdAir
- Tell your friends about StdAir and help them to get started using it

- If you find a bug, report it to us (on the [dedicated Sourceforge's Trac Web site](#)). Without your help we can never hope to produce a bug free code.
- Help us to improve the documentation by providing information about documentation bugs
- Answer support requests in the StdAir [discussion forums](#) on SourceForge. If you know the answer to a question, help others to overcome their StdAir problems.
- Help us to improve our algorithms. If you know of a better way (e.g., that is faster or requires less memory) to implement some of our algorithms, then let us know.
- Help to port StdAir to new platforms. If you manage to compile StdAir on a new platform, then tell how you did it.
- Send your code. If you have a good StdAir compatible code, which you can release under the LGPL, and you think it should be included in StdAir, then send it to the community.
- Become an StdAir developer. Send us (see the [People](#) page) an e-mail and tell what you can do for StdAir.

11 Make a new release

11.1 Introduction

This document describes briefly the recommended procedure of releasing a new version of StdAir using a Linux development machine and the SourceForge project site.

The following steps are required to make a release of the distribution package.

11.2 Initialisation

Clone locally the full [Git project](#):

```
cd ~
mkdir -p dev/sim
cd ~/dev/sim
git clone git://stdair.git.sourceforge.net/gitroot/stdair/stdair stdairgit
cd stdairgit
git checkout trunk
```

11.3 Branch creation

Create the branch, on your local clone, corresponding to the new release (say, 0.5.0):

```
cd ~/dev/sim/stdairgit
git checkout trunk
git checkout -b 0.5.0
```

Update the version in the various build system files, replacing 99.99.99 by the correct version number:

```
vi CMakeLists.txt
vi autogen.sh
```

Update the version and add a change-log in the ChangeLog and in the RPM specification files:

```
vi ChangeLog
vi stdair.spec
```

11.4 Commit and publish the release branch

Commit the new release:

```
cd ~/dev/sim/stdairgit
git add -A
git commit -m "[Release 0.5.0] Release of version 0.5.0."
git push
```

11.5 Update the change-log in the trunk as well

Update the change-log in the ChangeLog and RPM specification files:

```
cd ~/dev/sim/stdairgit
git checkout trunk
vi ChangeLog
vi stdair.spec
```

Commit the change-logs and publish the trunk (main development branch):

```
git commit -m "[Doc] Integrated the change-log of the release 0.5.0."
git push
```

11.6 Create distribution packages

Create the distribution packages using the following command:

```
cd ~/dev/sim/stdairgit
git checkout 0.5.0
rm -rf build && mkdir -p build
cd build
cmake -DCMAKE_INSTALL_PREFIX=/home/user/dev/deliveries/stdair-0.5.0 \
      -DCMAKE_BUILD_TYPE:String=Debug -DINSTALL_DOC:BOOL=ON ..
make check && make dist
```

This will configure, compile and check the package. The output packages will be named, for instance, `stdair-0.5.0.tar.gz` and `stdair-0.5.0.tar.bz2`.

11.7 Generation the RPM packages

Optionally, generate the RPM package (for instance, for [Fedora/RedHat](#)):

```
cd ~/dev/sim/stdairgit
git checkout 0.5.0
rm -rf build && mkdir -p build
cd build
cmake -DCMAKE_INSTALL_PREFIX=/home/user/dev/deliveries/stdair-99.99.99 \
      -DCMAKE_BUILD_TYPE:String=Debug -DINSTALL_DOC:BOOL=ON ..
make dist
```

To perform this step, rpm-build, rpmlint and rpmdevtools have to be available on the system.

```
cp stdair.spec ~/dev/packages/SPECS \
  && cp stdair-0.5.0.tar.bz2 ~/dev/packages/SOURCES
cd ~/dev/packages/SPECS
rpmbuild -ba stdair.spec
rpmlint -i ../SPECS/stdair.spec ../SRPMS/stdair-0.5.0-1.fc15.src.rpm \
  ../RPMS/noarch/stdair-* ../RPMS/i686/stdair-*
```

11.8 Update distributed change log

Update the NEWS and ChangeLog files with appropriate information, including what has changed since the previous release. Then commit and push the changes into the [StdAir's Git repository](#).

11.9 Create the binary package, including the documentation

Create the binary package, which includes HTML and PDF documentation, using the following command:

```
make package
```

The output binary package will be named, for instance, stdair-0.5.0-Linux.tar.bz2. That package contains both the HTML and PDF documentation. The binary package contains also the executables and shared libraries, as well as C++ header files, but all of those do not interest us for now.

11.10 Upload the files to SourceForge

Upload the distribution and documentation packages to the SourceForge server. Check [SourceForge help page on uploading software](#).

11.11 Upload the documentation to SourceForge

In order to update the Web site files, either:

- [synchronise them with rsync and SSH](#):

```
cd ~/dev/sim/stdairgit
git checkout 0.5.0
rsync -aiv doc/html/ doc/latex/refman.pdf joe,stdair@web.sourceforge.net:htdocs/
```

where `-aiv` options mean:

- `-a`: archive/mirror mode; equals `-rlptgoD` (no `-H`, `-A`, `-X`)
- `-v`: increase verbosity
- `-i`: output a change-summary for all updates
- Note the trailing slashes (/) at the end of both the source and target directories. It means that the content of the source directory (`doc/html`), rather than the directory itself, has to be copied into the content of the target directory.

- or use the [SourceForge Shell service](#).

11.12 Make a new post

- submit a new entry in the [SourceForge project-related news feed](#)
- make a new post on the [SourceForge hosted WordPress blog](#)
- and update, if necessary, [Trac tickets](#).

11.13 Send an email on the announcement mailing-list

Finally, you should send an announcement to stdair-announce@lists.sourceforge.net (see <https://lists.sourceforge.net/lists/listinfo/stdair-announce> for the archives)

12 Installation

12.1 Table of Contents

- [Fedora/RedHat Linux distributions](#)
- [StdAir Requirements](#)
- [Basic Installation](#)
- [Compilers and Options](#)
- [Compiling For Multiple Architectures](#)
- [Installation Names](#)
- [Optional Features](#)
- [Particular systems](#)

- [Specifying the System Type](#)
- [Sharing Defaults](#)
- [Defining Variables](#)
- ['cmake' Invocation](#)

12.2 Fedora/RedHat Linux distributions

Note that on [Fedora/RedHat](#) Linux distributions, RPM packages are available and can be installed with your usual package manager. For instance:

```
yum -y install stdair-devel stdair-doc
```

RPM packages can also be available on the [SourceForge download site](#).

12.3 StdAir Requirements

StdAir should compile without errors or warnings on most GNU/Linux systems, on UNIX systems like Solaris SunOS, and on POSIX based environments for Microsoft Windows like Cygwin or MinGW with MSYS. It can be also built on Microsoft Windows NT/2000/XP/Vista/7 using Microsoft's Visual C++ .NET, but our support for this compiler is limited. For GNU/Linux, SunOS, Cygwin and MinGW we assume that you have at least the following GNU software installed on your computer:

- GNU Autotools:
 - [autoconf](#),
 - [automake](#),
 - [libtool](#),
 - [make](#), version 3.72.1 or later (check version with ``make --version``)
- [GCC](#) - GNU C++ Compiler (g++), version 4.3.x or later (check version with ``gcc --version``)
- [Boost](#) - C++ STL extensions, version 1.35 or later (check version with ``grep "define BOOST_LIB_VERSION" /usr/include/boost/version.hpp``)
- [MySQL](#) - Database client libraries, version 5.0 or later (check version with ``mysql --version``)
- [SOI](#) - C++ database client library wrapper, version 3.0.0 or later (check version with ``soci-config --version``)

Optionally, you might need a few additional programs: [Doxygen](#), [LaTeX](#), [Dvips](#) and [Ghostscript](#), to generate the HTML and PDF documentation.

We strongly recommend that you use recent stable releases of the GCC, if possible. We do not actively work on supporting older versions of the GCC, and they may therefore (without prior notice) become unsupported in future releases of StdAir.

12.4 Basic Installation

Briefly, the shell commands `./cmake .. && make install` should configure, build and install this package. The following more-detailed instructions are generic; see the `README` file for instructions specific to this package. Some packages provide this `INSTALL` file but do not implement all of the features documented below. The lack of an optional feature in a given package is not necessarily a bug. More recommendations for GNU packages can be found in the info page corresponding to "Makefile Conventions: (standards)Makefile Conventions".

The `cmake` shell script attempts to guess correct values for various system-dependent variables used during compilation. It uses those values to create a `Makefile` in each directory of the package. It may also create one or more `.h` files containing system-dependent definitions. Finally, it creates a `CMakeCache.txt` cache file that you can refer to in the future to recreate the current configuration, and files `CMakeFiles` containing compiler output (useful mainly for debugging `cmake`).

It can also use an optional file (typically called `config.cache` and enabled with `--cache-file=config.cache` or simply `-C`) that saves the results of its tests to speed up reconfiguring. Caching is disabled by default to prevent problems with accidental use of stale cache files.

If you need to do unusual things to compile the package, please try to figure out how `configure` could check whether to do them, and mail diffs or instructions to the address given in the `README` so they can be considered for the next release. If you are using the cache, and at some point `config.cache` contains results you don't want to keep, you may remove or edit it.

The file `CMakeLists.txt` is used to create the `Makefile` files.

The simplest way to compile this package is:

1. `cd` to the directory containing the package's source code and type `./cmake ..` to configure the package for your system. Running `cmake` is generally fast. While running, it prints some messages telling which features it is checking for.
2. Type `make` to compile the package.
3. Optionally, type `make check` to run any self-tests that come with the package, generally using the just-built uninstalled binaries.
4. Type `make install` to install the programs and any data files and documentation. When installing into a prefix owned by root, it is recommended that the package be configured and built as a regular user, and only the `make install` phase executed with root privileges.
5. You can remove the program binaries and object files from the source code directory by typing `make clean`. To also remove the files that `configure` created (so you can compile the package for a different kind of computer), type `make distclean`. There is also a `make maintainer-clean` target,

but that is intended mainly for the package's developers. If you use it, you may have to get all sorts of other programs in order to regenerate files that came with the distribution.

6. Often, you can also type 'make uninstall' to remove the installed files again. In practice, not all packages have tested that uninstallation works correctly, even though it is required by the GNU Coding Standards.

12.5 Compilers and Options

Some systems require unusual options for compilation or linking that the 'cmake' script does not know about. Run './cmake --help' for details on some of the pertinent environment variables.

You can give 'cmake' initial values for configuration parameters by setting variables in the command line or in the environment. Here is an example:

```
./cmake CC=c99 CFLAGS=-g LIBS=-lposix
```

See also

[Defining Variables](#) for more details.

12.6 Compiling For Multiple Architectures

You can compile the package for more than one kind of computer at the same time, by placing the object files for each architecture in their own directory. To do this, you can use GNU 'make'. 'cd' to the directory where you want the object files and executables to go and run the 'configure' script. 'configure' automatically checks for the source code in the directory that 'configure' is in and in '..'. This is known as a "VPATH" build.

With a non-GNU 'make', it is safer to compile the package for one architecture at a time in the source code directory. After you have installed the package for one architecture, use 'make distclean' before reconfiguring for another architecture.

On MacOS X 10.5 and later systems, you can create libraries and executables that work on multiple system types--known as "fat" or "universal" binaries--by specifying multiple '-arch' options to the compiler but only a single '-arch' option to the preprocessor. Like this:

```
./configure CC="gcc -arch i386 -arch x86_64 -arch ppc -arch ppc64" \  
CXX="g++ -arch i386 -arch x86_64 -arch ppc -arch ppc64" \  
CPP="gcc -E" CXXCPP="g++ -E"
```

This is not guaranteed to produce working output in all cases, you may have to build one architecture at a time and combine the results using the 'lipo' tool if you have problems.

12.7 Installation Names

By default, 'make install' installs the package's commands under '/usr/local/bin', include files under '/usr/local/include', etc. You can specify an installation prefix other than '/usr/local' by giving 'configure' the option '--prefix=PREFIX', where PREFIX must be an absolute file name.

You can specify separate installation prefixes for architecture-specific files and architecture-independent files. If you pass the option '--exec-prefix=PREFIX' to 'configure', the package uses PREFIX as the prefix for installing programs and libraries. Documentation and other data files still use the regular prefix.

In addition, if you use an unusual directory layout you can give options like '--bindir=DIR' to specify different values for particular kinds of files. Run 'configure --help' for a list of the directories you can set and what kinds of files go in them. In general, the default for these options is expressed in terms of '\${prefix}', so that specifying just '--prefix' will affect all of the other directory specifications that were not explicitly provided.

The most portable way to affect installation locations is to pass the correct locations to 'configure'; however, many packages provide one or both of the following shortcuts of passing variable assignments to the 'make install' command line to change installation locations without having to reconfigure or recompile.

The first method involves providing an override variable for each affected directory. For example, 'make install prefix=/alternate/directory' will choose an alternate location for all directory configuration variables that were expressed in terms of '\${prefix}'. Any directories that were specified during 'configure', but not in terms of '\${prefix}', must each be overridden at install time for the entire installation to be relocated. The approach of makefile variable overrides for each directory variable is required by the GNU Coding Standards, and ideally causes no recompilation. However, some platforms have known limitations with the semantics of shared libraries that end up requiring recompilation when using this method, particularly noticeable in packages that use GNU Libtool.

The second method involves providing the 'DESTDIR' variable. For example, 'make install DESTDIR=/alternate/directory' will prepend '/alternate/directory' before all installation names. The approach of 'DESTDIR' overrides is not required by the

GNU Coding Standards, and does not work on platforms that have drive letters. On the other hand, it does better at avoiding recompilation issues, and works well even when some directory options were not specified in terms of `'${prefix}'` at `'configure'` time.

12.8 Optional Features

If the package supports it, you can cause programs to be installed with an extra prefix or suffix on their names by giving `'cmake'` the option `'--program-prefix=PREFIX'` or `'--program-suffix=SUFFIX'`.

Some packages pay attention to `'--enable-FEATURE'` options to `'configure'`, where `FEATURE` indicates an optional part of the package. They may also pay attention to `'--with-PACKAGE'` options, where `PACKAGE` is something like `'gnu-as'` or `'x'` (for the X Window System). The `'README'` should mention any `'--enable-'` and `'--with-'` options that the package recognizes.

For packages that use the X Window System, `'configure'` can usually find the X include and library files automatically, but if it doesn't, you can use the `'configure'` options `'--x-includes=DIR'` and `'--x-libraries=DIR'` to specify their locations.

Some packages offer the ability to configure how verbose the execution of `'make'` will be. For these packages, running `'./configure --enable-silent-rules'` sets the default to minimal output, which can be overridden with `'make V=1'`; while running `'./configure --disable-silent-rules'` sets the default to verbose, which can be overridden with `'make V=0'`.

12.9 Particular systems

On HP-UX, the default C compiler is not ANSI C compatible. If GNU CC is not installed, it is recommended to use the following options in order to use an ANSI C compiler:

```
./configure CC="cc -Ae -D_XOPEN_SOURCE=500"
```

and if that doesn't work, install pre-built binaries of GCC for HP-UX.

On OSF/1 a.k.a. Tru64, some versions of the default C compiler cannot parse its `'<wchar.h>'` header file. The option `'-nodtk'` can be used as a workaround. If GNU CC is not installed, it is therefore recommended to try

```
./configure CC="cc"
```

and if that doesn't work, try

```
./configure CC="cc -nodtk"
```

On Solaris, don't put `/usr/ucb` early in your `PATH`. This directory contains several dysfunctional programs; working variants of these programs are available in `/usr/bin`. So, if you need `/usr/ucb` in your `PATH`, put it *after* `/usr/bin`.

On Haiku, software installed for all users goes in `/boot/common`, not `/usr/local`. It is recommended to use the following options:

```
./cmake -DCMAKE_INSTALL_PREFIX=/boot/common
```

12.10 Specifying the System Type

There may be some features `configure` cannot figure out automatically, but needs to determine by the type of machine the package will run on. Usually, assuming the package is built to be run on the *same* architectures, `configure` can figure that out, but if it prints a message saying it cannot guess the machine type, give it the `--build=TYPE` option. TYPE can either be a short name for the system type, such as `sun4`, or a canonical name which has the form CPU-COMPANY-SYSTEM

where SYSTEM can have one of these forms:

- OS
- KERNEL-OS

See the file `config.sub` for the possible values of each field. If `config.sub` isn't included in this package, then this package doesn't need to know the machine type.

If you are *building* compiler tools for cross-compiling, you should use the option `--target=TYPE` to select the type of system they will produce code for.

If you want to *use* a cross compiler, that generates code for a platform different from the build platform, you should specify the "host" platform (i.e., that on which the generated programs will eventually be run) with `--host=TYPE`.

12.11 Sharing Defaults

If you want to set default values for `configure` scripts to share, you can create a site shell script called `config.site` that gives default values for variables like `CC`, `cache_file`, and `prefix`. `configure` looks for `PREFIX/share/config.site` if it exists, then `PREFIX/etc/config.site` if it exists. Or, you can set the `CONFIG_SITE` environment variable to the location

of the site script. A warning: not all 'configure' scripts look for a site script.

12.12 Defining Variables

Variables not defined in a site shell script can be set in the environment passed to 'configure'. However, some packages may run configure again during the build, and the customized values of these variables may be lost. In order to avoid this problem, you should set them in the 'configure' command line, using 'VAR=value'. For example:

```
./configure CC=/usr/local2/bin/gcc
```

causes the specified 'gcc' to be used as the C compiler (unless it is overridden in the site shell script).

Unfortunately, this technique does not work for 'CONFIG_SHELL' due to an Autoconf bug. Until the bug is fixed you can use this workaround:

```
CONFIG_SHELL=/bin/bash /bin/bash ./configure CONFIG_SHELL=/bin/bash
```

12.13 'cmake' Invocation

'cmake' recognizes the following options to control how it operates.

- '--help', '-h' print a summary of all of the options to 'configure', and exit.
- '--help=short', '--help=recursive' print a summary of the options unique to this package's 'configure', and exit. The 'short' variant lists options used only in the top level, while the 'recursive' variant lists options also present in any nested packages.
- '--version', '-V' print the version of Autoconf used to generate the 'configure' script, and exit.
- '--cache-file=FILE' enable the cache: use and save the results of the tests in FILE, traditionally 'config.cache'. FILE defaults to '/dev/null' to disable caching.
- '--config-cache', '-C' alias for '--cache-file=config.cache'.
- '--quiet', '--silent', '-q' do not print messages saying which checks are being made. To suppress all normal output, redirect it to '/dev/null' (any error messages will still be shown).
- '--srcdir=DIR' look for the package's source code in directory DIR. Usually 'configure' can determine that directory automatically.

- '--prefix=DIR' use DIR as the installation prefix.

See also

[Installation Names](#) for more details, including other options available for fine-tuning the installation locations.

- '--no-create', '-n' run the configure checks, but stop before creating any output files.

'cmake' also accepts some other, not widely useful, options. Run 'cmake --help' for more details.

The 'cmake' script produces an output like this:

```
cmake -DCMAKE_INSTALL_PREFIX=/home/user/dev/deliveries/stdair-0.50.0 \
-DLIB_SUFFIX=64 -DCMAKE_BUILD_TYPE:String=Debug -DINSTALL_DOC:BOOL=ON ..
-- The C compiler identification is GNU
-- The CXX compiler identification is GNU
-- Check for working C compiler: /usr/lib64/ccache/gcc
-- Check for working C compiler: /usr/lib64/ccache/gcc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: /usr/lib64/ccache/c++
-- Check for working CXX compiler: /usr/lib64/ccache/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Requires Git without specifying any version
-- Current Git revision name: e8beb4d11ff9blaf6b3f3e9ff1e92250aee0291a trunk
-- Requires PythonLibs-2.7
-- Found PythonLibs: /usr/lib64/libpython2.7.so (Required is at least version "2.7")
-- Found PythonLibs 2.7
-- Requires Boost-1.41
-- Boost version: 1.46.0
-- Found the following Boost libraries:
--   program_options
--   date_time
--   iostreams
--   serialization
--   filesystem
--   unit_test_framework
--   python
-- Found Boost version: 1.46.0
-- Found BoostWrapper: /usr/include (Required is at least version "1.41")
-- Requires ZeroMQ-2.0
-- Found ZeroMQ: /usr/lib64/libzmq.so (Required is at least version "2.0")
-- Found ZeroMQ version: 2.1
-- Requires MySQL-5.1
-- Using mysql-config: /usr/bin/mysql_config
-- Found MySQL: /usr/lib64/mysql/libmysqlclient.so (Required is at least version "5.1")
-- Found MySQL version: 5.5.14
-- Requires SOCI-3.0
-- Using soci-config: /usr/bin/soci-config
-- SOCI headers are buried
-- Found SOCI: /usr/lib64/libsoci_core.so (Required is at least version "3.0")
-- Found SOCIMySQL: /usr/lib64/libsoci_mysql.so (Required is at least version "3.0")
-- Found SOCI with MySQL back-end support version: 3.0.0
-- Requires Doxygen-1.7
-- Found Doxygen: /usr/bin/doxygen
-- Found DoxygenWrapper: /usr/bin/doxygen (Required is at least version "1.7")
```

```

-- Found Doxygen version: 1.7.4
-- Had to set the linker language for 'stdairlib' to CXX
-- Had to set the linker language for 'stdairuicllib' to CXX
-- Test 'StdAirTest' to be built with 'MPBomRoot.cpp;MPInventory.cpp;StandardAirlineITTestSuite.cpp'
--
-- =====
-- -----
-- ---      Project Information      ---
-- -----
-- PROJECT_NAME ..... : stdair
-- PACKAGE_PRETTY_NAME ..... : StdAir
-- PACKAGE ..... : stdair
-- PACKAGE_NAME ..... : STDAIR
-- PACKAGE_VERSION ..... : 0.50.0
-- GENERIC_LIB_VERSION ..... : 0.50.0
-- GENERIC_LIB_SOVERSION ..... : 99.99
--
-- -----
-- ---      Build Configuration      ---
-- -----
-- Modules to build ..... : stdair
-- Libraries to build/install ..... : stdairlib;stdairuicllib
-- Binaries to build/install ..... : stdair
-- Modules to test ..... : stdair
-- Binaries to test ..... : StdAirTesttst
--
-- * Module ..... : stdair
--   + Layers to build ..... : .;basic;bom;factory;dbadaptor;command;service
--   + Dependencies on other layers :
--   + Libraries to build/install . : stdairlib;stdairuicllib
--   + Executables to build/install : stdair
--   + Tests to perform ..... : StdAirTesttst
--
-- BUILD_SHARED_LIBS ..... : ON
-- CMAKE_BUILD_TYPE ..... : Debug
-- * CMAKE_C_FLAGS ..... :
-- * CMAKE_CXX_FLAGS ..... : -O2 -g -pipe -Wall -Wp,-D_FORTIFY_SOURCE=2 -fexceptions -fst
-- * BUILD_FLAGS ..... :
-- * COMPILE_FLAGS ..... :
-- CMAKE_MODULE_PATH ..... : /home/user/dev/sim/stdair/stdairgithub/config/
-- CMAKE_INSTALL_PREFIX ..... : /home/user/dev/deliveries/stdair-0.50.0
--
-- * Doxygen:
--   - DOXYGEN_VERSION ..... : 1.7.4
--   - DOXYGEN_EXECUTABLE ..... : /usr/bin/doxygen
--   - DOXYGEN_DOT_EXECUTABLE ..... : /usr/bin/dot
--   - DOXYGEN_DOT_PATH ..... : /usr/bin
--
-- -----
-- ---      Installation Configuration      ---
-- -----
-- INSTALL_LIB_DIR ..... : /home/user/dev/deliveries/stdair-0.50.0/lib64
-- INSTALL_BIN_DIR ..... : /home/user/dev/deliveries/stdair-0.50.0/bin
-- INSTALL_INCLUDE_DIR ..... : /home/user/dev/deliveries/stdair-0.50.0/include
-- INSTALL_DATA_DIR ..... : /home/user/dev/deliveries/stdair-0.50.0/share
-- INSTALL_SAMPLE_DIR ..... : /home/user/dev/deliveries/stdair-0.50.0/share/stdair/samples
-- INSTALL_DOC ..... : ON
--
-- -----
-- ---      Packaging Configuration      ---
-- -----
-- CPACK_PACKAGE_CONTACT ..... : Denis Arnaud <denis_arnaud - at - users dot sourceforge dot n

```

```

-- CPACK_PACKAGE_VENDOR ..... : Denis Arnaud
-- CPACK_PACKAGE_VERSION ..... : 0.50.0
-- CPACK_PACKAGE_DESCRIPTION_FILE . : /home/user/dev/sim/stdair/stdairgithub/README
-- CPACK_RESOURCE_FILE_LICENSE .... : /home/user/dev/sim/stdair/stdairgithub/COPYING
-- CPACK_GENERATOR ..... : TBZ2
-- CPACK_DEBIAN_PACKAGE_DEPENDS ... :
-- CPACK_SOURCE_GENERATOR ..... : TBZ2;TGZ
-- CPACK_SOURCE_PACKAGE_FILE_NAME . : stdair-0.50.0
--
-- -----
-- ---      External libraries      ---
-- -----
--
-- * Python:
--   - PYTHONLIBS_VERSION ..... : 2.7
--   - PYTHON_LIBRARIES ..... : /usr/lib64/libpython2.7.so
--   - PYTHON_INCLUDE_PATH ..... : /usr/include/python2.7
--   - PYTHON_INCLUDE_DIRS ..... : /usr/include/python2.7
--   - PYTHON_DEBUG_LIBRARIES ..... :
--   - Python_ADDITIONAL_VERSIONS . :
--
-- * ZeroMQ:
--   - ZeroMQ_VERSION ..... : 2.1
--   - ZeroMQ_LIBRARIES ..... : /usr/lib64/libzmq.so
--   - ZeroMQ_INCLUDE_DIR ..... : /usr/include
--
-- * Boost:
--   - Boost_VERSION ..... : 104600
--   - Boost_LIB_VERSION ..... : 1_46
--   - Boost_HUMAN_VERSION ..... : 1.46.0
--   - Boost_INCLUDE_DIRS ..... : /usr/include
--   - Boost required components .. : program_options;date_time;iostreams;serialization;filesystem;
--   - Boost required libraries ... : optimized;/usr/lib64/libboost_iostreams-mt.so;debug;/usr/lib64/libboost_iostreams-mt.so;
--
-- * MySQL:
--   - MYSQL_VERSION ..... : 5.5.14
--   - MYSQL_INCLUDE_DIR ..... : /usr/include/mysql
--   - MYSQL_LIBRARIES ..... : /usr/lib64/mysql/libmysqlclient.so
--
-- * SOCI:
--   - SOCI_VERSION ..... : 3.0.0
--   - SOCI_INCLUDE_DIR ..... : /usr/include/soci
--   - SOCI_MYSQL_INCLUDE_DIR ..... : /usr/include/soci
--   - SOCI_LIBRARIES ..... : /usr/lib64/libsoci_core.so
--   - SOCI_MYSQL_LIBRARIES ..... : /usr/lib64/libsoci_mysql.so
--
-- Change a value with: cmake -D<Variable>=<Value>
-- =====
--
-- Configuring done
-- Generating done
-- Build files have been written to: /home/user/dev/sim/stdair/stdairgithub/build

```

It is recommended that you check if your library has been compiled and linked properly and works as expected. To do so, you should execute the testing process 'make check'. As a result, you should obtain a similar report:

```

[ 0%] Built target hdr_cfg_stdair
[ 97%] Built target stdairlib
[100%] Built target StdAirTesttst

```

```
Scanning dependencies of target check_stdairtst
Test project /home/user/dev/sim/stdair/stdairgithub/build/test/stdair
  Start 1: StdAirTesttst
1/1 Test #1: StdAirTesttst ..... Passed    0.02 sec

100% tests passed, 0 tests failed out of 1

Total Test time (real) = 0.27 sec
[100%] Built target check_stdairtst
Scanning dependencies of target check
[100%] Built target check
```

Check if all the executed tests PASSED. If not, please contact us by filling a [bug-report](#).

Finally, you should install the compiled and linked library, include files and (optionally) HTML and PDF documentation by typing:

```
make install
```

Depending on the PREFIX settings during configuration, you might need the root (administrator) access to perform this step.

Eventually, you might invoke the following command

```
make clean
```

to remove all files created during compilation process, or even

```
cd ~/dev/sim/stdairgit
rm -rf build && mkdir build
cd build
```

to remove everything.

13 Linking with StdAir

13.1 Table of Contents

- [Introduction](#)
- [Using the pkg-config command](#)
- [Using the stdair-config script](#)
- [M4 macro for the GNU Autotools](#)
- [Using StdAir with dynamic linking](#)

13.2 Introduction

There are two convenient methods of linking your programs with the StdAir library. The first one employs the `'pkg-config'` command (see <http://pkgconfig.freedesktop.org/>), whereas the second one uses `'stdair-config'` script. These methods are shortly described below.

13.3 Using the pkg-config command

`'pkg-config'` is a helper tool used when compiling applications and libraries. It helps you insert the correct compiler and linker options. The syntax of the `'pkg-config'` is as follows:

```
pkg-config <options> <library_name>
```

For instance, assuming that you need to compile an StdAir based program `'my_prog.cpp'`, you should use the following command:

```
g++ `pkg-config --cflags stdair` -o my_prog my_prog.cpp \  
    `pkg-config --libs stdair`
```

For more information see the `'pkg-config'` man pages.

13.4 Using the stdair-config script

StdAir provides a shell script called `'stdair-config'`, which is installed by default in `'$prefix/bin'` (`'/usr/local/bin'`) directory. It can be used to simplify compilation and linking of StdAir based programs. The usage of this script is quite similar to the usage of the `'pkg-config'` command.

Assuming that you need to compile the program `'my_prog.cpp'` you can now do that with the following command:

```
g++ `stdair-config --cflags` -o my_prog my_prog.cpp `stdair-config --libs`
```

A list of `'stdair-config'` options can be obtained by typing:

```
stdair-config --help
```

If the `'stdair-config'` command is not found by your shell, you should add its location `'$prefix/bin'` to the `PATH` environment variable, e.g.:

```
export PATH=/usr/local/bin:$PATH
```

13.5 M4 macro for the GNU Autotools

A M4 macro file is delivered with StdAir, namely 'stdair.m4', which can be found in, e.g., '/usr/share/aclocal'. When used by a 'configure' script, thanks to the 'AM_PATH_STDPAIR' macro (specified in the M4 macro file), the following Makefile variables are then defined:

- 'STDAIR_VERSION' (e.g., defined to 0.2.0)
- 'STDAIR_CFLAGS' (e.g., defined to '-I\${prefix}/include')
- 'STDAIR_LIBS' (e.g., defined to '-L\${prefix}/lib -lstdair')

13.6 Using StdAir with dynamic linking

When using static linking some of the library routines in StdAir are copied into your executable program. This can lead to unnecessary large executables. To avoid having too large executable files you may use dynamic linking instead. Dynamic linking means that the actual linking is performed when the program is executed. This requires that the system is able to locate the shared StdAir library file during your program execution. If you install the StdAir library using a non-standard prefix, the 'LD_LIBRARY_PATH' environment variable might be used to inform the linker of the dynamic library location, e.g.:

```
export LD_LIBRARY_PATH=<StdAir installation prefix>/lib:$LD_LIBRARY_PATH
```

14 Test Rules

This section describes how the functionality of the StdAir library should be verified. In the 'test/stdair' subdirectory, test source files are provided. All functionality should be tested using these test source files.

14.1 The Test Source Files

Each new StdAir module/class should be accompanied with a test source file. The test source file is an implementation in C++ that tests the functionality of a function/class or a group of functions/classes called test suites. The test source file should test relevant parameter settings and input/output relations to guarantee correct functionality of the corresponding classes/functions. The test source files should be maintained using version control and updated whenever new functionality is added to the StdAir library.

The test source file should print relevant data to a standard output that can be used to verify the functionality. All relevant parameter settings should be tested.

The test source file should be placed in the 'test/stdair' subdirectory and should have a name ending with 'TestSuite.cpp'.

14.2 The Reference File

Consider a test source file named `'YieldTestSuite.cpp'`. A reference file named `'YieldTestSuite.ref'` should accompany the test source file. The reference file contains a reference printout of the standard output generated when running the test program. The reference file should be maintained using version control and updated according to the test source file.

14.3 Testing StdAir Library

One can compile and execute all test programs from the `'test/stdair'` sub-directory by typing:

```
% make check
```

after successful compilation of the StdAir library.

15 Users Guide

15.1 Table of Contents

- [Introduction](#)
- [Get Started](#)
 - [Get the StdAir library](#)
 - [Build the StdAir project](#)
 - [Build and Run the Tests](#)
 - [Install the StdAir Project \(Binaries, Documentation\)](#)
- [Exploring the Predefined BOM Tree](#)
 - [Airline Distribution BOM Tree](#)
 - [Airline Network BOM Tree](#)
 - [Airline Inventory BOM Tree](#)
- [Extending the BOM Tree](#)

15.2 Introduction

The `StdAir` library contains classes for airline business management. This document does not cover all the aspects of the `StdAir` library. It does however explain the most important things you need to know in order to start using `StdAir`.

15.3 Get Started

15.3.1 Get the StdAir library

15.3.2 Build the StdAir project

To run the configuration script the first time, go to the top directory (where the StdAir package has been un-packed), and issue either of the following two commands, depending on whether the StdAir project has been checked out from the Subversion repository or downloaded as a tar-ball package from the Sourceforge Web site:

- `./autogen.sh`
- `./configure`

15.3.3 Build and Run the Tests

15.3.4 Install the StdAir Project (Binaries, Documentation)

15.4 Exploring the Predefined BOM Tree

StdAir predefines a BOM (Business Object Model) tree specific to the airline IT arena.

15.4.1 Airline Distribution BOM Tree

- `stdair::TravelSolutionStruct`

15.4.2 Airline Network BOM Tree

- `stdair::FlightPeriod`

15.4.3 Airline Inventory BOM Tree

- `stdair::Inventory`
- `stdair::FlightDate`

15.4.3.1 Airline Inventory Marketing BOM Tree

- `stdair::SegmentDate`
- `stdair::SegmentCabin`
- `stdair::FareFamily`
- `stdair::BookingClass`

15.4.3.2 Airline Inventory Operating BOM Tree

- `stdair::LegDate`
- `stdair::LegCabin`
- `stdair::Bucket`

15.5 Extending the BOM Tree

16 Supported Systems

16.1 Table of Contents

- [Introduction](#)
- [StdAir 3.10.x](#)
 - [Linux Systems](#)
 - * [Fedora Core 4 with ATLAS](#)
 - * [Gentoo Linux with ACML](#)
 - * [Gentoo Linux with ATLAS](#)
 - * [Gentoo Linux with MKL](#)
 - * [Gentoo Linux with NetLib's BLAS and LAPACK](#)
 - * [Red Hat Enterprise Linux with StdAir External](#)
 - * [SUSE Linux 10.0 with NetLib's BLAS and LAPACK](#)
 - * [SUSE Linux 10.0 with MKL](#)
 - [Windows Systems](#)
 - * [Microsoft Windows XP with Cygwin](#)
 - * [Microsoft Windows XP with Cygwin and ATLAS](#)
 - * [Microsoft Windows XP with Cygwin and ACML](#)
 - * [Microsoft Windows XP with MinGW, MSYS and ACML](#)
 - * [Microsoft Windows XP with MinGW, MSYS and StdAir External](#)
 - * [Microsoft Windows XP with MS Visual C++ and Intel MKL](#)
 - [Unix Systems](#)
 - * [SunOS 5.9 with StdAir External](#)
- [StdAir 3.9.1](#)
- [StdAir 3.9.0](#)
- [StdAir 3.8.1](#)

16.2 Introduction

This page is intended to provide a list of StdAir supported systems, i.e. the systems on which configuration, installation and testing process of the StdAir library has been successful. Results are grouped based on minor release number. Therefore, only the latest tests for bug-fix releases are included. Besides, the information on this page is divided into sections dependent on the operating system.

Where necessary, some extra information is given for each tested configuration, e.g. external libraries installed, configuration commands used, etc.

If you manage to compile, install and test the StdAir library on a system not mentioned below, please let us know, so we could update this database.

16.3 StdAir 3.10.x

16.3.1 Linux Systems

16.3.1.1 Fedora Core 4 with ATLAS

- **Platform:** Intel Pentium 4
- **Operating System:** Fedora Core 4 (x86)
- **Compiler:** g++ (GCC) 4.0.2 20051125
- **StdAir release:** 3.10.0
- **External Libraries:** From FC4 distribution:
 - `fftw3.i386-3.0.1-3`
 - `fftw3-devel.i386-3.0.1-3`
 - `atlas-sse2.i386-3.6.0-8.fc4`
 - `atlas-sse2-devel.i386-3.6.0-8.fc4`
 - `blas.i386-3.0-35.fc4`
 - `lapack.i386-3.0-35.fc4`
- **Tests Status:** All tests PASSED
- **Comments:** StdAir configured with:

```
% CXXFLAGS="-O3 -pipe -march=pentium4" ./configure
```
- **Date:** March 7, 2006
- **Tester:** Tony Ottosson

16.3.1.2 Gentoo Linux with ACML

- **Platform:** AMD Sempron 3000+
- **Operating System:** Gentoo Linux 2006.0 (x86 arch)
- **Compiler(s):** g++ (GCC) 3.4.5
- **StdAir release:** 3.10.1
- **External Libraries:** Compiled and installed from portage tree:
 - sci-libs/acml-3.0.0
- **Tests Status:** All tests PASSED
- **Comments:** BLAS and LAPACK libs set by using the following system commands:

```
% eselect blas set ACML
% eselect lapack set ACML
```

StdAir configured with:

```
% export CPPFLAGS="-I/usr/include/acml"
% ./configure --with-blas="-lblas"
```
- **Date:** March 31, 2006
- **Tester:** Adam Piatyszek (ediap)

16.3.1.3 Gentoo Linux with ATLAS

- **Platform:** Intel Pentium M Centrino
- **Operating System:** Gentoo Linux 2006.0 (x86)
- **Compiler:** g++ (GCC) 3.4.5
- **StdAir release:** 3.10.1
- **External Libraries:** Compiled and installed from portage tree:
 - sci-libs/fftw-3.1
 - sci-libs/blas-atlas-3.6.0-r1
 - sci-libs/lapack-atlas-3.6.0
- **Tests Status:** All tests PASSED
- **Comments:** BLAS and LAPACK libs set by using the following system commands:

```
% eselect blas set ATLAS
% eselect lapack set ATLAS
```

StdAir configured with:

```
% ./configure --with-blas="-lblas"
```
- **Date:** March 31, 2006
- **Tester:** Adam Piatyszek (ediap)

16.3.1.4 Gentoo Linux with MKL

- **Platform:** Intel Pentium M Centrino
- **Operating System:** Gentoo Linux 2006.0 (x86 arch)
- **Compiler:** g++ (GCC) 3.4.5
- **StdAir release:** 3.10.0
- **External Libraries:** Intel Math Kernel Library (MKL) 8.0.1 installed manually in the following directory: `/opt/intel/mkl/8.0.1`
- **Tests Status:** All tests PASSED
- **Comments:** StdAir configured using the following commands:

```
% export LDFLAGS="-L/opt/intel/mkl/8.0.1/lib/32"
% export CPPFLAGS="-I/opt/intel/mkl/8.0.1/include"
% ./configure
```
- **Date:** February 28, 2006
- **Tester:** Adam Piatyszek (ediap)

16.3.1.5 Gentoo Linux with NetLib's BLAS and LAPACK

- **Platform:** Intel Pentium M Centrino
- **Operating System:** Gentoo Linux 2006.0 (x86)
- **Compiler:** g++ (GCC) 3.4.5
- **StdAir release:** 3.10.1
- **External Libraries:** Compiled and installed from portage tree:
 - `sci-libs/fftw-3.1`
 - `sci-libs/blas-reference-19940131-r2`
 - `sci-libs/cblas-reference-20030223`
 - `sci-libs/lapack-reference-3.0-r2`
- **Tests Status:** All tests PASSED
- **Comments:** BLAS and LAPACK libs set by using the following system commands:

```
% blas-config reference
% lapack-config reference
```

StdAir configured with:

```
% ./configure --with-blas="-lblas"
```
- **Date:** March 31, 2006
- **Tester:** Adam Piatyszek (ediap)

16.3.1.6 Red Hat Enterprise Linux with StdAir External

- **Platform:** Intel Pentium 4
- **Operating System:** Red Hat Enterprise Linux AS release 4 (Nahant Update 2)
- **Compiler:** g++ (GCC) 3.4.4 20050721 (Red Hat 3.4.4-2)
- **StdAir release:** 3.10.0
- **External Libraries:** BLAS, CBLAS, LAPACK and FFTW libraries from StdAir External 2.1.1 package
- **Tests Status:** All tests PASSED
- **Date:** March 7, 2006
- **Tester:** Erik G. Larsson

16.3.1.7 SUSE Linux 10.0 with NetLib's BLAS and LAPACK

- **Platform:** Intel Pentium 4 CPU 3.20GHz (64-bit)
- **Operating System:** SUSE Linux 10.0 (x86_64)
- **Compiler(s):** g++ (GCC) 4.0.2
- **StdAir release:** 3.10.0
- **External Libraries:** BLAS, LAPACK and FFTW libraries installed from OpenSuse 10.0 RPM repository:
 - blas-3.0-926
 - lapack-3.0-926
 - fftw3-3.0.1-114
 - fftw3-threads-3.0.1-114
 - fftw3-devel-3.0.1-114
- **Tests Status:** All tests PASSED
- **Comments:** StdAir configured with:

```
% export CXXFLAGS="-m64 -march=nocona -O3 -pipe"
% ./configure --with-lapack="/usr/lib64/liblapack.so.3"
```
- **Date:** March 1, 2006
- **Tester:** Adam Piatyszek (ediap)

16.3.1.8 SUSE Linux 10.0 with MKL

- **Platform:** Intel Pentium 4 CPU 3.20GHz (64-bit)
- **Operating System:** SUSE Linux 10.0 (x86_64)
- **Compiler(s):** g++ (GCC) 4.0.2
- **StdAir release:** 3.10.0
- **External Libraries:** Intel Math Kernel Library (MKL) 8.0.1 installed manually in the following directory: /opt/intel/mkl/8.0.1
- **Tests Status:** All tests PASSED
- **Comments:** StdAir configured with:

```
% export CXXFLAGS="-m64 -march=nocona -O3 -pipe"
% export LDFLAGS="-L/opt/intel/mkl/8.0.1/lib/em64t"
% export CPPFLAGS="-I/opt/intel/mkl/8.0.1/include"
% ./configure
```
- **Date:** March 1, 2006
- **Tester:** Adam Piatyszek (ediap)

16.3.2 Windows Systems

16.3.2.1 Microsoft Windows XP with Cygwin

- **Platform:** AMD Sempron 3000+
- **Operating System:** Microsoft Windows XP SP2, Cygwin 1.5.19-4
- **Compiler(s):** g++ (GCC) 3.4.4 (cygming special)
- **StdAir release:** 3.10.1
- **External Libraries:** Installed from Cygwin's repository:
 - fftw-3.0.1-2
 - fftw-dev-3.0.1-1
 - lapack-3.0-4
- **Tests Status:** All tests PASSED
- **Comments:** Only static library can be built. StdAir configured with:

```
% ./configure
```
- **Date:** March 31, 2006
- **Tester:** Adam Piatyszek (ediap)

16.3.2.2 Microsoft Windows XP with Cygwin and ATLAS

- **Platform:** AMD Sempron 3000+
- **Operating System:** Microsoft Windows XP SP2, Cygwin 1.5.19-4
- **Compiler(s):** g++ (GCC) 3.4.4 (cygming special)
- **StdAir release:** 3.10.1
- **External Libraries:** Installed from Cygwin's repository:

```
- fftw-3.0.1-2
- fftw-dev-3.0.1-1
```

ATLAS BLAS and LAPACK libraries from StdAir External 2.1.1 package configured using:

```
% ./configure --enable-atlas --disable-fftw
```

- **Tests Status:** All tests PASSED
- **Comments:** Only static library can be built. StdAir configured with:

```
% export LDFLAGS="-L/usr/local/lib"
% ./configure
```

- **Date:** March 31, 2006
- **Tester:** Adam Piatyszek (ediap)

16.3.2.3 Microsoft Windows XP with Cygwin and ACML

- **Platform:** AMD Sempron 3000+
- **Operating System:** Microsoft Windows XP SP2, Cygwin 1.5.19-4
- **Compiler(s):** g++ (GCC) 3.4.4 (cygming special)
- **StdAir release:** 3.10.2
- **External Libraries:** ACML version 3.1.0 (acml3.1.0-32-win32-g77.exe) installed into a default directory, i.e. "c:\Program Files\AMD\acml3.1.0"
- **Tests Status:** All tests PASSED
- **Comments:** Only static library can be built. StdAir configured with:

```
% export LDFLAGS="-L/cygdrive/c/Progra~1/AMD/acml3.1.0/gnu32/lib"
% export CPPFLAGS="-I/cygdrive/c/Progra~1/AMD/acml3.1.0/gnu32/include"
% ./configure --enable-debug
```

- **Date:** May 15, 2006
- **Tester:** Adam Piatyszek (ediap)

16.3.2.4 Microsoft Windows XP with MinGW, MSYS and ACML

- **Platform:** AMD Sempron 3000+
- **Operating System:** Microsoft Windows XP SP2, MinGW 5.0.2, MSYS 1.0.10
- **Compiler(s):** g++ (GCC) 3.4.4 (mingw special)
- **StdAir release:** 3.10.2
- **External Libraries:** ACML version 3.1.0 (acml3.1.0-32-win32-g77.exe) installed into a default directory, i.e. "c:\Program Files\AMD\acml3.1.0"
- **Tests Status:** All tests PASSED
- **Comments:** Only static library can be built. StdAir configured with:

```
% export LDFLAGS="-L/c/Progra~1/AMD/acml3.1.0/gnu32/lib"
% export CPPFLAGS="-I/c/Progra~1/AMD/acml3.1.0/gnu32/include"
% ./configure --enable-debug
```
- **Date:** May 15, 2006
- **Tester:** Adam Piatyszek (ediap)

16.3.2.5 Microsoft Windows XP with MinGW, MSYS and StdAir External

- **Platform:** AMD Sempron 3000+
- **Operating System:** Microsoft Windows XP SP2, MinGW 5.0.2, MSYS 1.0.10
- **Compiler(s):** g++ (GCC) 3.4.4 (mingw special)
- **StdAir release:** 3.10.5
- **External Libraries:** BLAS, CBLAS, LAPACK and FFTW libraries from StdAir External 2.2.0 package
- **Tests Status:** All tests PASSED
- **Comments:** Only static library can be built. StdAir configured with:

```
% export LDFLAGS="-L/usr/local/lib"
% export CPPFLAGS="-I/usr/local/include"
% export CXXFLAGS="-Wall -O3 -march=athlon-tbird -pipe"
% ./configure --disable-html-doc
```
- **Date:** August 11, 2006
- **Tester:** Adam Piatyszek (ediap)

16.3.2.6 Microsoft Windows XP with MS Visual C++ and Intel MKL

- **Platform:** AMD Sempron 3000+
- **Operating System:** Microsoft Windows XP SP2
- **Compiler(s):** Microsoft Visual C++ 2005 .NET
- **StdAir release:** 3.10.5
- **External Libraries:** Intel Math Kernel Library (MKL) 8.1 installed manually in the following directory: "C:\Program Files\Intel\MKL\8.1"
- **Tests Status:** Not fully tested. Some StdAir based programs compiled and run with success.
- **Comments:** Only static library can be built. StdAir built by opening the "win32\stdair.vcproj" project file in MSVC++ and executing "Build -> Build Solution" command from menu.
- **Date:** August 11, 2006
- **Tester:** Adam Piatyszek (ediap)

16.3.3 Unix Systems

16.3.3.1 SunOS 5.9 with StdAir External

- **Platform:** SUNW, Sun-Blade-100 (SPARC)
- **Operating System:** SunOS 5.9 Generic_112233-10
- **Compiler(s):** g++ (GCC) 3.4.5
- **StdAir release:** 3.10.2
- **External Libraries:** BLAS, CBLAS, LAPACK and FFTW libraries from StdAir External 2.1.1 package. The following configuration command has been used:

```
% export CFLAGS="-mcpu=ultrasparc -O2 -pipe -funroll-all-loops"  
% ./configure
```

- **Tests Status:** All tests PASSED
- **Comments:** StdAir configured with:

```
% export LDFLAGS="-L/usr/local/lib"  
% export CPPFLAGS="-I/usr/local/include"  
% export CXXFLAGS="-mcpu=ultrasparc -O2 -pipe"  
% ./configure --enable-debug
```

- **Date:** May 15, 2006
- **Tester:** Adam Piatyszek (ediap)

17 StdAir Supported Systems (Previous Releases)

17.1 StdAir 3.9.1

17.2 StdAir 3.9.0

17.3 StdAir 3.8.1

18 Tutorials

18.1 Table of Contents

- [Introduction](#)
 - [Preparing the StdAir Project for Development](#)
- [Build a Predefined BOM Tree](#)
 - [Instantiate the BOM Root Object](#)
 - [Instantiate the \(Airline\) Inventory Object](#)
 - [Link the Inventory Object with the BOM Root](#)
 - [Build Another Airline Inventory](#)
 - [Dump The BOM Tree Content](#)
 - [Result of the Tutorial Program](#)
- [Extend the Pre-Defined BOM Tree](#)
 - [Extend an Airline Inventory Object](#)
 - [Build the Specific BOM Objects](#)
 - [Result of the Tutorial Program](#)

18.2 Introduction

This page contains some tutorial examples that will help you getting started using StdAir. Most examples show how to construct some simple business objects, i.e., instances of the so-named Business Object Model (BOM).

18.2.1 Preparing the StdAir Project for Development

The source code for these examples can be found in the `batches` and `test/stdair` directories. They are compiled along with the rest of the StdAir project. See the User Guide ([Users Guide](#)) for more details on how to build the StdAir project.

18.3 Build a Predefined BOM Tree

A few steps:

- [Instantiate the BOM Root Object](#)
- [Instantiate the \(Airline\) Inventory Object](#)
- [Link the Inventory Object with the BOM Root](#)

18.3.1 Instantiate the BOM Root Object

First, a BOM root object (i.e., a root for all the classes in the project) is instantiated by the `stdair::STDAIR_ServiceContext` context object, when the `stdair::STDAIR_Service` is itself instantiated. The corresponding `StdAir` type (class) is `stdair::BomRoot`.

In the following sample, that object is named `ioBomRoot`, and is given as input/output parameter of the `stdair::CmdBomManager::buildSampleBom()` method:

```
void CmdBomManager::buildSampleBom (BomRoot& ioBomRoot) {
```

18.3.2 Instantiate the (Airline) Inventory Object

An airline inventory object can then be instantiated. Let us give it the "BA" airline code (corresponding to [British Airways](#)) as the object key. That is, an object (let us name it `lBAKey`) of type (class) `stdair::InventoryKey` has first to be instantiated.

```
const InventoryKey lBAKey ("BA");
```

Thanks to that key, an airline inventory object, i.e. of type (class) `stdair::Inventory`, can be instantiated. Let us name that airline inventory object `lBAInv`.

```
Inventory& lBAInv = FacBom<Inventory>::instance().create (lBAKey);
```

18.3.3 Link the Inventory Object with the BOM Root

Then, both objects have to be linked: the airline inventory object (`stdair::Inventory`) has to be linked with the root of the BOM tree (`stdair::BomRoot`). That operation is as simple as using the `stdair::FacBomManager::addToListAndMap()` method:

```
FacBomManager::addToListAndMap (ioBomRoot, lBAInv);  
FacBomManager::linkWithParent (ioBomRoot, lBAInv);
```

18.3.4 Build Another Airline Inventory

Another airline inventory object, corresponding to the Air France ([Air France](#)) company, is instantiated the same way:

```
const InventoryKey lAFKey ("AF");
Inventory& lAFInv = FacBom<Inventory>::instance().create (lAFKey);
FacBomManager::addToListAndMap (ioBomRoot, lAFInv);
FacBomManager::linkWithParent (ioBomRoot, lAFInv);
```

See the corresponding full program ([C++ Class Building Sample StdAir BOM Trees](#)) for more details.

18.3.5 Dump The BOM Tree Content

From the BomRoot (of type `stdair::BomRoot`) object instance, the list of airline inventories (of type `stdair::Inventory`) can then be retrieved...

```
const InventoryList_T& lInventoryList =
    BomManager::getList<Inventory> (iBomRoot);
```

... and browsed:

```
for (InventoryList_T::const_iterator itInv = lInventoryList.begin();
     itInv != lInventoryList.end(); ++itInv, ++invIdx) {
    const Inventory* lInv_ptr = *itInv;
    assert (lInv_ptr != NULL);

    // Retrieve the inventory key (airline code)
    const AirlineCode_T& lAirlineCode = lInv_ptr->getAirlineCode();

    // Display only the requested inventories
    if (iAirlineCode == "all" || iAirlineCode == lAirlineCode) {
        // Get the list of flight-dates for that inventory
        list (oStream, *lInv_ptr, invIdx, iFlightNumber);
    }
}

// ////////////////////////////////////////
void BomDisplay::list (std::ostream& oStream, const Inventory& iInventory,
                     const unsigned short iInventoryIndex,
                     const FlightNumber_T& iFlightNumber) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    // Check whether there are FlightDate objects
    if (BomManager::hasMap<FlightDate> (iInventory) == false) {
        return;
    }

    //
    const AirlineCode_T& lAirlineCode = iInventory.getAirlineCode();
    oStream << iInventoryIndex << ". " << lAirlineCode << std::endl;
```

```

// Browse the flight-dates
unsigned short lCurrentFlightNumber = 0;
unsigned short flightNumberIdx = 0;
unsigned short departureDateIdx = 1;
const FlightDateMap_T& lFlightDateList =
    BomManager::getMap<FlightDate> (iInventory);
for (FlightDateMap_T::const_iterator itFD = lFlightDateList.begin();
     itFD != lFlightDateList.end(); ++itFD, ++departureDateIdx) {
    const FlightDate* lFD_ptr = itFD->second;
    assert (lFD_ptr != NULL);

    // Retrieve the key of the flight-date
    const FlightNumber_T& lFlightNumber = lFD_ptr->getFlightNumber();
    const Date_T& lFlightDateDate = lFD_ptr->getDepartureDate();

    // Display only the requested flight number
    if (iFlightNumber == 0 || iFlightNumber == lFlightNumber) {
        //
        if (lCurrentFlightNumber != lFlightNumber) {
            lCurrentFlightNumber = lFlightNumber;
            ++flightNumberIdx; departureDateIdx = 1;
            oStream << " " << iInventoryIndex << "." << flightNumberIdx << ". "
                << lAirlineCode << lFlightNumber << std::endl;
        }

        oStream << " " << iInventoryIndex << "." << flightNumberIdx
            << "." << departureDateIdx << ". "
            << lAirlineCode << lFlightNumber << " / " << lFlightDateDate
            << std::endl;
    }
}
}

// ////////////////////////////////////////
void BomDisplay::listAirportPairDateRange (std::ostream& oStream,
                                           const BomRoot& iBomRoot) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    // Check whether there are AirportPair objects
    if (BomManager::hasList<AirportPair> (iBomRoot) == false) {
        return;
    }

    const AirportPairList_T& lAirportPairList =
        BomManager::getList<AirportPair> (iBomRoot);
    for (AirportPairList_T::const_iterator itAir = lAirportPairList.begin();
         itAir != lAirportPairList.end(); ++itAir ) {
        const AirportPair* lAir_ptr = *itAir;
        assert (lAir_ptr != NULL);

        // Check whether there are date-period objects
        assert (BomManager::hasList<DatePeriod> (*lAir_ptr) == true);

        // Browse the date-period objects
        const DatePeriodList_T& lDatePeriodList =
            BomManager::getList<DatePeriod> (*lAir_ptr);

        for (DatePeriodList_T::const_iterator itDP = lDatePeriodList.begin();
             itDP != lDatePeriodList.end(); ++itDP) {
            const DatePeriod* lDP_ptr = *itDP;

```

```

        assert (lDP_ptr != NULL);

        // Display the date-period object
        oStream << lAir_ptr->describeKey()
                <<" / " << lDP_ptr->describeKey() << std::endl;
    }

}

// ////////////////////////////////////////
void BomDisplay::csvDisplay (std::ostream& oStream,

```

See the corresponding full program ([C++ Utility Class Browsing and Dumping the StdAir BOM Tree](#)) for more details.

18.3.6 Result of the Tutorial Program

When the `stdair.cpp` program is run (with the `-b` option), the output should look like:

```

[D]../batches/stdair.cpp:243: Welcome to stdair
[D]../batches/stdair/command/CmdBomManager.cpp:41: StdAir will build the BOM tree from
    built-in specifications.
[D]../batches/stdair.cpp:286:
=====
BomRoot:  -- ROOT --
=====
+++++
Inventory: BA
+++++
*****
FlightDate: BA9, 2011-Jun-10
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
apsed, Distance, Capacity,
BA9 2011-Jun-10, LHR-BKK, 2011-Jun-10, 21:45:00, 2011-Jun-11, 15:40:00, 11:05:00,
    1, 06:50:00, 9900, 0,
BA9 2011-Jun-10, BKK-SYD, 2011-Jun-11, 17:05:00, 2011-Jun-12, 15:40:00, 09:05:00,
    1, 13:30:00, 8100, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
    Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
BA9 2011-Jun-10, LHR-BKK 2011-Jun-10, Y, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 9, 0, 0
    , 3.52965e-319, 0, 0,
BA9 2011-Jun-10, BKK-SYD 2011-Jun-11, Y, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 9, 0, 0
    , 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,

```

```

*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
BA9 2011-Jun-10, LHR-SYD 2011-Jun-10, Y, EcoSaver, 0, 0, 0, 0, 9, 0,
BA9 2011-Jun-10, LHR-BKK 2011-Jun-10, Y, EcoSaver, 0, 0, 0, 0, 9, 0,
BA9 2011-Jun-10, BKK-SYD 2011-Jun-11, Y, EcoSaver, 0, 0, 0, 0, 9, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
      (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
BA9 2011-Jun-10, LHR-SYD 2011-Jun-10, Y, EcoSaver, Q, 0 (0), 0, 0, 0, 0, 0 (0), 0
      , 0, 0, 0, 0, 0,
+++++
Inventory: AF
+++++
*****
FlightDate: AF84, 2011-Mar-20
*****
*****
Leg-Dates:
-----
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
AF84 2011-Mar-20, CDG-SFO, 2011-Mar-20, 10:40:00, 2011-Mar-20, 12:50:00, 11:10:00
      , 0, -09:00:00, 9900, 0,
*****
*****
LegCabins:
-----
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
AF84 2011-Mar-20, CDG-SFO 2011-Mar-20, Y, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 9, 0,
      0, 0, 0, 0,
*****
*****
Buckets:
-----
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****
*****
SegmentCabins:
-----
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
AF84 2011-Mar-20, CDG-SFO 2011-Mar-20, Y, EcoSaver, 0, 0, 0, 0, 9, 0,
*****
*****
Subclasses:
-----
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
      (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
AF84 2011-Mar-20, CDG-SFO 2011-Mar-20, Y, EcoSaver, Q, 0 (0), 0, 0, 0, 0, 0 (0),
      0, 0, 0, 0, 0, 0,

```

See the corresponding full program ([Command-Line Utility to Demonstrate Typical StdAir Usage](#)) for more details.

18.4 Extend the Pre-Defined BOM Tree

Now that we master how to instantiate the pre-defined StdAir classes, let us see how to extend that BOM.

18.4.1 Extend an Airline Inventory Object

For instance, let us assume that some (IT) provider (e.g., you) would like to have a specific implementation of the `Inventory` object. The corresponding class has just to extend the `stdair::Inventory` class:

```
namespace myprovider {  
    class Inventory : public stdair::Inventory {
```

The STL containers have to be defined accordingly too:

```
typedef std::list<Inventory*> InventoryList_T;
```

See the full class definition ([Specific Implementation of an Airline Inventory](#)) and implementation ([Specific Implementation of an Airline Inventory](#)) for more details.

18.4.2 Build the Specific BOM Objects

The BOM root object (`stdair::BomRoot`) is instantiated the classical way:

```
stdair::BomRoot& lBomRoot = stdairService.getBomRoot();
```

Then, the specific implementation of the airline inventory object (`myprovider::Inventory`) can be instantiated the same way as a standard `Inventory` (`stdair::Inventory`) would be:

```
const stdair::InventoryKey lBAKey (lBAAirlineCode);  
myprovider::Inventory& lBAInv =  
    stdair::FacBom<myprovider::Inventory>::instance().create (lBAKey);
```

Then, the specific implementation of the airline inventory object (`myprovider::Inventory`) is linked to the root of the BOM tree (`stdair::BomRoot`) the same way as the standard `Inventory` (`stdair::Inventory`) would be:

```
stdair::FacBomManager::addToList (lBomRoot, lBAInv);
```

Another specific airline inventory object is instantiated the same way:

```
const stdair::InventoryKey lAFKey (lFAirlineCode);
myprovider::Inventory& lAFInv =
    stdair::FacBom<myprovider::Inventory>::instance().create (lAFKey);
stdair::FacBomManager::addToList (lBomRoot, lAFInv);
```

From the BomRoot (of type `stdair::BomRoot`) object instance, the list of specific airline inventories (of type `stdair::Inventory`) can then be retrieved...

```
const myprovider::InventoryList_T& lInventoryList =
    stdair::BomManager::getList<myprovider::Inventory> (lBomRoot);
```

... and browsed:

```
for (myprovider::InventoryList_T::const_iterator itInv =
    lInventoryList.begin(); itInv != lInventoryList.end();
    ++itInv, ++idx) {
    const myprovider::Inventory* lInv_ptr = *itInv;
    BOOST_REQUIRE (lInv_ptr != NULL);

    BOOST_CHECK_EQUAL (lInventoryKeyArray[idx], lInv_ptr->describeKey());
    BOOST_CHECK_MESSAGE (lInventoryKeyArray[idx] == lInv_ptr->describeKey(),
        "They inventory key, '" << lInventoryKeyArray[idx]
        << "'", does not match that of the Inventory object: '"
        << lInv_ptr->describeKey() << "'");
}
```

18.4.3 Result of the Tutorial Program

When this program is run, the output should look like:

```
Inventory: BA
Inventory: AF
```

See the corresponding full program ([Command-Line Test to Demonstrate How To Extend StdAir BOM](#)) for more details.

19 Command-Line Utility to Demonstrate Typical StdAir Usage

```
*/
// STL
#include <cassert>
#include <iostream>
#include <sstream>
#include <fstream>
#include <string>
// Boost (Extended STL)
#include <boost/date_time/posix_time/posix_time.hpp>
#include <boost/date_time/gregorian/gregorian.hpp>
#include <boost/program_options.hpp>
#include <boost/tokenizer.hpp>
```

```

#include <boost/lexical_cast.hpp>
// StdAir
#include <stdair/stdair_types.hpp>
#include <stdair/bom/BomArchive.hpp>
#include <stdair/bom/BookingRequestStruct.hpp>
#include <stdair/bom/TravelSolutionStruct.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/STDAIR_Service.hpp>
#include <stdair/config/stdair-paths.hpp>

// ////////// Constants //////////
const std::string K_STDAIR_DEFAULT_LOG_FILENAME ("stdair.log");

const std::string K_STDAIR_DEFAULT_INPUT_FILENAME (STDAIR_SAMPLE_DIR
                                                    "/schedule01.csv");

const bool K_STDAIR_DEFAULT_BUILT_IN_INPUT = false;

const bool K_STDAIR_DEFAULT_BUILT_FOR_RMOL = false;

const bool K_STDAIR_DEFAULT_BUILT_FOR_CRIS = false;

const int K_STDAIR_EARLY_RETURN_STATUS = 99;

// ////////// Parsing of Options & Configuration //////////
// A helper function to simplify the main part.
template<class T> std::ostream& operator<< (std::ostream& os,
                                           const std::vector<T>& v) {
    std::copy (v.begin(), v.end(), std::ostream_iterator<T> (std::cout, " "));
    return os;
}

int readConfiguration (int argc, char* argv[], bool& ioIsBuiltin,
                      bool& ioIsForRMOL, bool& ioIsForCRIS,
                      stdair::Filename_T& ioInputFilename,
                      std::string& ioLogFilename) {
    // Default for the built-in input
    ioIsBuiltin = K_STDAIR_DEFAULT_BUILT_IN_INPUT;

    // Default for the RMOL input
    ioIsForRMOL = K_STDAIR_DEFAULT_BUILT_FOR_RMOL;

    // Default for the CRIS input
    ioIsForCRIS = K_STDAIR_DEFAULT_BUILT_FOR_CRIS;

    // Declare a group of options that will be allowed only on command line
    boost::program_options::options_description generic ("Generic options");
    generic.add_options()
        ("prefix", "print installation prefix")
        ("version,v", "print version string")
        ("help,h", "produce help message");

    // Declare a group of options that will be allowed both on command
    // line and in config file

    boost::program_options::options_description config ("Configuration");
    config.add_options()
        ("builtin,b",
         "The sample BOM tree can be either built-in or parsed from an input file. Th
         at latter must then be given with the -i/--input option")
        ("rmol,r",
         "Build a sample BOM tree for RMOL (i.e., a dummy flight-date with a single l

```

```

        eg-cabin)")
    ("crs,c",
     "Build a sample BOM tree for CRS")
    ("input,i",
     boost::program_options::value< std::string >(&ioInputFilename)->default_value(
       K_STDAIR_DEFAULT_INPUT_FILENAME),
     "(CVS) input file for the demand distributions")
    ("log,l",
     boost::program_options::value< std::string >(&ioLogFilename)->default_value(
       K_STDAIR_DEFAULT_LOG_FILENAME),
     "Filename for the logs")
    ;

    // Hidden options, will be allowed both on command line and
    // in config file, but will not be shown to the user.
    boost::program_options::options_description hidden ("Hidden options");
    hidden.add_options()
        ("copyright",
         boost::program_options::value< std::vector<std::string> >(),
         "Show the copyright (license)");

    boost::program_options::options_description cmdline_options;
    cmdline_options.add(generic).add(config).add(hidden);

    boost::program_options::options_description config_file_options;
    config_file_options.add(config).add(hidden);
    boost::program_options::options_description visible ("Allowed options");
    visible.add(generic).add(config);

    boost::program_options::positional_options_description p;
    p.add ("copyright", -1);

    boost::program_options::variables_map vm;
    boost::program_options::
        store (boost::program_options::command_line_parser (argc, argv).
              options (cmdline_options).positional(p).run(), vm);

    std::ifstream ifs ("stdair.cfg");
    boost::program_options::store (parse_config_file (ifs, config_file_options),
                                  vm);
    boost::program_options::notify (vm);

    if (vm.count ("help")) {
        std::cout << visible << std::endl;
        return K_STDAIR_EARLY_RETURN_STATUS;
    }

    if (vm.count ("version")) {
        std::cout << PACKAGE_NAME << ", version " << PACKAGE_VERSION << std::endl;
        return K_STDAIR_EARLY_RETURN_STATUS;
    }

    if (vm.count ("prefix")) {
        std::cout << "Installation prefix: " << PREFIXDIR << std::endl;
        return K_STDAIR_EARLY_RETURN_STATUS;
    }

    if (vm.count ("builtin")) {
        ioIsBuiltin = true;
    }

    if (vm.count ("rmol")) {

```

```

        ioIsForRMOL = true;

        // The RMOL sample tree takes precedence over the default built-in BOM tree
        ioIsBuiltin = false;
    }

    if (vm.count ("crs")) {
        ioIsForCRS = true;

        // The RMOL sample tree takes precedence over the default built-in BOM tree
        ioIsBuiltin = false;
    }

    const std::string isBuiltinStr = (ioIsBuiltin == true)? "yes": "no";
    std::cout << "The BOM should be built-in? " << isBuiltinStr << std::endl;

    const std::string isForRMOLStr = (ioIsForRMOL == true)? "yes": "no";
    std::cout << "The BOM should be built-in for RMOL? " << isForRMOLStr
        << std::endl;

    const std::string isForCRSStr = (ioIsForCRS == true)? "yes": "no";
    std::cout << "The BOM should be built-in for CRS? " << isForCRSStr
        << std::endl;

    if (ioIsBuiltin == false && ioIsForRMOL == false && ioIsForCRS == false) {
        if (vm.count ("input")) {
            ioInputFilename = vm["input"].as< std::string >();
            std::cout << "Input filename is: " << ioInputFilename << std::endl;
        } else {
            std::cerr << "Either one among the -b/--builtin, -r/--rmol, -c/--crs "
                << "or -i/--input options must be specified" << std::endl;
        }
    }

    if (vm.count ("log")) {
        ioLogFilename = vm["log"].as< std::string >();
        std::cout << "Log filename is: " << ioLogFilename << std::endl;
    }

    return 0;
}

// ////////////////////////////////// M A I N //////////////////////////////////
int main (int argc, char* argv[]) {

    // State whether the BOM tree should be built-in or parsed from an
    // input file
    bool isBuiltin;

    // State whether a sample BOM tree should be built for RMOL.
    bool isForRMOL;

    // State whether a sample BOM tree should be built for the CRS.
    bool isForCRS;

    // Input file name
    stdair::Filename_T lInputFilename;

    // Output log File
    std::string lLogFilename;

```

```
// Call the command-line option parser
const int lOptionParserStatus =
    readConfiguration (argc, argv, isBuiltin, isForRMOL, isForCRS,
                      lInputFilename, lLogFilename);

if (lOptionParserStatus == K_STDAIR_EARLY_RETURN_STATUS) {
    return 0;
}

// Set the log parameters
std::ofstream logOutputFile;
// Open and clean the log outputfile
logOutputFile.open (lLogFilename.c_str());
logOutputFile.clear();

const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
stdair::STDAIR_Service stdairService (lLogParams);

// DEBUG
STDAIR_LOG_DEBUG ("Welcome to stdair");

// Check whether or not a (CSV) input file should be read
if (isBuiltin == true || isForRMOL == true || isForCRS == true) {

    if (isForRMOL == true) {
        // Build the sample BOM tree for RMOL
        stdairService.buildDummyInventory (300);
    } else if (isForCRS == true) {
        //
        stdair::TravelSolutionList_T lTravelSolutionList;
        stdairService.buildSampleTravelSolutions (lTravelSolutionList);

        // Build the sample BOM tree for CRS
        const stdair::BookingRequestStruct& lBookingRequest =
            stdairService.buildSampleBookingRequest();

        // DEBUG: Display the travel solution and booking request
        STDAIR_LOG_DEBUG ("Booking request: " << lBookingRequest.display());

        const std::string& lCSVDump =
            stdairService.csvDisplay (lTravelSolutionList);
        STDAIR_LOG_DEBUG (lCSVDump);
    } else {
        assert (isBuiltin == true);

        // Build a sample BOM tree
        stdairService.buildSampleBom();
    }
} else {
    // Read the input file
    //stdairService.readFromInputFile (lInputFilename);

    // DEBUG
    STDAIR_LOG_DEBUG ("StdAir will parse " << lInputFilename
                     << " and build the corresponding BOM tree.");
}

// DEBUG: Display the whole BOM tree
```

```

const std::string& lCSVDump = stdairService.csvDisplay();
STDAIR_LOG_DEBUG (lCSVDump);

// Close the Log outputFile
logOutputFile.close();

/*
Note: as that program is not intended to be run on a server in
production, it is better not to catch the exceptions. When it
happens (that an exception is throwned), that way we get the
call stack.
*/

return 0;
}

/*!

```

20 Specific Implementation of a BOM Root

```

*/
// ////////////////////////////////////////
// Import section
// ////////////////////////////////////////
// STL
#include <cassert>
// StdAir Test
#include <test/stdair/MPBomRoot.hpp>

namespace myprovider {

// ////////////////////////////////////////
BomRoot::BomRoot (const Key_T& iKey) : stdair::BomRoot (iKey) {
}

// ////////////////////////////////////////
BomRoot::~BomRoot () {
}

}

/*!

```

21 Specific Implementation of a BOM Root

```

*/
// ////////////////////////////////////////
// Import section
// ////////////////////////////////////////
// STL
#include <string>
// StdAir
#include <stdair/bom/BomRoot.hpp>

namespace myprovider {

class BomRoot : public stdair::BomRoot {

```

```

public:
    // //////////// Display support methods ////////////
    std::string toString() const { return describeKey(); }

    const std::string describeKey() const { return std::string (""); }

public:
    BomRoot (const Key_T&);
    ~BomRoot ();
    BomRoot ();
    BomRoot (const BomRoot&);
};

}
/*!

```

22 Specific Implementation of an Airline Inventory

```

*/
// ////////////////////////////////////////
// Import section
// ////////////////////////////////////////
// STL
#include <cassert>
// StdAir
#include <stdair/stdair_inventory_types.hpp>
// StdAir Test
#include <test/stdair/MPInventory.hpp>

namespace myprovider {

    // ////////////////////////////////////////
    Inventory::Inventory (const Key_T& iKey) : stdair::Inventory (iKey) {
    }

    // ////////////////////////////////////////
    Inventory::~~Inventory () {
    }

    // ////////////////////////////////////////
    std::string Inventory::toString() const {
        std::ostringstream oStr;
        oStr << _key.toString();
        return oStr.str();
    }

    // ////////////////////////////////////////
    const std::string Inventory::describeKey() const {
        return _key.toString();
    }

}
/*!

```

23 Specific Implementation of an Airline Inventory

```

*/

```

```
// ////////////////////////////////////////
// Import section
// ////////////////////////////////////////
// STL
#include <list>
// StdAir
#include <stdair/bom/Inventory.hpp>

namespace myprovider {

    class Inventory : public stdair::Inventory {
    public:
        // /////////// Display support methods ///////////
        std::string toString() const;

        const std::string describeKey() const;

    public:
        Inventory (const Key_T&);
        ~Inventory();
        Inventory ();
        Inventory (const Inventory&);
    };

    // /////////// Type definitions ///////////
    typedef std::list<Inventory*> InventoryList_T;

}
/*!
```

24 Command-Line Test to Demonstrate How To Extend StdAir BOM

```
*/
// ////////////////////////////////////////
// Import section
// ////////////////////////////////////////
// STL
#include <sstream>
#include <fstream>
#include <string>
// Boost MPL
#include <boost/mpl/push_back.hpp>
#include <boost/mpl/vector.hpp>
#include <boost/mpl/at.hpp>
#include <boost/mpl/assert.hpp>
#include <boost/type_traits/is_same.hpp>
// Boost Unit Test Framework (UTF)
#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MAIN
#define BOOST_TEST_MODULE StdAirTest
#if BOOST_VERSION >= 103900
#include <boost/test/unit_test.hpp>
#else // BOOST_VERSION >= 103900
#include <boost/test/test_tools.hpp>
#include <boost/test/results_reporter.hpp>
#include <boost/test/unit_test_suite.hpp>
#include <boost/test/output_test_stream.hpp>
#include <boost/test/unit_test_log.hpp>
#include <boost/test/framework.hpp>
```

```

#include <boost/test/detail/unit_test_parameters.hpp>
#endif // BOOST_VERSION >= 103900
// Boost Serialisation
#include <boost/archive/text_oarchive.hpp>
#include <boost/archive/text_iarchive.hpp>
// StdAir
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/STDAIR_Service.hpp>
#include <stdair/basic/float_utils.hpp>
#include <stdair/bom/BomDisplay.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/factory/FacBom.hpp>
#include <stdair/factory/FacBomManager.hpp>
// StdAir Test Suite
#include <test/stdair/StdairTestLib.hpp>
#include <test/stdair/MPInventory.hpp>

namespace boost_utf = boost::unit_test;

#if BOOST_VERSION >= 103900

// (Boost) Unit Test XML Report
std::ofstream utfReportStream ("StandardAirlineITTestSuite_utfresults.xml");

struct UnitTestConfig {
    UnitTestConfig() {
        boost_utf::unit_test_log.set_stream (utfReportStream);
        boost_utf::unit_test_log.set_format (boost_utf::XML);
        boost_utf::unit_test_log.set_threshold_level (boost_utf::log_test_units);
        // boost_utf::unit_test_log.set_threshold_level (boost_utf::log_successful_tests);
    }

    ~UnitTestConfig() {
    }
};

// ////////////////////////////////// Main: Unit Test Suite //////////////////////////////////

// Set the UTF configuration (re-direct the output to a specific file)
BOOST_GLOBAL_FIXTURE (UnitTestConfig);

// Start the test suite
BOOST_AUTO_TEST_SUITE (master_test_suite)

BOOST_AUTO_TEST_CASE (float_comparison_test) {
    float a = 0.2f;
    a = 5*a;
    const float b = 1.0f;

    // Test the Boost way
    BOOST_CHECK_MESSAGE (a == b, "The two floats (" << a << " and " << b
        << ") should be equal, but are not");
    BOOST_CHECK_CLOSE (a, b, 0.0001);

    // Test the Google way
    const FloatingPoint<float> lhs (a), rhs (b);
    BOOST_CHECK_MESSAGE (lhs.AlmostEquals (rhs),

```

```

        "The two floats (" << a << " and " << b
        << ") should be equal, but are not");
    }

BOOST_AUTO_TEST_CASE (mpl_structure_test) {
    const stdair::ClassCode_T lBookingClassCodeA ("A");
    const stdair_test::BookingClass lA (lBookingClassCodeA);
    const stdair_test::Cabin lCabin (lA);

    BOOST_CHECK_EQUAL (lCabin.toString(), lBookingClassCodeA);
    BOOST_CHECK_MESSAGE (lCabin.toString() == lBookingClassCodeA,
        "The cabin key, '" << lCabin.toString()
        << "' is not equal to '" << lBookingClassCodeA << "'");

    // MPL
    typedef boost::mpl::vector<stdair_test::BookingClass> MPL_BookingClass;
    typedef boost::mpl::push_back<MPL_BookingClass,
        stdair_test::Cabin>::type types;

    if (boost::is_same<stdair_test::BookingClass,
        stdair_test::Cabin::child>::value == false) {
        BOOST_REQUIRE ("The two types mut be equal, but are not");
    }

    if (boost::is_same<boost::mpl::at_c<types, 1>::type,
        stdair_test::Cabin>::value == false) {
        BOOST_REQUIRE ("The type must be stdair_test::Cabin, but is not");
    }
}

BOOST_AUTO_TEST_CASE (stdair_service_initialisation_test) {
    // Output log File
    const std::string lLogFilename ("StandardAirlineITTestSuite_init.log");

    // Set the log parameters
    std::ofstream logOutputFile;

    // Open and clean the log outputfile
    logOutputFile.open (lLogFilename.c_str());
    logOutputFile.clear();

    // Initialise the stdair BOM
    const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
    stdair::STDAIR_Service stdairService (lLogParams);

    // Retrieve (a reference on) the top of the BOM tree
    stdair::BomRoot& lBomRoot = stdairService.getBomRoot();

    // Retrieve the BomRoot key, and compare it to the expected one
    const std::string& lBomRootKeyStr = lBomRoot.describeKey();
    const std::string lBomRootString (" -- ROOT -- ");

    // DEBUG
    STDAIR_LOG_DEBUG ("The BOM root key is '" << lBomRootKeyStr
        << "'. It should be equal to '" << lBomRootString << "'");

    BOOST_CHECK_EQUAL (lBomRootKeyStr, lBomRootString);
    BOOST_CHECK_MESSAGE (lBomRootKeyStr == lBomRootString,
        "The BOM root key, '" << lBomRootKeyStr
        << "', should be equal to '" << lBomRootString
        << "', but is not.");
}

```

```

// Build a sample BOM tree
stdairService.buildSampleBom();

// DEBUG: Display the whole BOM tree
const std::string& lCSVDump = stdairService.csvDisplay();
STDAIR_LOG_DEBUG (lCSVDump);

// Close the Log outputFile
logOutputFile.close();
}

BOOST_AUTO_TEST_CASE (bom_structure_instantiation_test) {
    // Step 0.0: initialisation
    // Create the root of the Bom tree (i.e., a BomRoot object)
    stdair::BomRoot& lBomRoot =
        stdair::FacBom<stdair::BomRoot>::instance().create();

    // Step 0.1: Inventory level
    // Create an Inventory (BA)
    const stdair::AirlineCode_T lBAAirlineCode ("BA");
    const stdair::InventoryKey lBAKey (lBAAirlineCode);
    myprovider::Inventory& lBAInv =
        stdair::FacBom<myprovider::Inventory>::instance().create (lBAKey);
    stdair::FacBomManager::addToList (lBomRoot, lBAInv);

    BOOST_CHECK_EQUAL (lBAInv.describeKey(), lBAAirlineCode);
    BOOST_CHECK_MESSAGE (lBAInv.describeKey() == lBAAirlineCode,
        "The inventory key, '" << lBAInv.describeKey()
        << "', should be equal to '" << lBAAirlineCode
        << "', but is not");

    // Create an Inventory for AF
    const stdair::AirlineCode_T lFAAirlineCode ("AF");
    const stdair::InventoryKey lAFKey (lFAAirlineCode);
    myprovider::Inventory& lAFInv =
        stdair::FacBom<myprovider::Inventory>::instance().create (lAFKey);
    stdair::FacBomManager::addToList (lBomRoot, lAFInv);

    BOOST_CHECK_EQUAL (lAFInv.describeKey(), lFAAirlineCode);
    BOOST_CHECK_MESSAGE (lAFInv.describeKey() == lFAAirlineCode,
        "The inventory key, '" << lAFInv.describeKey()
        << "', should be equal to '" << lFAAirlineCode
        << "', but is not");

    // Browse the inventories
    const myprovider::InventoryList_T& lInventoryList =
        stdair::BomManager::getList<myprovider::Inventory> (lBomRoot);
    const std::string lInventoryKeyArray[2] = {lBAAirlineCode, lFAAirlineCode};
    short idx = 0;
    for (myprovider::InventoryList_T::const_iterator itInv =
        lInventoryList.begin(); itInv != lInventoryList.end();
        ++itInv, ++idx) {
        const myprovider::Inventory* lInv_ptr = *itInv;
        BOOST_REQUIRE (lInv_ptr != NULL);

        BOOST_CHECK_EQUAL (lInventoryKeyArray[idx], lInv_ptr->describeKey());
        BOOST_CHECK_MESSAGE (lInventoryKeyArray[idx] == lInv_ptr->describeKey(),
            "They inventory key, '" << lInventoryKeyArray[idx]
            << "', does not match that of the Inventory object: '"
            << lInv_ptr->describeKey() << "'");
    }
}

```

```

BOOST_AUTO_TEST_CASE (bom_structure_serialisation_test) {

    // Backup (thanks to Boost.Serialisation) file
    const std::string lBackupFilename = "StandardAirlineITTestSuite_serial.txt";

    // Output log File
    const std::string lLogFilename ("StandardAirlineITTestSuite_serial.log");

    // Set the log parameters
    std::ofstream logOutputFile;

    // Open and clean the log outputfile
    logOutputFile.open (lLogFilename.c_str());
    logOutputFile.clear();

    // Initialise the stdair BOM
    const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
    stdair::STDAIR_Service stdairService (lLogParams);

    // Build a sample BOM tree
    stdairService.buildSampleBom();

    // DEBUG: Display the whole BOM tree
    const std::string& lCSVDump = stdairService.csvDisplay();
    STDAIR_LOG_DEBUG (lCSVDump);

    // Retrieve (a reference on) the top of the BOM tree
    stdair::BomRoot& lBomRoot = stdairService.getBomRoot();

    // Retrieve the BomRoot key, and compare it to the expected one
    const std::string lBAInvKeyStr ("BA");
    stdair::Inventory* lBAInv_ptr = lBomRoot.getInventory (lBAInvKeyStr);

    // DEBUG
    STDAIR_LOG_DEBUG ("There should be an Inventory object corresponding to the ' "
        << lBAInvKeyStr << " key.");

    BOOST_REQUIRE_MESSAGE (lBAInv_ptr != NULL,
        "An Inventory object should exist with the key, ' "
        << lBAInvKeyStr << " ".");

    // create and open a character archive for output
    std::ofstream ofs (lBackupFilename.c_str());

    // save data to archive
    {
        boost::archive::text_oarchive oa (ofs);
        // write class instance to archive
        oa << lBomRoot;
        // archive and stream closed when destructors are called
    }

    // ... some time later restore the class instance to its original state
    stdair::BomRoot& lRestoredBomRoot =
        stdair::FacBom<stdair::BomRoot>::instance().create();
    {
        // create and open an archive for input
        std::ifstream ifs (lBackupFilename.c_str());
        boost::archive::text_iarchive ia(ifs);
        // read class state from archive
        ia >> lRestoredBomRoot;
    }
}

```

```

    // archive and stream closed when destructors are called
}

// DEBUG: Display the whole BOM tree
std::ostringstream oRestoredCSVDumpStr;
stdair::BomDisplay::csvDisplay (oRestoredCSVDumpStr, lRestoredBomRoot);
STDAIR_LOG_DEBUG (oRestoredCSVDumpStr.str());

// Retrieve the BomRoot key, and compare it to the expected one
const std::string& lBomRootKeyStr = lRestoredBomRoot.describeKey();
const std::string lBomRootString (" -- ROOT -- ");

// DEBUG
STDAIR_LOG_DEBUG ("The BOM root key is '" << lBomRootKeyStr
                  << "'. It should be equal to '" << lBomRootString << "'");

BOOST_CHECK_EQUAL (lBomRootKeyStr, lBomRootString);
BOOST_CHECK_MESSAGE (lBomRootKeyStr == lBomRootString,
                    "The BOM root key, '" << lBomRootKeyStr
                    << "', should be equal to '" << lBomRootString
                    << "', but is not.");

// Retrieve the Inventory
stdair::Inventory* lRestoredBAInv_ptr =
    lRestoredBomRoot.getInventory (lBAInvKeyStr);

// DEBUG
STDAIR_LOG_DEBUG ("There should be an Inventory object corresponding to the '"
                  << lBAInvKeyStr << "' key.");

BOOST_CHECK_MESSAGE (lRestoredBAInv_ptr != NULL,
                    "An Inventory object should exist with the key, '"
                    << lBAInvKeyStr << "'");

// Close the Log outputFile
logOutputFile.close();
}

// End the test suite
BOOST_AUTO_TEST_SUITE_END()

#else // BOOST_VERSION >= 103900
boost_utf::test_suite* init_unit_test_suite (int, char* []) {
    boost_utf::test_suite* test = BOOST_TEST_SUITE ("Unit test example 1");
    return test;
}
#endif // BOOST_VERSION >= 103900

/*!
```

25 Module Index

25.1 Modules

Here is a list of all modules:

Abstract part of the Business Object Model (BOM)

149

Part of the Business Object Model (BOM) handling	149
--	-----

26 Directory Hierarchy

26.1 Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

batches	151
stdair	157
basic	150
bom	151
command	155
config	155
dbadaptor	155
factory	156
service	156
ui	157
cmdline	155
test	157
stdair	156

27 Namespace Index

27.1 Namespace List

Here is a list of all namespaces with brief descriptions:

boost (Forward declarations)	158
boost::serialization	158
bpt	158
soci	158
stdair (Handle on the StdAir library context)	158

stdair::LOG	235
stdair_test	236
swift (The wrapper namespace)	237

28 Class Index

28.1 Class Hierarchy

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

stdair::BasChronometer	259
stdair::BasFileMgr	264
std::basic_fstream< char >	
std::basic_fstream< wchar_t >	
std::basic_ifstream< char >	
std::basic_ifstream< wchar_t >	
std::basic_ios< char >	
std::basic_ios< wchar_t >	
std::basic_iostream< char >	
std::basic_iostream< wchar_t >	
std::basic_istream< char >	
std::basic_istream< wchar_t >	
std::basic_istreamstream< char >	
std::basic_istreamstream< wchar_t >	
std::basic_ofstream< char >	
std::basic_ofstream< wchar_t >	
std::basic_ostream< char >	
std::basic_ostream< wchar_t >	
std::basic_ostreamstream< char >	
std::basic_ostreamstream< wchar_t >	
std::basic_string< char >	
std::basic_string< wchar_t >	
std::basic_stringstream< char >	
std::basic_stringstream< wchar_t >	
stdair::BomAbstract	268
stdair::AirlineClassList	237
stdair::AirlineFeature	245
stdair::AirportPair	253
stdair::BomHolder< BOM >	280
stdair::BomRoot	302

stdair::BookingClass	310
stdair::Bucket	332
stdair::DatePeriod	352
stdair::EventQueue	375
stdair::FareFamily	413
stdair::FareFeatures	420
stdair::FlightDate	434
stdair::FlightPeriod	443
stdair::GuillotineBlock	457
stdair::Inventory	471
stdair::LegCabin	484
stdair::LegDate	502
stdair::OnDDate	526
stdair::PosChannel	556
stdair::SegmentCabin	584
stdair::SegmentDate	597
stdair::SegmentPeriod	609
stdair::TimePeriod	653
stdair::YieldFeatures	670
stdair::YieldStore	679
stdair::BomArchive	271
stdair::BomDisplay	272
stdair::BomJSONExport	286
stdair::BomJSONImport	287
stdair::BomKeyManager	288
stdair::BomManager	290
stdair::BomRetriever	293

stdair_test::BookingClass	309
stdair_test::Cabin	340
stdair::CmdAbstract	344
stdair::CmdBomManager	344
stdair::CmdBomSerialiser	345
stdair::DBManagerForAirlines	360
COMMAND	348
stdair::ContinuousAttributeLite< T >	349
stdair::date_time_element< MIN, MAX >	351
stdair::DbAbstract	359
stdair::DBSessionManager	361
stdair::DefaultDCPList	362
stdair::DefaultDtdFratMap	363
stdair::DefaultDtdProbMap	363
stdair::DictionaryManager	368
stdair::FacAbstract	398
stdair::FacBom< BOM >	399
stdair::FacBomManager	401
stdair::FacServiceAbstract	406
stdair::FacSTDAIRServiceContext	408
stdair::FacSupervisor	410
stdair::FlagSaver	434
FloatingPoint< RawType >	449
stdair::KeyAbstract	479
stdair::AirlineClassListKey	242
stdair::AirlineFeatureKey	249
stdair::AirportPairKey	257

<code>stdair::BomHolderKey</code>	284
<code>stdair::BomRootKey</code>	307
<code>stdair::BookingClassKey</code>	324
<code>stdair::BucketKey</code>	337
<code>stdair::DatePeriodKey</code>	357
<code>stdair::EventQueueKey</code>	388
<code>stdair::FareFamilyKey</code>	418
<code>stdair::FareFeaturesKey</code>	426
<code>stdair::FlightDateKey</code>	441
<code>stdair::FlightPeriodKey</code>	447
<code>stdair::GuillotineBlockKey</code>	467
<code>stdair::InventoryKey</code>	476
<code>stdair::LegCabinKey</code>	499
<code>stdair::LegDateKey</code>	511
<code>stdair::OnDDateKey</code>	532
<code>stdair::ParsedKey</code>	539
<code>stdair::PosChannelKey</code>	560
<code>stdair::SegmentCabinKey</code>	595
<code>stdair::SegmentDateKey</code>	606
<code>stdair::SegmentPeriodKey</code>	616
<code>stdair::TimePeriodKey</code>	657
<code>stdair::YieldFeaturesKey</code>	674
<code>stdair::YieldStoreKey</code>	683
<code>stdair::Logger</code>	513
<code>stdair::RootException</code>	577
<code>stdair::DocumentNotFoundException</code>	369
<code>stdair::EventException</code>	374

stdair::EventQueueException	387
stdair::FileNotFoundException	432
stdair::KeyNotFoundException	482
stdair::MemoryAllocationException	514
stdair::NonInitialisedContainerException	515
stdair::NonInitialisedDBSessionManagerException	517
stdair::NonInitialisedLogServiceException	518
stdair::NonInitialisedRelationShipException	519
stdair::NonInitialisedServiceException	520
stdair::ObjectLinkingException	523
stdair::ObjectNotFoundException	524
stdair::ParserException	543
stdair::CodeConversionException	345
stdair::CodeDuplicationException	346
stdair::ObjectCreationDuplicationException	522
stdair::ParsingFileFailedException	544
stdair::SerialisationException	618
stdair::SQLDatabaseException	631
stdair::SQLDatabaseConnectionImpossibleException	629
stdair::RootFilePath	579
stdair::InputFilePath	470
stdair::ServiceAbstract	619
stdair::STDAIR_ServiceContext	649
swift::SKeymap	625
swift::SReadline	632
stdair::STDAIR_Service	637
stdair::StructAbstract	650

<code>stdair::AirlineStruct</code>	251
<code>stdair::BasDBParams</code>	260
<code>stdair::BasLogParams</code>	265
<code>stdair::BookingRequestStruct</code>	326
<code>stdair::CancellationStruct</code>	341
<code>stdair::DemandGenerationMethod</code>	364
<code>stdair::DoWStruct</code>	370
<code>stdair::EventStruct</code>	390
<code>stdair::EventType</code>	395
<code>stdair::FareOptionStruct</code>	428
<code>stdair::ForecastingMethod</code>	453
<code>stdair::OptimisationNotificationStruct</code>	535
<code>stdair::PartnershipTechnique</code>	545
<code>stdair::PassengerType</code>	550
<code>stdair::PeriodStruct</code>	553
<code>stdair::ProgressStatus</code>	563
<code>stdair::ProgressStatusSet</code>	567
<code>stdair::RandomGeneration</code>	570
<code>stdair::RMEventStruct</code>	574
<code>stdair::SampleType</code>	580
<code>stdair::ServiceInitialisationType</code>	621
<code>stdair::SnapshotStruct</code>	627
<code>stdair::TravelSolutionStruct</code>	659
<code>stdair::VirtualClassStruct</code>	667
<code>stdair::YieldRange</code>	676
<code>soci::type_conversion< stdair::AirlineStruct ></code>	664
<code>TypeWithSize< size ></code>	665

[TypeWithSize< 4 >](#) [665](#)

[TypeWithSize< 8 >](#) [666](#)

29 Class Index

29.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

[stdair::AirlineClassList](#) (Class representing the actual attributes for a segment-features) [237](#)

[stdair::AirlineClassListKey](#) (Key of airport-pair) [242](#)

[stdair::AirlineFeature](#) [245](#)

[stdair::AirlineFeatureKey](#) [249](#)

[stdair::AirlineStruct](#) [251](#)

[stdair::AirportPair](#) (Class representing the actual attributes for an airport-pair) [253](#)

[stdair::AirportPairKey](#) (Key of airport-pair) [257](#)

[stdair::BasChronometer](#) [259](#)

[stdair::BasDBParams](#) (Structure holding the parameters for connection to a database) [260](#)

[stdair::BasFileMgr](#) [264](#)

[stdair::BasLogParams](#) (Structure holding parameters for logging) [265](#)

[stdair::BomAbstract](#) (Base class for the Business Object Model (BOM) layer) [268](#)

[stdair::BomArchive](#) (Utility class to archive/restore BOM objects with Boost serialisation) [271](#)

[stdair::BomDisplay](#) (Utility class to display StdAir objects with a pretty format) [272](#)

[stdair::BomHolder< BOM >](#) (Class representing the holder of BOM object containers (list and map)) [280](#)

[stdair::BomHolderKey](#) [284](#)

[stdair::BomJSONExport](#) (Utility class to export StdAir objects in a JSON format) [286](#)

stdair::BomJSONImport (Utility class to import StdAir objects in a JSON format)	287
stdair::BomKeyManager (Utility class to extract key structures from strings)	288
stdair::BomManager (Utility class for StdAir-based objects)	290
stdair::BomRetriever (Utility class to retrieve StdAir objects)	293
stdair::BomRoot (Class representing the actual attributes for the Bom root)	302
stdair::BomRootKey (Key of the BOM structure root)	307
stdair_test::BookingClass	309
stdair::BookingClass	310
stdair::BookingClassKey	324
stdair::BookingRequestStruct (Structure holding the elements of a booking request)	326
stdair::Bucket (Class representing the actual attributes for an airline booking class)	332
stdair::BucketKey (Key of booking-class)	337
stdair_test::Cabin	340
stdair::CancellationStruct (Structure holding the elements of a travel solution)	341
stdair::CmdAbstract	344
stdair::CmdBomManager	344
stdair::CmdBomSerialiser	345
stdair::CodeConversionException	345
stdair::CodeDuplicationException	346
COMMAND	348
stdair::ContinuousAttributeLite< T > (Class modeling the distribution of values that can be taken by a continuous attribute)	349
stdair::date_time_element< MIN, MAX >	351
stdair::DatePeriod (Class representing the actual attributes for a fare date-period)	352

stdair::DatePeriodKey (Key of date-period)	357
stdair::DbAbstract	359
stdair::DBManagerForAirlines	360
stdair::DBSessionManager	361
stdair::DefaultDCPList	362
stdair::DefaultDtdFratMap	363
stdair::DefaultDtdProbMap	363
stdair::DemandGenerationMethod (Enumeration of demand (booking request) generation methods)	364
stdair::DictionaryManager (Class wrapper of dictionary business methods)	368
stdair::DocumentNotFoundException	369
stdair::DoWStruct	370
stdair::EventException	374
stdair::EventQueue (Class holding event structures)	375
stdair::EventQueueException	387
stdair::EventQueueKey	388
stdair::EventStruct	390
stdair::EventType	395
stdair::FacAbstract	398
stdair::FacBom< BOM > (Base class for Factory layer)	399
stdair::FacBomManager (Utility class for linking StdAir-based objects)	401
stdair::FacServiceAbstract	406
stdair::FacSTDAIRServiceContext (Factory for Bucket)	408
stdair::FacSupervisor	410
stdair::FareFamily (Class representing the actual attributes for a family fare)	413
stdair::FareFamilyKey (Key of a given fare family, made of a fare family code)	418

stdair::FareFeatures (Class representing the actual attributes for a fare date-period)	420
stdair::FareFeaturesKey (Key of date-period)	426
stdair::FareOptionStruct (Structure holding the elements of a fare option)	428
stdair::FileNotFoundException	432
stdair::FlagSaver	434
stdair::FlightDate (Class representing the actual attributes for an airline flight-date)	434
stdair::FlightDateKey (Key of a given flight-date, made of a flight number and a departure date)	441
stdair::FlightPeriod	443
stdair::FlightPeriodKey	447
FloatingPoint< RawType >	449
stdair::ForecastingMethod	453
stdair::GuillotineBlock (Class representing the actual attributes for an airline guillotine-block)	457
stdair::GuillotineBlockKey (Key of a given guillotine block, made of a guillotine number)	467
stdair::InputFilePath	470
stdair::Inventory (Class representing the actual attributes for an airline inventory)	471
stdair::InventoryKey (Key of a given inventory, made of the airline code)	476
Note that that key allows to differentiate two objects at the same level only. For instance, the segment-date key allows to differentiate two segment-dates under a given flight-date, but does not allow to differentiate two segemnt-dates in general)	
	479
stdair::KeyNotFoundException	482
stdair::LegCabin (Class representing the actual attributes for an airline leg-cabin)	484
stdair::LegCabinKey (Key of a given leg-cabin, made of a cabin code (only))	499
stdair::LegDate	502
stdair::LegDateKey	511

stdair::Logger	513
stdair::MemoryAllocationException	514
stdair::NonInitialisedContainerException	515
stdair::NonInitialisedDBSessionManagerException	517
stdair::NonInitialisedLogServiceException	518
stdair::NonInitialisedRelationShipException	519
stdair::NonInitialisedServiceException	520
stdair::ObjectCreationgDuplicationException	522
stdair::ObjectLinkingException	523
stdair::ObjectNotFoundException	524
stdair::OnDDate (Class representing the actual attributes for an airline flight-date)	526
stdair::OnDDateKey (Key of a given O&D-date, made of a list of OnD strings. a OnD string contains the airline code, the flight number, the date and the segment (origin and destination))	532
stdair::OptimisationNotificationStruct	535
stdair::ParsedKey	539
stdair::ParserException	543
stdair::ParsingFileFailedException	544
stdair::PartnershipTechnique (Enumeration of partnership techniques)	545
stdair::PassengerType	550
stdair::PeriodStruct	553
stdair::PosChannel (Class representing the actual attributes for a fare point of sale)	556
stdair::PosChannelKey (Key of point of sale and channel)	560
stdair::ProgressStatus	563
stdair::ProgressStatusSet	567
stdair::RandomGeneration (Class holding a random generator)	570
stdair::RMEventStruct	574

stdair::RootException (Root of the stdair exceptions)	577
stdair::RootFilePath (Root of the input and output files)	579
stdair::SampleType (Enumeration of BOM sample types)	580
stdair::SegmentCabin (Class representing the actual attributes for an air-line segment-cabin)	584
stdair::SegmentCabinKey (Key of a given segment-cabin, made of a cabin code (only))	595
stdair::SegmentDate (Class representing the actual attributes for an air-line segment-date)	597
stdair::SegmentDateKey (Key of a given segment-date, made of an origin and a destination airports)	606
stdair::SegmentPeriod	609
stdair::SegmentPeriodKey	616
stdair::SerialisationException	618
stdair::ServiceAbstract	619
stdair::ServiceInitialisationType (Enumeration of service initialisation types)	621
swift::SKeymap (The readline keymap wrapper)	625
stdair::SnapshotStruct	627
stdair::SQLDatabaseConnectionImpossibleException	629
stdair::SQLDatabaseException	631
swift::SReadline (The readline library wrapper)	632
stdair::STDAIR_Service (Interface for the STDAIR Services)	637
stdair::STDAIR_ServiceContext (Class holding the context of the Stdair services)	649
stdair::StructAbstract (Base class for the light structures)	650
stdair::TimePeriod (Class representing the actual attributes for a fare time-period)	653
stdair::TimePeriodKey (Key of time-period)	657
stdair::TravelSolutionStruct (Structure holding the elements of a travel solution)	659

soci::type_conversion< stdair::AirlineStruct >	664
TypeWithSize< size >	665
TypeWithSize< 4 >	665
TypeWithSize< 8 >	666
stdair::VirtualClassStruct	667
stdair::YieldFeatures (Class representing the actual attributes for a yield date-period)	670
stdair::YieldFeaturesKey (Key of date-period)	674
stdair::YieldRange	676
stdair::YieldStore	679
stdair::YieldStoreKey	683

30 File Index

30.1 File List

Here is a list of all files with brief descriptions:

batches/stdair.cpp	685
stdair/stdair_basic_types.hpp	1218
stdair/stdair_date_time_types.hpp	1220
stdair/stdair_db.hpp	1221
stdair/stdair_demand_types.hpp	1223
stdair/stdair_event_types.hpp	1224
stdair/stdair_exceptions.hpp	1226
stdair/stdair_fare_types.hpp	1228
stdair/stdair_file.hpp	1229
stdair/stdair_inventory_types.hpp	1232
stdair/stdair_log.hpp	1234
stdair/stdair_maths_types.hpp	1236

stdair/stdair_rm_types.hpp	1237
stdair/STDAIR_Service.hpp	1238
stdair/stdair_service_types.hpp	1241
stdair/stdair_types.hpp	1242
stdair/basic/BasChronometer.cpp	690
stdair/basic/BasChronometer.hpp	692
stdair/basic/BasConst.cpp	696
stdair/basic/BasConst_BomDisplay.hpp	704
stdair/basic/BasConst_BookingClass.hpp	705
stdair/basic/BasConst_DefaultObject.hpp	707
stdair/basic/BasConst_Event.hpp	709
stdair/basic/BasConst_General.hpp	710
stdair/basic/BasConst_Inventory.hpp	712
stdair/basic/BasConst_Period_BOM.hpp	713
stdair/basic/BasConst_Request.hpp	715
stdair/basic/BasConst_TravelSolution.hpp	716
stdair/basic/BasConst_Yield.hpp	717
stdair/basic/BasDBParams.cpp	718
stdair/basic/BasDBParams.hpp	719
stdair/basic/BasFileMgr.cpp	721
stdair/basic/BasFileMgr.hpp	722
stdair/basic/BasLogParams.cpp	723
stdair/basic/BasLogParams.hpp	724
stdair/basic/BasParserHelperTypes.hpp	727
stdair/basic/BasParserTypes.hpp	728
stdair/basic/BasTypes.hpp	729
stdair/basic/ContinuousAttributeLite.hpp	730

stdair/basic/DemandGenerationMethod.cpp	734
stdair/basic/DemandGenerationMethod.hpp	737
stdair/basic/DictionaryManager.cpp	739
stdair/basic/DictionaryManager.hpp	740
stdair/basic/EventType.cpp	740
stdair/basic/EventType.hpp	743
stdair/basic/float_utils.hpp	744
stdair/basic/float_utils_google.hpp	744
stdair/basic/ForecastingMethod.cpp	749
stdair/basic/ForecastingMethod.hpp	751
stdair/basic/PartnershipTechnique.cpp	753
stdair/basic/PartnershipTechnique.hpp	756
stdair/basic/PassengerType.cpp	757
stdair/basic/PassengerType.hpp	759
stdair/basic/ProgressStatus.cpp	760
stdair/basic/ProgressStatus.hpp	762
stdair/basic/ProgressStatusSet.cpp	764
stdair/basic/ProgressStatusSet.hpp	766
stdair/basic/RandomGeneration.cpp	767
stdair/basic/RandomGeneration.hpp	769
stdair/basic/SampleType.cpp	771
stdair/basic/SampleType.hpp	773
stdair/basic/ServiceInitialisationType.cpp	774
stdair/basic/ServiceInitialisationType.hpp	777
stdair/basic/StructAbstract.hpp	779
stdair/basic/YieldRange.cpp	781
stdair/basic/YieldRange.hpp	782

stdair/bom/ AirlineClassList.cpp	784
stdair/bom/ AirlineClassList.hpp	785
stdair/bom/ AirlineClassListKey.cpp	788
stdair/bom/ AirlineClassListKey.hpp	790
stdair/bom/ AirlineClassListTypes.hpp	792
stdair/bom/ AirlineFeature.cpp	793
stdair/bom/ AirlineFeature.hpp	794
stdair/bom/ AirlineFeatureKey.cpp	795
stdair/bom/ AirlineFeatureKey.hpp	796
stdair/bom/ AirlineFeatureTypes.hpp	798
stdair/bom/ AirlineStruct.cpp	798
stdair/bom/ AirlineStruct.hpp	800
stdair/bom/ AirportPair.cpp	801
stdair/bom/ AirportPair.hpp	802
stdair/bom/ AirportPairKey.cpp	804
stdair/bom/ AirportPairKey.hpp	805
stdair/bom/ AirportPairTypes.hpp	807
stdair/bom/ BomAbstract.hpp	808
stdair/bom/ BomArchive.cpp	810
stdair/bom/ BomArchive.hpp	812
stdair/bom/ BomDisplay.cpp	812
stdair/bom/ BomDisplay.hpp	831
stdair/bom/ BomHolder.hpp	833
stdair/bom/ BomHolderKey.cpp	835
stdair/bom/ BomHolderKey.hpp	836
stdair/bom/ BomJSONExport.cpp	837
stdair/bom/ BomJSONExport.hpp	845

stdair/bom/BomJSONImport.cpp	846
stdair/bom/BomJSONImport.hpp	848
stdair/bom/BomKeyManager.cpp	849
stdair/bom/BomKeyManager.hpp	851
stdair/bom/BomManager.hpp	852
stdair/bom/BomRetriever.cpp	858
stdair/bom/BomRetriever.hpp	866
stdair/bom/BomRoot.cpp	868
stdair/bom/BomRoot.hpp	870
stdair/bom/BomRootKey.cpp	872
stdair/bom/BomRootKey.hpp	874
stdair/bom/BookingClass.cpp	875
stdair/bom/BookingClass.hpp	877
stdair/bom/BookingClassKey.cpp	882
stdair/bom/BookingClassKey.hpp	883
stdair/bom/BookingClassTypes.hpp	884
stdair/bom/BookingRequestStruct.cpp	885
stdair/bom/BookingRequestStruct.hpp	889
stdair/bom/BookingRequestTypes.hpp	892
stdair/bom/Bucket.cpp	893
stdair/bom/Bucket.hpp	895
stdair/bom/BucketKey.cpp	897
stdair/bom/BucketKey.hpp	899
stdair/bom/BucketTypes.hpp	901
stdair/bom/CancellationStruct.cpp	902
stdair/bom/CancellationStruct.hpp	903
stdair/bom/CancellationTypes.hpp	905

stdair/bom/DatePeriod.cpp	906
stdair/bom/DatePeriod.hpp	907
stdair/bom/DatePeriodKey.cpp	909
stdair/bom/DatePeriodKey.hpp	910
stdair/bom/DatePeriodTypes.hpp	911
stdair/bom/DoWStruct.cpp	912
stdair/bom/DoWStruct.hpp	915
stdair/bom/EventQueue.cpp	916
stdair/bom/EventQueue.hpp	922
stdair/bom/EventQueueKey.cpp	926
stdair/bom/EventQueueKey.hpp	927
stdair/bom/EventQueueTypes.hpp	928
stdair/bom/EventStruct.cpp	929
stdair/bom/EventStruct.hpp	933
stdair/bom/EventTypes.hpp	936
stdair/bom/FareFamily.cpp	937
stdair/bom/FareFamily.hpp	938
stdair/bom/FareFamilyKey.cpp	941
stdair/bom/FareFamilyKey.hpp	942
stdair/bom/FareFamilyTypes.hpp	944
stdair/bom/FareFeatures.cpp	945
stdair/bom/FareFeatures.hpp	947
stdair/bom/FareFeaturesKey.cpp	949
stdair/bom/FareFeaturesKey.hpp	951
stdair/bom/FareFeaturesTypes.hpp	953
stdair/bom/FareOptionStruct.cpp	953
stdair/bom/FareOptionStruct.hpp	956

stdair/bom/FareOptionTypes.hpp	958
stdair/bom/FlightDate.cpp	959
stdair/bom/FlightDate.hpp	961
stdair/bom/FlightDateKey.cpp	963
stdair/bom/FlightDateKey.hpp	965
stdair/bom/FlightDateTypes.hpp	967
stdair/bom/FlightPeriod.cpp	968
stdair/bom/FlightPeriod.hpp	969
stdair/bom/FlightPeriodKey.cpp	970
stdair/bom/FlightPeriodKey.hpp	971
stdair/bom/FlightPeriodTypes.hpp	972
stdair/bom/GuillotineBlock.cpp	973
stdair/bom/GuillotineBlock.hpp	979
stdair/bom/GuillotineBlockKey.cpp	983
stdair/bom/GuillotineBlockKey.hpp	985
stdair/bom/GuillotineBlockTypes.hpp	987
stdair/bom/Inventory.cpp	988
stdair/bom/Inventory.hpp	989
stdair/bom/InventoryKey.cpp	992
stdair/bom/InventoryKey.hpp	994
stdair/bom/InventoryTypes.hpp	995
stdair/bom/key_types.hpp	996
stdair/bom/KeyAbstract.hpp	998
stdair/bom/LegCabin.cpp	999
stdair/bom/LegCabin.hpp	1002
stdair/bom/LegCabinKey.cpp	1008
stdair/bom/LegCabinKey.hpp	1009

stdair/bom/LegCabinTypes.hpp	1011
stdair/bom/LegDate.cpp	1012
stdair/bom/LegDate.hpp	1014
stdair/bom/LegDateKey.cpp	1017
stdair/bom/LegDateKey.hpp	1018
stdair/bom/LegDateTypes.hpp	1020
stdair/bom/OnDDate.cpp	1020
stdair/bom/OnDDate.hpp	1023
stdair/bom/OnDDateKey.cpp	1026
stdair/bom/OnDDateKey.hpp	1028
stdair/bom/OnDDateTypes.hpp	1030
stdair/bom/OptimisationNotificationStruct.cpp	1031
stdair/bom/OptimisationNotificationStruct.hpp	1033
stdair/bom/OptimisationNotificationTypes.hpp	1036
stdair/bom/ParsedKey.cpp	1037
stdair/bom/ParsedKey.hpp	1040
stdair/bom/PeriodStruct.cpp	1041
stdair/bom/PeriodStruct.hpp	1043
stdair/bom/PosChannel.cpp	1044
stdair/bom/PosChannel.hpp	1045
stdair/bom/PosChannelKey.cpp	1047
stdair/bom/PosChannelKey.hpp	1048
stdair/bom/PosChannelTypes.hpp	1050
stdair/bom/RMEventStruct.cpp	1050
stdair/bom/RMEventStruct.hpp	1052
stdair/bom/RMEventTypes.hpp	1053
stdair/bom/SegmentCabin.cpp	1054

stdair/bom/SegmentCabin.hpp	1056
stdair/bom/SegmentCabinKey.cpp	1060
stdair/bom/SegmentCabinKey.hpp	1062
stdair/bom/SegmentCabinTypes.hpp	1064
stdair/bom/SegmentDate.cpp	1064
stdair/bom/SegmentDate.hpp	1066
stdair/bom/SegmentDateKey.cpp	1070
stdair/bom/SegmentDateKey.hpp	1072
stdair/bom/SegmentDateTypes.hpp	1073
stdair/bom/SegmentPeriod.cpp	1074
stdair/bom/SegmentPeriod.hpp	1075
stdair/bom/SegmentPeriodKey.cpp	1077
stdair/bom/SegmentPeriodKey.hpp	1078
stdair/bom/SegmentPeriodTypes.hpp	1080
stdair/bom/SnapshotStruct.cpp	1080
stdair/bom/SnapshotStruct.hpp	1082
stdair/bom/SnapshotTypes.hpp	1083
stdair/bom/TimePeriod.cpp	1084
stdair/bom/TimePeriod.hpp	1085
stdair/bom/TimePeriodKey.cpp	1087
stdair/bom/TimePeriodKey.hpp	1088
stdair/bom/TimePeriodTypes.hpp	1090
stdair/bom/TravelSolutionStruct.cpp	1091
stdair/bom/TravelSolutionStruct.hpp	1094
stdair/bom/TravelSolutionTypes.hpp	1096
stdair/bom/VirtualClassStruct.cpp	1098
stdair/bom/VirtualClassStruct.hpp	1099

stdair/bom/VirtualClassTypes.hpp	1101
stdair/bom/YieldFeatures.cpp	1102
stdair/bom/YieldFeatures.hpp	1104
stdair/bom/YieldFeaturesKey.cpp	1105
stdair/bom/YieldFeaturesKey.hpp	1107
stdair/bom/YieldFeaturesTypes.hpp	1108
stdair/bom/YieldStore.cpp	1109
stdair/bom/YieldStore.hpp	1110
stdair/bom/YieldStoreKey.cpp	1111
stdair/bom/YieldStoreKey.hpp	1112
stdair/bom/YieldStoreTypes.hpp	1113
stdair/command/CmdAbstract.cpp	1114
stdair/command/CmdAbstract.hpp	1114
stdair/command/CmdBomManager.cpp	1115
stdair/command/CmdBomManager.hpp	1157
stdair/command/CmdBomSerialiser.cpp	1160
stdair/command/CmdBomSerialiser.hpp	1164
stdair/command/DBManagerForAirlines.cpp	1165
stdair/command/DBManagerForAirlines.hpp	1167
stdair/config/stdair-paths.hpp	1170
stdair/dbadaptor/DbAbstract.cpp	1171
stdair/dbadaptor/DbAbstract.hpp	1172
stdair/dbadaptor/DbAirline.cpp	1173
stdair/dbadaptor/DbAirline.hpp	1175
stdair/factory/FacAbstract.cpp	1175
stdair/factory/FacAbstract.hpp	1176
stdair/factory/FacBom.hpp	1177

stdair/factory/FacBomManager.hpp	1179
stdair/service/DBSessionManager.cpp	1185
stdair/service/DBSessionManager.hpp	1187
stdair/service/FacServiceAbstract.cpp	1188
stdair/service/FacServiceAbstract.hpp	1189
stdair/service/FacSTDAIRServiceContext.cpp	1190
stdair/service/FacSTDAIRServiceContext.hpp	1191
stdair/service/FacSupervisor.cpp	1192
stdair/service/FacSupervisor.hpp	1194
stdair/service/Logger.cpp	1195
stdair/service/Logger.hpp	1198
stdair/service/ServiceAbstract.cpp	1200
stdair/service/ServiceAbstract.hpp	1201
stdair/service/STDAIR_Service.cpp	1203
stdair/service/STDAIR_ServiceContext.cpp	1213
stdair/service/STDAIR_ServiceContext.hpp	1215
stdair/ui/cmdline/readline_autocomp.hpp	1247
stdair/ui/cmdline/SReadline.hpp (C++ wrapper around libreadline)	1253
test/stdair/MPBomRoot.cpp	1260
test/stdair/MPBomRoot.hpp	1261
test/stdair/MPInventory.cpp	1262
test/stdair/MPInventory.hpp	1262
test/stdair/StandardAirlineITTestSuite.cpp	1263
test/stdair/StdairTestLib.hpp	1268

31 Module Documentation

31.1 Abstract part of the Business Object Model (BOM)

Author

Anh Quan Nguyen <quannaus@users.sourceforge.net>

Date

20/01/2010

31.2 Part of the Business Object Model (BOM) handling

(hash-like)keys

Author

Anh Quan Nguyen <quannaus@users.sourceforge.net>

Date

20/01/2010

32 Directory Documentation

32.1 stdair/basic/ Directory Reference

Files

- file [BasChronometer.cpp](#)
- file [BasChronometer.hpp](#)
- file [BasConst.cpp](#)
- file [BasConst_BomDisplay.hpp](#)
- file [BasConst_BookingClass.hpp](#)
- file [BasConst_DefaultObject.hpp](#)
- file [BasConst_Event.hpp](#)
- file [BasConst_General.hpp](#)
- file [BasConst_Inventory.hpp](#)
- file [BasConst_Period_BOM.hpp](#)
- file [BasConst_Request.hpp](#)
- file [BasConst_TravelSolution.hpp](#)
- file [BasConst_Yield.hpp](#)
- file [BasDBParams.cpp](#)
- file [BasDBParams.hpp](#)
- file [BasFileMgr.cpp](#)
- file [BasFileMgr.hpp](#)
- file [BasLogParams.cpp](#)
- file [BasLogParams.hpp](#)
- file [BasParserHelperTypes.hpp](#)

- file [BasParserTypes.hpp](#)
- file [BasTypes.hpp](#)
- file [ContinuousAttributeLite.hpp](#)
- file [DemandGenerationMethod.cpp](#)
- file [DemandGenerationMethod.hpp](#)
- file [DictionaryManager.cpp](#)
- file [DictionaryManager.hpp](#)
- file [EventType.cpp](#)
- file [EventType.hpp](#)
- file [float_utils.hpp](#)
- file [float_utils_google.hpp](#)
- file [ForecastingMethod.cpp](#)
- file [ForecastingMethod.hpp](#)
- file [PartnershipTechnique.cpp](#)
- file [PartnershipTechnique.hpp](#)
- file [PassengerType.cpp](#)
- file [PassengerType.hpp](#)
- file [ProgressStatus.cpp](#)
- file [ProgressStatus.hpp](#)
- file [ProgressStatusSet.cpp](#)
- file [ProgressStatusSet.hpp](#)
- file [RandomGeneration.cpp](#)
- file [RandomGeneration.hpp](#)
- file [SampleType.cpp](#)
- file [SampleType.hpp](#)
- file [ServiceInitialisationType.cpp](#)
- file [ServiceInitialisationType.hpp](#)
- file [StructAbstract.hpp](#)
- file [YieldRange.cpp](#)
- file [YieldRange.hpp](#)

32.2 batches/ Directory Reference

Files

- file [stdair.cpp](#)

32.3 stdair/bom/ Directory Reference

Files

- file [AirlineClassList.cpp](#)
- file [AirlineClassList.hpp](#)
- file [AirlineClassListKey.cpp](#)
- file [AirlineClassListKey.hpp](#)

- file [AirlineClassListTypes.hpp](#)
- file [AirlineFeature.cpp](#)
- file [AirlineFeature.hpp](#)
- file [AirlineFeatureKey.cpp](#)
- file [AirlineFeatureKey.hpp](#)
- file [AirlineFeatureTypes.hpp](#)
- file [AirlineStruct.cpp](#)
- file [AirlineStruct.hpp](#)
- file [AirportPair.cpp](#)
- file [AirportPair.hpp](#)
- file [AirportPairKey.cpp](#)
- file [AirportPairKey.hpp](#)
- file [AirportPairTypes.hpp](#)
- file [BomAbstract.hpp](#)
- file [BomArchive.cpp](#)
- file [BomArchive.hpp](#)
- file [BomDisplay.cpp](#)
- file [BomDisplay.hpp](#)
- file [BomHolder.hpp](#)
- file [BomHolderKey.cpp](#)
- file [BomHolderKey.hpp](#)
- file [BomJSONExport.cpp](#)
- file [BomJSONExport.hpp](#)
- file [BomJSONImport.cpp](#)
- file [BomJSONImport.hpp](#)
- file [BomKeyManager.cpp](#)
- file [BomKeyManager.hpp](#)
- file [BomManager.hpp](#)
- file [BomRetriever.cpp](#)
- file [BomRetriever.hpp](#)
- file [BomRoot.cpp](#)
- file [BomRoot.hpp](#)
- file [BomRootKey.cpp](#)
- file [BomRootKey.hpp](#)
- file [BookingClass.cpp](#)
- file [BookingClass.hpp](#)
- file [BookingClassKey.cpp](#)
- file [BookingClassKey.hpp](#)
- file [BookingClassTypes.hpp](#)
- file [BookingRequestStruct.cpp](#)
- file [BookingRequestStruct.hpp](#)
- file [BookingRequestTypes.hpp](#)
- file [Bucket.cpp](#)
- file [Bucket.hpp](#)
- file [BucketKey.cpp](#)
- file [BucketKey.hpp](#)

- file [BucketTypes.hpp](#)
- file [CancellationStruct.cpp](#)
- file [CancellationStruct.hpp](#)
- file [CancellationTypes.hpp](#)
- file [DatePeriod.cpp](#)
- file [DatePeriod.hpp](#)
- file [DatePeriodKey.cpp](#)
- file [DatePeriodKey.hpp](#)
- file [DatePeriodTypes.hpp](#)
- file [DoWStruct.cpp](#)
- file [DoWStruct.hpp](#)
- file [EventQueue.cpp](#)
- file [EventQueue.hpp](#)
- file [EventQueueKey.cpp](#)
- file [EventQueueKey.hpp](#)
- file [EventQueueTypes.hpp](#)
- file [EventStruct.cpp](#)
- file [EventStruct.hpp](#)
- file [EventTypes.hpp](#)
- file [FareFamily.cpp](#)
- file [FareFamily.hpp](#)
- file [FareFamilyKey.cpp](#)
- file [FareFamilyKey.hpp](#)
- file [FareFamilyTypes.hpp](#)
- file [FareFeatures.cpp](#)
- file [FareFeatures.hpp](#)
- file [FareFeaturesKey.cpp](#)
- file [FareFeaturesKey.hpp](#)
- file [FareFeaturesTypes.hpp](#)
- file [FareOptionStruct.cpp](#)
- file [FareOptionStruct.hpp](#)
- file [FareOptionTypes.hpp](#)
- file [FlightDate.cpp](#)
- file [FlightDate.hpp](#)
- file [FlightDateKey.cpp](#)
- file [FlightDateKey.hpp](#)
- file [FlightDateTypes.hpp](#)
- file [FlightPeriod.cpp](#)
- file [FlightPeriod.hpp](#)
- file [FlightPeriodKey.cpp](#)
- file [FlightPeriodKey.hpp](#)
- file [FlightPeriodTypes.hpp](#)
- file [GuillotineBlock.cpp](#)
- file [GuillotineBlock.hpp](#)
- file [GuillotineBlockKey.cpp](#)
- file [GuillotineBlockKey.hpp](#)

- file [GuillotineBlockTypes.hpp](#)
- file [Inventory.cpp](#)
- file [Inventory.hpp](#)
- file [InventoryKey.cpp](#)
- file [InventoryKey.hpp](#)
- file [InventoryTypes.hpp](#)
- file [key_types.hpp](#)
- file [KeyAbstract.hpp](#)
- file [LegCabin.cpp](#)
- file [LegCabin.hpp](#)
- file [LegCabinKey.cpp](#)
- file [LegCabinKey.hpp](#)
- file [LegCabinTypes.hpp](#)
- file [LegDate.cpp](#)
- file [LegDate.hpp](#)
- file [LegDateKey.cpp](#)
- file [LegDateKey.hpp](#)
- file [LegDateTypes.hpp](#)
- file [OnDDate.cpp](#)
- file [OnDDate.hpp](#)
- file [OnDDateKey.cpp](#)
- file [OnDDateKey.hpp](#)
- file [OnDDateTypes.hpp](#)
- file [OptimisationNotificationStruct.cpp](#)
- file [OptimisationNotificationStruct.hpp](#)
- file [OptimisationNotificationTypes.hpp](#)
- file [ParsedKey.cpp](#)
- file [ParsedKey.hpp](#)
- file [PeriodStruct.cpp](#)
- file [PeriodStruct.hpp](#)
- file [PosChannel.cpp](#)
- file [PosChannel.hpp](#)
- file [PosChannelKey.cpp](#)
- file [PosChannelKey.hpp](#)
- file [PosChannelTypes.hpp](#)
- file [RMEventStruct.cpp](#)
- file [RMEventStruct.hpp](#)
- file [RMEventTypes.hpp](#)
- file [SegmentCabin.cpp](#)
- file [SegmentCabin.hpp](#)
- file [SegmentCabinKey.cpp](#)
- file [SegmentCabinKey.hpp](#)
- file [SegmentCabinTypes.hpp](#)
- file [SegmentDate.cpp](#)
- file [SegmentDate.hpp](#)
- file [SegmentDateKey.cpp](#)

- file [SegmentDateKey.hpp](#)
- file [SegmentDateTypes.hpp](#)
- file [SegmentPeriod.cpp](#)
- file [SegmentPeriod.hpp](#)
- file [SegmentPeriodKey.cpp](#)
- file [SegmentPeriodKey.hpp](#)
- file [SegmentPeriodTypes.hpp](#)
- file [SnapshotStruct.cpp](#)
- file [SnapshotStruct.hpp](#)
- file [SnapshotTypes.hpp](#)
- file [TimePeriod.cpp](#)
- file [TimePeriod.hpp](#)
- file [TimePeriodKey.cpp](#)
- file [TimePeriodKey.hpp](#)
- file [TimePeriodTypes.hpp](#)
- file [TravelSolutionStruct.cpp](#)
- file [TravelSolutionStruct.hpp](#)
- file [TravelSolutionTypes.hpp](#)
- file [VirtualClassStruct.cpp](#)
- file [VirtualClassStruct.hpp](#)
- file [VirtualClassTypes.hpp](#)
- file [YieldFeatures.cpp](#)
- file [YieldFeatures.hpp](#)
- file [YieldFeaturesKey.cpp](#)
- file [YieldFeaturesKey.hpp](#)
- file [YieldFeaturesTypes.hpp](#)
- file [YieldStore.cpp](#)
- file [YieldStore.hpp](#)
- file [YieldStoreKey.cpp](#)
- file [YieldStoreKey.hpp](#)
- file [YieldStoreTypes.hpp](#)

32.4 stdair/ui/cmdline/ Directory Reference

Files

- file [readline_autocomp.hpp](#)
- file [SReadline.hpp](#)

C++ wrapper around libreadline.

32.5 stdair/command/ Directory Reference

Files

- file [CmdAbstract.cpp](#)
- file [CmdAbstract.hpp](#)
- file [CmdBomManager.cpp](#)
- file [CmdBomManager.hpp](#)
- file [CmdBomSerialiser.cpp](#)
- file [CmdBomSerialiser.hpp](#)
- file [DBManagerForAirlines.cpp](#)
- file [DBManagerForAirlines.hpp](#)

32.6 stdair/config/ Directory Reference

Files

- file [stdair-paths.hpp](#)

32.7 stdair/dbadaptor/ Directory Reference

Files

- file [DbAbstract.cpp](#)
- file [DbAbstract.hpp](#)
- file [DbAirline.cpp](#)
- file [DbAirline.hpp](#)

32.8 stdair/factory/ Directory Reference

Files

- file [FacAbstract.cpp](#)
- file [FacAbstract.hpp](#)
- file [FacBom.hpp](#)
- file [FacBomManager.hpp](#)

32.9 stdair/service/ Directory Reference

Files

- file [DBSessionManager.cpp](#)
- file [DBSessionManager.hpp](#)
- file [FacServiceAbstract.cpp](#)
- file [FacServiceAbstract.hpp](#)

- file [FacSTDAIRServiceContext.cpp](#)
- file [FacSTDAIRServiceContext.hpp](#)
- file [FacSupervisor.cpp](#)
- file [FacSupervisor.hpp](#)
- file [Logger.cpp](#)
- file [Logger.hpp](#)
- file [ServiceAbstract.cpp](#)
- file [ServiceAbstract.hpp](#)
- file [STDAIR_Service.cpp](#)
- file [STDAIR_ServiceContext.cpp](#)
- file [STDAIR_ServiceContext.hpp](#)

32.10 test/stdair/ Directory Reference

Files

- file [MPBomRoot.cpp](#)
- file [MPBomRoot.hpp](#)
- file [MPIInventory.cpp](#)
- file [MPIInventory.hpp](#)
- file [StandardAirlineITTestSuite.cpp](#)
- file [StdairTestLib.hpp](#)

32.11 stdair/ Directory Reference

Directories

- directory [basic](#)
- directory [bom](#)
- directory [command](#)
- directory [config](#)
- directory [dbadaptor](#)
- directory [factory](#)
- directory [service](#)
- directory [ui](#)

Files

- file [stdair_basic_types.hpp](#)
- file [stdair_date_time_types.hpp](#)
- file [stdair_db.hpp](#)
- file [stdair_demand_types.hpp](#)
- file [stdair_event_types.hpp](#)
- file [stdair_exceptions.hpp](#)
- file [stdair_fare_types.hpp](#)

- file [stdair_file.hpp](#)
- file [stdair_inventory_types.hpp](#)
- file [stdair_log.hpp](#)
- file [stdair_maths_types.hpp](#)
- file [stdair_rm_types.hpp](#)
- file [STDAIR_Service.hpp](#)
- file [stdair_service_types.hpp](#)
- file [stdair_types.hpp](#)

32.12 test/ Directory Reference

Directories

- directory [stdair](#)

32.13 stdair/ui/ Directory Reference

Directories

- directory [cmdline](#)

33 Namespace Documentation

33.1 boost Namespace Reference

Forward declarations.

Namespaces

- namespace [serialization](#)

33.1.1 Detailed Description

Forward declarations.

33.2 boost::serialization Namespace Reference

33.3 bpt Namespace Reference

Typedefs

- typedef char [ptree](#)

33.3.1 Typedef Documentation

33.3.1.1 typedef char bpt::ptree

Definition at line 35 of file [BomJSONExport.cpp](#).

33.4 soci Namespace Reference

Classes

- struct [type_conversion< stdair::AirlineStruct >](#)

33.5 stdair Namespace Reference

Handle on the StdAir library context.

Namespaces

- namespace [LOG](#)

Classes

- struct [BasChronometer](#)
- struct [DefaultDCPList](#)
- struct [DefaultDtdFratMap](#)
- struct [DefaultDtdProbMap](#)
- struct [BasDBParams](#)
Structure holding the parameters for connection to a database.
- struct [BasFileMgr](#)
- struct [BasLogParams](#)
Structure holding parameters for logging.
- struct [date_time_element](#)
- struct [ContinuousAttributeLite](#)
Class modeling the distribution of values that can be taken by a continuous attribute.
- struct [DemandGenerationMethod](#)
Enumeration of demand (booking request) generation methods.
- class [DictionaryManager](#)
Class wrapper of dictionary business methods.
- struct [EventType](#)
- struct [ForecastingMethod](#)
- struct [PartnershipTechnique](#)
Enumeration of partnership techniques.
- struct [PassengerType](#)
- struct [ProgressStatus](#)

- struct [ProgressStatusSet](#)
- struct [RandomGeneration](#)
Class holding a random generator.
- struct [SampleType](#)
Enumeration of BOM sample types.
- struct [ServiceInitialisationType](#)
Enumeration of service initialisation types.
- struct [StructAbstract](#)
Base class for the light structures.
- class [YieldRange](#)
- class [AirlineClassList](#)
Class representing the actual attributes for a segment-features.
- struct [AirlineClassListKey](#)
Key of airport-pair.
- class [AirlineFeature](#)
- struct [AirlineFeatureKey](#)
- struct [AirlineStruct](#)
- class [AirportPair](#)
Class representing the actual attributes for an airport-pair.
- struct [AirportPairKey](#)
Key of airport-pair.
- class [BomAbstract](#)
Base class for the Business Object Model (BOM) layer.
- class [BomArchive](#)
Utility class to archive/restore BOM objects with Boost serialisation.
- class [BomDisplay](#)
Utility class to display StdAir objects with a pretty format.
- class [BomHolder](#)
Class representing the holder of BOM object containers (list and map).
- struct [BomHolderKey](#)
- struct [FlagSaver](#)
- class [BomJSONExport](#)
Utility class to export StdAir objects in a JSON format.
- class [BomJSONImport](#)
Utility class to import StdAir objects in a JSON format.
- class [BomKeyManager](#)
Utility class to extract key structures from strings.
- class [BomManager](#)
Utility class for StdAir-based objects.
- class [BomRetriever](#)
Utility class to retrieve StdAir objects.
- class [BomRoot](#)
Class representing the actual attributes for the Bom root.
- struct [BomRootKey](#)

Key of the BOM structure root.

- class [BookingClass](#)
- struct [BookingClassKey](#)
- struct [BookingRequestStruct](#)

Structure holding the elements of a booking request.

- class [Bucket](#)

Class representing the actual attributes for an airline booking class.

- struct [BucketKey](#)

Key of booking-class.

- struct [CancellationStruct](#)

Structure holding the elements of a travel solution.

- class [DatePeriod](#)

Class representing the actual attributes for a fare date-period.

- struct [DatePeriodKey](#)

Key of date-period.

- struct [DoWStruct](#)
- class [EventQueue](#)

Class holding event structures.

- struct [EventQueueKey](#)
- struct [EventStruct](#)
- class [FareFamily](#)

Class representing the actual attributes for a family fare.

- struct [FareFamilyKey](#)

Key of a given fare family, made of a fare family code.

- class [FareFeatures](#)

Class representing the actual attributes for a fare date-period.

- struct [FareFeaturesKey](#)

Key of date-period.

- struct [FareOptionStruct](#)

Structure holding the elements of a fare option.

- class [FlightDate](#)

Class representing the actual attributes for an airline flight-date.

- struct [FlightDateKey](#)

Key of a given flight-date, made of a flight number and a departure date.

- class [FlightPeriod](#)
- struct [FlightPeriodKey](#)
- class [GuillotineBlock](#)

Class representing the actual attributes for an airline guillotine-block.

- struct [GuillotineBlockKey](#)

Key of a given guillotine block, made of a guillotine number.

- class [Inventory](#)

Class representing the actual attributes for an airline inventory.

- struct [InventoryKey](#)

Key of a given inventory, made of the airline code.

- struct [KeyAbstract](#)
*Base class for the keys of Business Object Model (BOM) layer.
Note that that key allows to differentiate two objects at the same level only. For instance, the segment-date key allows to differentiate two segment-dates under a given flight-date, but does not allow to differentiate two segemnt-dates in general.*
- class [LegCabin](#)
Class representing the actual attributes for an airline leg-cabin.
- struct [LegCabinKey](#)
Key of a given leg-cabin, made of a cabin code (only).
- class [LegDate](#)
- struct [LegDateKey](#)
- class [OnDDate](#)
Class representing the actual attributes for an airline flight-date.
- struct [OnDDateKey](#)
Key of a given O&D-date, made of a list of OnD strings. a OnD string contains the airline code, the flight number, the date and the segment (origin and destination).
- struct [OptimisationNotificationStruct](#)
- struct [ParsedKey](#)
- struct [PeriodStruct](#)
- class [PosChannel](#)
Class representing the actual attributes for a fare point of sale.
- struct [PosChannelKey](#)
Key of point of sale and channel.
- struct [RMEventStruct](#)
- class [SegmentCabin](#)
Class representing the actual attributes for an airline segment-cabin.
- struct [SegmentCabinKey](#)
Key of a given segment-cabin, made of a cabin code (only).
- class [SegmentDate](#)
Class representing the actual attributes for an airline segment-date.
- struct [SegmentDateKey](#)
Key of a given segment-date, made of an origin and a destination airports.
- class [SegmentPeriod](#)
- struct [SegmentPeriodKey](#)
- struct [SnapshotStruct](#)
- class [TimePeriod](#)
Class representing the actual attributes for a fare time-period.
- struct [TimePeriodKey](#)
Key of time-period.
- struct [TravelSolutionStruct](#)
Structure holding the elements of a travel solution.
- struct [VirtualClassStruct](#)
- class [YieldFeatures](#)
Class representing the actual attributes for a yield date-period.
- struct [YieldFeaturesKey](#)

Key of date-period.

- class [YieldStore](#)
- struct [YieldStoreKey](#)
- class [CmdAbstract](#)
- class [CmdBomManager](#)
- class [CmdBomSerialiser](#)
- class [DBManagerForAirlines](#)
- class [DbAbstract](#)
- class [FacAbstract](#)
- class [FacBom](#)

Base class for Factory layer.

- class [FacBomManager](#)

Utility class for linking StdAir-based objects.

- class [DBSessionManager](#)
- class [FacServiceAbstract](#)
- class [FacSTDAIRServiceContext](#)

Factory for [Bucket](#).

- class [FacSupervisor](#)
- class [Logger](#)
- class [ServiceAbstract](#)
- class [STDAIR_ServiceContext](#)

Class holding the context of the Stdair services.

- class [RootException](#)

Root of the stdair exceptions.

- class [FileNotFoundException](#)
- class [NonInitialisedLogServiceException](#)
- class [NonInitialisedServiceException](#)
- class [NonInitialisedContainerException](#)
- class [NonInitialisedRelationshipException](#)
- class [MemoryAllocationException](#)
- class [ObjectLinkingException](#)
- class [DocumentNotFoundException](#)
- class [ParserException](#)
- class [SerialisationException](#)
- class [KeyNotFoundException](#)
- class [CodeConversionException](#)
- class [CodeDuplicationException](#)
- class [ObjectCreationDuplicationException](#)
- class [ObjectNotFoundException](#)
- class [ParsingFileFailedException](#)
- class [SQLDatabaseException](#)
- class [NonInitialisedDBSessionManagerException](#)
- class [SQLDatabaseConnectionImpossibleException](#)
- class [EventException](#)
- class [EventQueueException](#)
- class [RootFilePath](#)

Root of the input and output files.

- class [InputFilePath](#)
- class [STDAIR_Service](#)

Interface for the STDAIR Services.

Typedefs

- typedef [date_time_element](#)< 0, 23 > [hour_t](#)
- typedef [date_time_element](#)< 0, 59 > [minute_t](#)
- typedef [date_time_element](#)< 0, 59 > [second_t](#)
- typedef [date_time_element](#)< 1900, 2100 > [year_t](#)
- typedef [date_time_element](#)< 1, 12 > [month_t](#)
- typedef [date_time_element](#)< 1, 31 > [day_t](#)
- typedef std::istreambuf_iterator< char > [base_iterator_t](#)
- typedef boost::spirit::multi_pass< [base_iterator_t](#) > [iterator_t](#)
- typedef boost::spirit::qi::int_parser< unsigned int, 10, 1, 1 > [int1_p_t](#)
- typedef boost::spirit::qi::uint_parser< int, 10, 2, 2 > [uint2_p_t](#)
- typedef boost::spirit::qi::uint_parser< int, 10, 4, 4 > [uint4_p_t](#)
- typedef boost::spirit::qi::uint_parser< int, 10, 1, 4 > [uint1_4_p_t](#)
- typedef boost::spirit::qi::uint_parser< [hour_t](#), 10, 2, 2 > [hour_p_t](#)
- typedef boost::spirit::qi::uint_parser< [minute_t](#), 10, 2, 2 > [minute_p_t](#)
- typedef boost::spirit::qi::uint_parser< [second_t](#), 10, 2, 2 > [second_p_t](#)
- typedef boost::spirit::qi::uint_parser< [year_t](#), 10, 4, 4 > [year_p_t](#)
- typedef boost::spirit::qi::uint_parser< [month_t](#), 10, 2, 2 > [month_p_t](#)
- typedef boost::spirit::qi::uint_parser< [day_t](#), 10, 2, 2 > [day_p_t](#)
- typedef unsigned short [DictionaryKey_T](#)
- typedef std::list< [AirlineClassList](#) * > [AirlineClassListList_T](#)
- typedef std::map< const [MapKey_T](#), [AirlineClassList](#) * > [AirlineClassListMap_T](#)
- typedef std::pair< [MapKey_T](#), [AirlineClassList](#) * > [AirlineClassListWithKey_T](#)
- typedef std::list< [AirlineClassListWithKey_T](#) > [AirlineClassListDetailedList_T](#)
- typedef std::list< [AirlineFeature](#) * > [AirlineFeatureList_T](#)
- typedef std::map< const [MapKey_T](#), [AirlineFeature](#) * > [AirlineFeatureMap_T](#)
- typedef std::list< [AirportPair](#) * > [AirportPairList_T](#)
- typedef std::map< const [MapKey_T](#), [AirportPair](#) * > [AirportPairMap_T](#)
- typedef std::pair< [MapKey_T](#), [AirportPair](#) * > [AirportPairWithKey_T](#)
- typedef std::list< [AirportPairWithKey_T](#) > [AirportPairDetailedList_T](#)
- typedef std::map< const std::type_info *, [BomAbstract](#) * > [HolderMap_T](#)
- typedef boost::tokenizer< boost::char_separator< char > > [Tokeniser_T](#)
- typedef std::list< [BookingClass](#) * > [BookingClassList_T](#)
- typedef std::map< const [MapKey_T](#), [BookingClass](#) * > [BookingClassMap_T](#)
- typedef boost::shared_ptr< [BookingRequestStruct](#) > [BookingRequestPtr_T](#)
- typedef std::string [DemandGeneratorKey_T](#)
- typedef std::list< [Bucket](#) * > [BucketList_T](#)
- typedef std::map< const [MapKey_T](#), [Bucket](#) * > [BucketMap_T](#)
- typedef boost::shared_ptr< [CancellationStruct](#) > [CancellationPtr_T](#)
- typedef std::list< [DatePeriod](#) * > [DatePeriodList_T](#)

- typedef std::map< const [MapKey_T](#), [DatePeriod](#) * > [DatePeriodMap_T](#)
- typedef std::pair< [MapKey_T](#), [DatePeriod](#) * > [DatePeriodWithKey_T](#)
- typedef std::list< [DatePeriodWithKey_T](#) > [DatePeriodDetailedList_T](#)
- typedef std::list< [EventQueue](#) * > [EventQueueList_T](#)
- typedef std::map< const [MapKey_T](#), [EventQueue](#) * > [EventQueueMap_T](#)
- typedef std::pair< const [LongDuration_T](#), [EventStruct](#) > [EventListElement_T](#)
- typedef std::map< const [LongDuration_T](#), [EventStruct](#) > [EventList_T](#)
- typedef std::list< [FareFamily](#) * > [FareFamilyList_T](#)
- typedef std::map< const [MapKey_T](#), [FareFamily](#) * > [FareFamilyMap_T](#)
- typedef std::list< [FareFeatures](#) * > [FareFeaturesList_T](#)
- typedef std::map< const [MapKey_T](#), [FareFeatures](#) * > [FareFeaturesMap_T](#)
- typedef std::pair< [MapKey_T](#), [FareFeatures](#) * > [FareFeaturesWithKey_T](#)
- typedef std::list< [FareFeaturesWithKey_T](#) > [FareFeaturesDetailedList_T](#)
- typedef std::list< [FareOptionStruct](#) > [FareOptionList_T](#)
- typedef std::list< [FlightDate](#) * > [FlightDateList_T](#)
- typedef std::map< const [MapKey_T](#), [FlightDate](#) * > [FlightDateMap_T](#)
- typedef std::list< [FlightPeriod](#) * > [FlightPeriodList_T](#)
- typedef std::map< const [MapKey_T](#), [FlightPeriod](#) * > [FlightPeriodMap_T](#)
- typedef std::list< [GuillotineBlock](#) * > [GuillotineBlockList_T](#)
- typedef std::map< const [MapKey_T](#), [GuillotineBlock](#) * > [GuillotineBlockMap_T](#)
- typedef std::map< const [SegmentCabin](#) *, [BlockNumber_T](#) > [SegmentCabinIndexMap_T](#)
- typedef std::map< const [MapKey_T](#), [BlockIndex_T](#) > [ValueTypeIndexMap_T](#)
- typedef std::list< [Inventory](#) * > [InventoryList_T](#)
- typedef std::map< const [MapKey_T](#), [Inventory](#) * > [InventoryMap_T](#)
- typedef std::string [MapKey_T](#)
- typedef std::list< std::string > [KeyList_T](#)
- typedef std::list< [LegCabin](#) * > [LegCabinList_T](#)
- typedef std::map< const [MapKey_T](#), [LegCabin](#) * > [LegCabinMap_T](#)
- typedef std::list< [LegDate](#) * > [LegDateList_T](#)
- typedef std::map< const [MapKey_T](#), [LegDate](#) * > [LegDateMap_T](#)
- typedef std::list< [OnDDate](#) * > [OnDDateList_T](#)
- typedef std::map< const [MapKey_T](#), [OnDDate](#) * > [OnDDateMap_T](#)
- typedef std::pair< std::string, [YieldDemandPair_T](#) > [StringDemandStructPair_T](#)
- typedef std::map< std::string, [YieldDemandPair_T](#) > [StringDemandStructMap_T](#)
- typedef std::map< std::string, [CabinClassPairList_T](#) > [StringCabinClassPairListMap_T](#)
- typedef std::pair< std::string, [CabinClassPairList_T](#) > [StringCabinClassPair_T](#)
- typedef std::map< [CabinCode_T](#), [WTPDemandPair_T](#) > [CabinForecastMap_T](#)
- typedef std::pair< [CabinCode_T](#), [WTPDemandPair_T](#) > [CabinForecastPair_T](#)
- typedef boost::shared_ptr< [OptimisationNotificationStruct](#) > [OptimisationNotificationPtr_T](#)
- typedef std::list< [PosChannel](#) * > [PosChannelList_T](#)
- typedef std::map< const [MapKey_T](#), [PosChannel](#) * > [PosChannelMap_T](#)
- typedef std::pair< [MapKey_T](#), [PosChannel](#) * > [PosChannelWithKey_T](#)
- typedef std::list< [PosChannelWithKey_T](#) > [PosChannelDetailedList_T](#)

- typedef boost::shared_ptr< RMEventStruct > RMEventPtr_T
- typedef std::list< RMEventStruct > RMEventList_T
- typedef std::list< SegmentCabin * > SegmentCabinList_T
- typedef std::map< const MapKey_T, SegmentCabin * > SegmentCabinMap_T
- typedef std::list< SegmentDate * > SegmentDateList_T
- typedef std::map< const MapKey_T, SegmentDate * > SegmentDateMap_T
- typedef std::list< SegmentPeriod * > SegmentPeriodList_T
- typedef std::map< const MapKey_T, SegmentPeriod * > SegmentPeriodMap_T
- typedef std::pair< MapKey_T, SegmentPeriod * > SegmentPeriodWithKey_T
- typedef std::list< SegmentPeriodWithKey_T > SegmentPeriodDetailedList_T
- typedef boost::shared_ptr< SnapshotStruct > SnapshotPtr_T
- typedef std::list< TimePeriod * > TimePeriodList_T
- typedef std::map< const MapKey_T, TimePeriod * > TimePeriodMap_T
- typedef std::pair< MapKey_T, TimePeriod * > TimePeriodWithKey_T
- typedef std::list< TimePeriodWithKey_T > TimePeriodDetailedList_T
- typedef std::list< TravelSolutionStruct > TravelSolutionList_T
- typedef KeyList_T SegmentPath_T
- typedef std::list< SegmentPath_T > SegmentPathList_T
- typedef std::map< const ClassCode_T, Availability_T > ClassAvailabilityMap_T
- typedef std::list< ClassAvailabilityMap_T > ClassAvailabilityMapHolder_T
- typedef std::map< const ClassCode_T, YieldValue_T > ClassYieldMap_T
- typedef std::list< ClassYieldMap_T > ClassYieldMapHolder_T
- typedef std::list< BidPriceVector_T > BidPriceVectorHolder_T
- typedef std::map< const ClassCode_T, const BidPriceVector_T * > ClassBpvMap_T
- typedef std::list< ClassBpvMap_T > ClassBpvMapHolder_T
- typedef std::list< VirtualClassStruct > VirtualClassList_T
- typedef std::map< const Yield_T, VirtualClassStruct > VirtualClassMap_T
- typedef std::list< YieldFeatures * > YieldFeaturesList_T
- typedef std::map< const MapKey_T, YieldFeatures * > YieldFeaturesMap_T
- typedef std::pair< MapKey_T, YieldFeatures * > YieldFeaturesWithKey_T
- typedef std::list< YieldFeaturesWithKey_T > YieldFeaturesDetailedList_T
- typedef std::list< YieldStore * > YieldStoreList_T
- typedef std::map< const MapKey_T, YieldStore * > YieldStoreMap_T
- typedef std::string LocationCode_T
- typedef unsigned long int Distance_T
- typedef LocationCode_T AirportCode_T
- typedef LocationCode_T CityCode_T
- typedef std::string KeyDescription_T
- typedef std::string AirlineCode_T
- typedef unsigned short FlightNumber_T
- typedef unsigned short GuillotineNumber_T
- typedef std::string CabinCode_T
- typedef std::string FamilyCode_T
- typedef std::string ClassCode_T
- typedef unsigned long Identity_T

- typedef std::string [TripType_T](#)
- typedef double [MonetaryValue_T](#)
- typedef double [RealNumber_T](#)
- typedef double [Percentage_T](#)
- typedef double [PriceValue_T](#)
- typedef double [YieldValue_T](#)
- typedef std::string [PriceCurrency_T](#)
- typedef double [Revenue_T](#)
- typedef double [Multiplier_T](#)
- typedef double [NbOfSeats_T](#)
- typedef unsigned int [Count_T](#)
- typedef short [PartySize_T](#)
- typedef double [NbOfRequests_T](#)
- typedef [NbOfRequests_T](#) [NbOfBookings_T](#)
- typedef [NbOfRequests_T](#) [NbOfCancellations_T](#)
- typedef unsigned short [NbOfTravelSolutions_T](#)
- typedef std::string [ClassList_String_T](#)
- typedef unsigned short [NbOfSegments_T](#)
- typedef unsigned short [NbOfAirlines_T](#)
- typedef double [Availability_T](#)
- typedef double [Fare_T](#)
- typedef bool [Flag_T](#)
- typedef unsigned int [UnsignedIndex_T](#)
- typedef std::string [Filename_T](#)
- typedef std::string [FileAddress_T](#)
- typedef float [ProgressPercentage_T](#)
- typedef boost::posix_time::time_duration [Duration_T](#)
- typedef boost::gregorian::date [Date_T](#)
- typedef boost::posix_time::time_duration [Time_T](#)
- typedef boost::posix_time::ptime [DateTime_T](#)
- typedef boost::gregorian::date_period [DatePeriod_T](#)
- typedef std::string [DOW_String_T](#)
- typedef boost::gregorian::date_duration [DateOffset_T](#)
- typedef unsigned int [DayDuration_T](#)
- typedef bool [SaturdayStay_T](#)
- typedef long int [IntDuration_T](#)
- typedef unsigned long long int [LongDuration_T](#)
- typedef float [FloatDuration_T](#)
- typedef soci::session [DBSession_T](#)
- typedef soci::statement [DBRequestStatement_T](#)
- typedef std::string [DBConnectionName_T](#)
- typedef bool [ChangeFees_T](#)
- typedef bool [NonRefundable_T](#)
- typedef unsigned int [SaturdayStayRatio_T](#)
- typedef unsigned int [ChangeFeesRatio_T](#)
- typedef unsigned int [NonRefundableRatio_T](#)

- typedef std::string [PassengerType_T](#)
- typedef std::string [DistributionPatternId_T](#)
- typedef std::string [CancellationRateCurveId_T](#)
- typedef std::string [AirlinePreferenceId_T](#)
- typedef std::pair< [Percentage_T](#), [Percentage_T](#) > [CancellationNoShowRatePair_T](#)
- typedef std::string [CharacteristicsPatternId_T](#)
- typedef std::string [CharacteristicsIndex_T](#)
- typedef double [WTP_T](#)
- typedef boost::tuples::tuple< double, [WTP_T](#) > [CharacteristicsWTP_tuple_T](#)
- typedef std::pair< [WTP_T](#), [MeanStdDevPair_T](#) > [WTPDemandPair_T](#)
- typedef [NbOfRequests_T](#) [NbOfNoShows_T](#)
- typedef double [MatchingIndicator_T](#)
- typedef std::string [DemandStreamKeyStr_T](#)
- typedef std::string [ChannelLabel_T](#)
- typedef std::string [FrequentFlyer_T](#)
- typedef std::string [RequestStatus_T](#)
- typedef std::map< [Identity_T](#), [Identity_T](#) > [BookingTSIDMap_T](#)
- typedef std::pair< [CabinCode_T](#), [ClassCode_T](#) > [CabinClassPair_T](#)
- typedef std::list< [CabinClassPair_T](#) > [CabinClassPairList_T](#)
- typedef double [ProportionFactor_T](#)
- typedef std::list< [ProportionFactor_T](#) > [ProportionFactorList_T](#)
- typedef std::string [OnDString_T](#)
- typedef std::list< [OnDString_T](#) > [OnDStringList_T](#)
- typedef std::string [EventName_T](#)
- typedef double [NbOfEvents_T](#)
- typedef std::string [EventQueueId_T](#)
- typedef std::string [EventGeneratorKey_T](#)
- typedef double [NbOfFareRules_T](#)
- typedef std::string [NetworkId_T](#)
- typedef std::vector< [AirlineCode_T](#) > [AirlineCodeList_T](#)
- typedef std::vector< [ClassList_String_T](#) > [ClassList_StringList_T](#)
- typedef std::vector< [ClassCode_T](#) > [ClassCodeList_T](#)
- typedef unsigned short [SubclassCode_T](#)
- typedef std::string [FlightPathCode_T](#)
- typedef std::map< [CabinCode_T](#), [ClassList_String_T](#) > [CabinBookingClassMap_T](#)
- typedef double [CabinCapacity_T](#)
- typedef double [NbOfFlightDates_T](#)
- typedef double [CommittedSpace_T](#)
- typedef double [UPR_T](#)
- typedef double [BookingLimit_T](#)
- typedef double [AuthorizationLevel_T](#)
- typedef double [CapacityAdjustment_T](#)
- typedef double [BlockSpace_T](#)
- typedef bool [AvailabilityStatus_T](#)

- typedef std::vector< [Availability_T](#) > [BucketAvailabilities_T](#)
- typedef double [NbOfYields_T](#)
- typedef double [NbOfInventoryControlRules_T](#)
- typedef bool [CensorshipFlag_T](#)
- typedef short [DTD_T](#)
- typedef short [DCP_T](#)
- typedef std::list< [DCP_T](#) > [DCPList_T](#)
- typedef std::map< [DTD_T](#), [RealNumber_T](#) > [DTDFratMap_T](#)
- typedef std::map< [FloatDuration_T](#), float > [DTDProbMap_T](#)
- typedef std::vector< [CensorshipFlag_T](#) > [CensorshipFlagList_T](#)
- typedef double [BookingRatio_T](#)
- typedef double [Yield_T](#)
- typedef unsigned int [YieldLevel_T](#)
- typedef std::map< [YieldLevel_T](#), [MeanStdDevPair_T](#) > [YieldLevelDemandMap_T](#)
- typedef std::pair< [Yield_T](#), [MeanStdDevPair_T](#) > [YieldDemandPair_T](#)
- typedef double [BidPrice_T](#)
- typedef std::vector< [BidPrice_T](#) > [BidPriceVector_T](#)
- typedef unsigned int [SeatIndex_T](#)
- typedef std::string [ControlMode_T](#)
- typedef double [OverbookingRate_T](#)
- typedef double [ProtectionLevel_T](#)
- typedef std::vector< double > [EmsrValueList_T](#)
- typedef std::vector< double > [BookingLimitVector_T](#)
- typedef std::vector< double > [ProtectionLevelVector_T](#)
- typedef boost::multi_array< double, 2 > [SnapshotBlock_T](#)
- typedef [SnapshotBlock_T](#)::index_range [SnapshotBlockRange_T](#)
- typedef [SnapshotBlock_T](#)::array_view< 1 >::type [SegmentCabinDTDSnapshotView_T](#)
- typedef [SnapshotBlock_T](#)::array_view< 2 >::type [SegmentCabinDTDRangeSnapshotView_T](#)
- typedef [SnapshotBlock_T](#)::const_array_view< 1 >::type [ConstSegmentCabinDTDSnapshotView_T](#)
- typedef [SnapshotBlock_T](#)::const_array_view< 2 >::type [ConstSegmentCabinDTDRangeSnapshotView_T](#)
- typedef unsigned short [BlockNumber_T](#)
- typedef unsigned short [BlockIndex_T](#)
- typedef unsigned int [ReplicationNumber_T](#)
- typedef unsigned long int [ExponentialSeed_T](#)
- typedef unsigned long int [UniformSeed_T](#)
- typedef unsigned long int [RandomSeed_T](#)
- typedef boost::minstd_rand [BaseGenerator_T](#)
- typedef boost::uniform_real [UniformDistribution_T](#)
- typedef boost::variate_generator< [BaseGenerator_T](#) &, [UniformDistribution_T](#) > [UniformGenerator_T](#)
- typedef boost::normal_distribution [NormalDistribution_T](#)

- typedef boost::variate_generator< BaseGenerator_T &, NormalDistribution_T > NormalGenerator_T
- typedef boost::exponential_distribution ExponentialDistribution_T
- typedef boost::variate_generator< BaseGenerator_T &, ExponentialDistribution_T > ExponentialGenerator_T
- typedef double MeanValue_T
- typedef double StdDevValue_T
- typedef std::pair< MeanValue_T, StdDevValue_T > MeanStdDevPair_T
- typedef float Probability_T
- typedef std::string ForecasterMode_T
- typedef short HistoricalDataLimit_T
- typedef std::string OptimizerMode_T
- typedef NbOfBookings_T PolicyDemand_T
- typedef std::vector< double > GeneratedDemandVector_T
- typedef std::vector< GeneratedDemandVector_T > GeneratedDemandVectorHolder_T
- typedef double SellupProbability_T
- typedef std::vector< SellupProbability_T > SellupProbabilityVector_T
- typedef std::vector< double > SellupFactorHolder_T
- typedef boost::shared_ptr< STDAIR_Service > STDAIR_ServicePtr_T

Functions

- const std::string DEFAULT_BOM_ROOT_KEY (" -- ROOT -- ")
- const double DEFAULT_EPSILON_VALUE (0.0001)
- const unsigned int DEFAULT_FLIGHT_SPEED (900)
- const NbOfFlightDates_T DEFAULT_NB_OF_FLIGHTDATES (0.0)
- const Duration_T NULL_BOOST_TIME_DURATION (-1,-1,-1)
- const unsigned int DEFAULT_NB_OF_DAYS_IN_A_YEAR (365)
- const unsigned int DEFAULT_NUMBER_OF_SUBDIVISIONS (1000)
- const DayDuration_T DEFAULT_DAY_DURATION (0)
- const DatePeriod_T BOOST_DEFAULT_DATE_PERIOD (Date_T(2007, 1, 1), Date_T(2007, 1, 1))
- const DOW_String_T DEFAULT_DOW_STRING ("0000000")
- const DateOffset_T DEFAULT_DATE_OFFSET (0)
- const Date_T DEFAULT_DATE (2010, boost::gregorian::Jan, 1)
- const DateTime_T DEFAULT_DATETIME (DEFAULT_DATE, NULL_BOOST_TIME_DURATION)
- const Duration_T DEFAULT_EPSILON_DURATION (0, 0, 0, 1)
- const Count_T SECONDS_IN_ONE_DAY (86400)
- const Count_T MILLISECONDS_IN_ONE_SECOND (1000)
- const RandomSeed_T DEFAULT_RANDOM_SEED (120765987)
- const AirportCode_T AIRPORT_LHR ("LHR")
- const AirportCode_T AIRPORT_SYD ("SYD")
- const CityCode_T POS_LHR ("LHR")
- const Date_T DATE_20110115 (2011, boost::gregorian::Jan, 15)

- const [Date_T DATE_20111231](#) (2011, boost::gregorian::Dec, 31)
- const [DayDuration_T NO_ADVANCE_PURCHASE](#) (0)
- const [SaturdayStay_T SATURDAY_STAY](#) (true)
- const [SaturdayStay_T NO_SATURDAY_STAY](#) (false)
- const [ChangeFees_T CHANGE_FEES](#) (true)
- const [ChangeFees_T NO_CHANGE_FEES](#) (false)
- const [NonRefundable_T NON_REFUNDABLE](#) (true)
- const [NonRefundable_T No_NON_REFUNDABLE](#) (false)
- const [SaturdayStay_T DEFAULT_BOM_TREE_SATURDAY_STAY](#) (true)
- const [ChangeFees_T DEFAULT_BOM_TREE_CHANGE_FEES](#) (true)
- const [NonRefundable_T DEFAULT_BOM_TREE_NON_REFUNDABLE](#) (true)
- const [DayDuration_T NO_STAY_DURATION](#) (0)
- const [AirlineCode_T AIRLINE_CODE_BA](#) ("BA")
- const [CabinCode_T CABIN_Y](#) ("Y")
- const [ClassCode_T CLASS_CODE_Y](#) ("Y")
- const [ClassCode_T CLASS_CODE_Q](#) ("Q")
- const [AirportCode_T AIRPORT_SIN](#) ("SIN")
- const [AirportCode_T AIRPORT_BKK](#) ("BKK")
- const [CityCode_T POS_SIN](#) ("SIN")
- const [CabinCode_T CABIN_ECO](#) ("Eco")
- const [FrequentFlyer_T FREQUENT_FLYER_MEMBER](#) ("M")
- const [ClassCode_T DEFAULT_FAMILY_CODE](#) ("0")
- const [NbOfAirlines_T DEFAULT_NBOFAIRLINES](#) (0)
- const [FlightPathCode_T DEFAULT_FLIGHTPATH_CODE](#) ("")
- const [Distance_T DEFAULT_DISTANCE_VALUE](#) (0)
- const [ClassCode_T DEFAULT_CLOSED_CLASS_CODE](#) ("CC")
- const [NbOfBookings_T DEFAULT_CLASS_NB_OF_BOOKINGS](#) (0)
- const [NbOfBookings_T DEFAULT_CLASS_TOTAL_NB_OF_BOOKINGS](#) (0)
- const [NbOfBookings_T DEFAULT_CLASS_UNCONSTRAINED_DEMAND](#) (0)
- const [NbOfBookings_T DEFAULT_CLASS_REMAINING_DEMAND_MEAN](#) (0)
- const [NbOfBookings_T DEFAULT_CLASS_REMAINING_DEMAND_STANDARD_DEVIATION](#) (0)
- const [NbOfCancellations_T DEFAULT_CLASS_NB_OF_CANCELLATIONS](#) (0)
- const [NbOfNoShows_T DEFAULT_CLASS_NB_OF_NOSHOWS](#) (0)
- const [CabinCapacity_T DEFAULT_CABIN_CAPACITY](#) (100.0)
- const [CommittedSpace_T DEFAULT_COMMITTED_SPACE](#) (0.0)
- const [BlockSpace_T DEFAULT_BLOCK_SPACE](#) (0.0)
- const [Availability_T DEFAULT_NULL_AVAILABILITY](#) (0.0)
- const [Availability_T DEFAULT_AVAILABILITY](#) (9.0)
- const [Availability_T MAXIMAL_AVAILABILITY](#) (9999.0)
- const [CensorshipFlag_T DEFAULT_CLASS_CENSORSHIPFLAG](#) (false)
- const [BookingLimit_T DEFAULT_CLASS_BOOKING_LIMIT](#) (9999.0)
- const [AuthorizationLevel_T DEFAULT_CLASS_AUTHORIZATION_LEVEL](#) (9999.0)
- const [AuthorizationLevel_T DEFAULT_CLASS_MAX_AUTHORIZATION_LEVEL](#) (9999.0)
- const [AuthorizationLevel_T DEFAULT_CLASS_MIN_AUTHORIZATION_LEVEL](#) (0.0)

- const [OverbookingRate_T](#) DEFAULT_CLASS_OVERBOOKING_RATE (0.0)
- const [BookingRatio_T](#) DEFAULT_OND_BOOKING_RATE (0.0)
- const [Fare_T](#) DEFAULT_FARE_VALUE (0.0)
- const [Yield_T](#) DEFAULT_CLASS_YIELD_VALUE (0.0)
- const [Revenue_T](#) DEFAULT_REVENUE_VALUE (0.0)
- const [Percentage_T](#) DEFAULT_LOAD_FACTOR_VALUE (100.0)
- const [Yield_T](#) DEFAULT_YIELD_VALUE (0.0)
- const [Yield_T](#) DEFAULT_YIELD_MAX_VALUE (std::numeric_limits< double >::max())
- const [NbOfBookings_T](#) DEFAULT_YIELD_NB_OF_BOOKINGS (0.0)
- const [Identity_T](#) DEFAULT_BOOKING_NUMBER (0)
- const [NbOfCancellations_T](#) DEFAULT_YIELD_NB_OF_CANCELLATIONS (0.0)
- const [NbOfNoShows_T](#) DEFAULT_YIELD_NB_OF_NOSHOWS (0.0)
- const [Availability_T](#) DEFAULT_YIELD_AVAILABILITY (0.0)
- const [CensorshipFlag_T](#) DEFAULT_YIELD_CENSORSHIPFLAG (false)
- const [BookingLimit_T](#) DEFAULT_YIELD_BOOKING_LIMIT (0.0)
- const [OverbookingRate_T](#) DEFAULT_YIELD_OVERBOOKING_RATE (0.0)
- const [Fare_T](#) DEFAULT_OND_FARE_VALUE (0.0)
- const [EventQueueID_T](#) DEFAULT_EVENT_QUEUE_ID ("EQ01")
- const [Count_T](#) DEFAULT_PROGRESS_STATUS (0)
- const [Date_T](#) DEFAULT_EVENT_OLDEST_DATE (2008, boost::gregorian::Jan, 1)
- const [DateTime_T](#) DEFAULT_EVENT_OLDEST_DATETIME (DEFAULT_EVENT_OLDEST_DATE, NULL_BOOST_TIME_DURATION)
- const [PartySize_T](#) DEFAULT_PARTY_SIZE (1)
- const [DayDuration_T](#) DEFAULT_STAY_DURATION (7)
- const [WTP_T](#) DEFAULT_WTP (1000.0)
- const [Date_T](#) DEFAULT_PREFERRED_DEPARTURE_DATE (DEFAULT_DEPARTURE_DATE)
- const [Duration_T](#) DEFAULT_PREFERRED_DEPARTURE_TIME (8, 0, 0)
- const [DateOffset_T](#) DEFAULT_ADVANCE_PURCHASE (22)
- const [Date_T](#) DEFAULT_REQUEST_DATE (DEFAULT_PREFERRED_DEPARTURE_DATE-DEFAULT_ADVANCE_PURCHASE)
- const [Duration_T](#) DEFAULT_REQUEST_TIME (8, 0, 0)
- const [DateTime_T](#) DEFAULT_REQUEST_DATE_TIME (DEFAULT_REQUEST_DATE, DEFAULT_REQUEST_TIME)
- const [CabinCode_T](#) DEFAULT_PREFERRED_CABIN ("M")
- const [CityCode_T](#) DEFAULT_POS ("ALL")
- const [ChannelLabel_T](#) DEFAULT_CHANNEL ("DC")
- const [ChannelLabel_T](#) CHANNEL_DN ("DN")
- const [ChannelLabel_T](#) CHANNEL_IN ("IN")
- const [TripType_T](#) TRIP_TYPE_ONE_WAY ("OW")
- const [TripType_T](#) TRIP_TYPE_ROUND_TRIP ("RT")
- const [TripType_T](#) TRIP_TYPE_INBOUND ("RI")
- const [TripType_T](#) TRIP_TYPE_OUTBOUND ("RO")
- const [FrequentFlyer_T](#) DEFAULT_FF_TIER ("N")
- const [PriceValue_T](#) DEFAULT_VALUE_OF_TIME (100.0)
- const [Duration_T](#) DEFAULT_MINIMAL_CONNECTION_TIME (0, 30, 0)

- const [Duration_T](#) [DEFAULT_MAXIMAL_CONNECTION_TIME](#) (24, 0, 0)
- const [MatchingIndicator_T](#) [DEFAULT_MATCHING_INDICATOR](#) (0.0)
- const [PriceCurrency_T](#) [DEFAULT_CURRENCY](#) ("EUR")
- const [AvailabilityStatus_T](#) [DEFAULT_AVAILABILITY_STATUS](#) (false)
- const [AirlineCode_T](#) [DEFAULT_AIRLINE_CODE](#) ("XX")
- const [AirlineCode_T](#) [DEFAULT_NULL_AIRLINE_CODE](#) ("")
- const [FlightNumber_T](#) [DEFAULT_FLIGHT_NUMBER](#) (9999)
- const [GuillotineNumber_T](#) [DEFAULT_GUILLOTINE_NUMBER](#) (9999)
- const [Date_T](#) [DEFAULT_DEPARTURE_DATE](#) (1900, boost::gregorian::Jan, 1)
- const [AirportCode_T](#) [DEFAULT_AIRPORT_CODE](#) ("XXX")
- const [AirportCode_T](#) [DEFAULT_NULL_AIRPORT_CODE](#) ("")
- const [AirportCode_T](#) [DEFAULT_ORIGIN](#) ("XXX")
- const [AirportCode_T](#) [DEFAULT_DESTINATION](#) ("XXX")
- const [CabinCode_T](#) [DEFAULT_CABIN_CODE](#) ("X")
- const [FamilyCode_T](#) [DEFAULT_FARE_FAMILY_CODE](#) ("EcoSaver")
- const [FamilyCode_T](#) [DEFAULT_NULL_FARE_FAMILY_CODE](#) ("NoFF")
- const [ClassCode_T](#) [DEFAULT_CLASS_CODE](#) ("X")
- const [ClassCode_T](#) [DEFAULT_NULL_CLASS_CODE](#) ("")
- const [BidPrice_T](#) [DEFAULT_BID_PRICE](#) (0.0)
- const unsigned short [MAXIMAL_NUMBER_OF_LEGS_IN_FLIGHT](#) (7)
- const unsigned short [MAXIMAL_NUMBER_OF_SEGMENTS_IN_OND](#) (3)
- const [SeatIndex_T](#) [DEFAULT_SEAT_INDEX](#) (1)
- const std::string [DEFAULT_FARE_FAMILY_VALUE_TYPE](#) ("FF")
- const std::string [DEFAULT_SEGMENT_CABIN_VALUE_TYPE](#) ("SC")
- const std::string [DEFAULT_KEY_FLD_DELIMITER](#) (";")
- const std::string [DEFAULT_KEY_SUB_FLD_DELIMITER](#) (",")
- const boost::char_separator< char > [DEFAULT_KEY_TOKEN_DELIMITER](#) (";,")
- template<int MIN, int MAX>
[date_time_element](#)< MIN, MAX > [operator*](#) (const [date_time_element](#)< MIN, MAX > &o1, const [date_time_element](#)< MIN, MAX > &o2)
- template<int MIN, int MAX>
[date_time_element](#)< MIN, MAX > [operator+](#) (const [date_time_element](#)< MIN, MAX > &o1, const [date_time_element](#)< MIN, MAX > &o2)
- template void [AirlineClassListKey::serialize](#)< [ba::text_oarchive](#) > ([ba::text_oarchive](#) &, unsigned int)
- template void [AirlineClassListKey::serialize](#)< [ba::text_iarchive](#) > ([ba::text_iarchive](#) &, unsigned int)
- void [jsonLegDateExport](#) ([bpt::ptree](#) &ioPropertyTree, const [FlightDate](#) &iFlightDate)
- void [jsonSegmentDateExport](#) ([bpt::ptree](#) &ioPropertyTree, const [FlightDate](#) &iFlightDate)
- void [jsonLegCabinExport](#) ([bpt::ptree](#) &ioPropertyTree, const [FlightDate](#) &iFlightDate)
- void [jsonSegmentCabinExport](#) ([bpt::ptree](#) &ioPropertyTree, const [FlightDate](#) &iFlightDate)

- void [jsonFareFamilyExport](#) (bpt::ptree &ioPropertyTree, const [FlightDate](#) &iFlightDate)
- void [jsonBucketExport](#) (bpt::ptree &ioPropertyTree, const [FlightDate](#) &iFlightDate)
- void [jsonBookingClassExport](#) (bpt::ptree &ioPropertyTree, const [BookingClass](#) &iBookingClass, const std::string &iLeadingString)
- void [jsonBookingClassExport](#) (bpt::ptree &ioPropertyTree, const [FlightDate](#) &iFlightDate)
- template void [BomRootKey::serialize](#)< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void [BomRootKey::serialize](#)< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- void [intDisplay](#) (std::ostream &oStream, const int &iInt)
- template void [BucketKey::serialize](#)< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void [BucketKey::serialize](#)< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void [FareFamilyKey::serialize](#)< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void [FareFamilyKey::serialize](#)< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void [FlightDateKey::serialize](#)< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void [FlightDateKey::serialize](#)< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void [GuillotineBlockKey::serialize](#)< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void [GuillotineBlockKey::serialize](#)< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void [InventoryKey::serialize](#)< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void [InventoryKey::serialize](#)< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void [OnDDateKey::serialize](#)< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void [OnDDateKey::serialize](#)< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- const boost::char_separator< char > [TokeniserDashSeparator](#) ("-")
- const boost::char_separator< char > [TokeniserTimeSeparator](#) (":")
- template void [SegmentCabinKey::serialize](#)< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void [SegmentCabinKey::serialize](#)< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void [SegmentDateKey::serialize](#)< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void [SegmentDateKey::serialize](#)< ba::text_iarchive > (ba::text_iarchive &, unsigned int)

- template<class Archive , class BOM_OBJECT1 , class BOM_OBJECT2 >
void [serialiseHelper](#) (BOM_OBJECT1 &ioObject1, Archive &ioArchive, const unsigned int iFileVersion)
- template void [BomRoot::serialize](#)< [ba::text_oarchive](#) > ([ba::text_oarchive](#) &, unsigned int)
- template void [BomRoot::serialize](#)< [ba::text_iarchive](#) > ([ba::text_iarchive](#) &, unsigned int)
- template void [Inventory::serialize](#)< [ba::text_oarchive](#) > ([ba::text_oarchive](#) &, unsigned int)
- template void [Inventory::serialize](#)< [ba::text_iarchive](#) > ([ba::text_iarchive](#) &, unsigned int)
- template void [FlightDate::serialize](#)< [ba::text_oarchive](#) > ([ba::text_oarchive](#) &, unsigned int)
- template void [FlightDate::serialize](#)< [ba::text_iarchive](#) > ([ba::text_iarchive](#) &, unsigned int)
- template void [SegmentDate::serialize](#)< [ba::text_oarchive](#) > ([ba::text_oarchive](#) &, unsigned int)
- template void [SegmentDate::serialize](#)< [ba::text_iarchive](#) > ([ba::text_iarchive](#) &, unsigned int)
- template void [SegmentCabin::serialize](#)< [ba::text_oarchive](#) > ([ba::text_oarchive](#) &, unsigned int)
- template void [SegmentCabin::serialize](#)< [ba::text_iarchive](#) > ([ba::text_iarchive](#) &, unsigned int)

Variables

- const std::string [DOW_STR](#) []
- const [CensorshipFlagList_T](#) [DEFAULT_CLASS_CENSORSHIPFLAG_LIST](#)
- const [Date_T](#) [DEFAULT_DICO_STUDIED_DATE](#)
- const [AirlineCodeList_T](#) [DEFAULT_AIRLINE_CODE_LIST](#)
- const [ClassList_StringList_T](#) [DEFAULT_CLASS_CODE_LIST](#)
- const [BidPriceVector_T](#) [DEFAULT_BID_PRICE_VECTOR](#) = std::vector<[BidPrice_T](#)>()
- const int [DEFAULT_MAX_DTD](#) = 365
- const [DCPList_T](#) [DEFAULT_DCP_LIST](#) = [DefaultDCPList::init](#)()
- const [DTDFratMap_T](#) [DEFAULT_DTD_FRAT5COEF_MAP](#)
- const [DTDProbMap_T](#) [DEFAULT_DTD_PROB_MAP](#)
- const [OnDStringList_T](#) [DEFAULT_OND_STRING_LIST](#)
- const std::string [DISPLAY_LEVEL_STRING_ARRAY](#) [51]
- const std::string [DEFAULT_KEY_FLD_DELIMITER](#)
- const std::string [DEFAULT_KEY_SUB_FLD_DELIMITER](#)
- const boost::char_separator< char > [DEFAULT_KEY_TOKEN_DELIMITER](#)
- const [Distance_T](#) [DEFAULT_DISTANCE_VALUE](#)
- const [ClassCode_T](#) [DEFAULT_CLOSED_CLASS_CODE](#)
- const [NbOfBookings_T](#) [DEFAULT_CLASS_NB_OF_BOOKINGS](#)
- const [NbOfBookings_T](#) [DEFAULT_CLASS_TOTAL_NB_OF_BOOKINGS](#)
- const [NbOfBookings_T](#) [DEFAULT_CLASS_UNCONSTRAINED_DEMAND](#)

- const [NbOfBookings_T](#) DEFAULT_CLASS_REMAINING_DEMAND_MEAN
- const [NbOfBookings_T](#) DEFAULT_CLASS_REMAINING_DEMAND_STANDARD_DEVIATION
- const [NbOfCancellations_T](#) DEFAULT_CLASS_NB_OF_CANCELLATIONS
- const [NbOfNoShows_T](#) DEFAULT_CLASS_NB_OF_NOSHOWS
- const [CabinCapacity_T](#) DEFAULT_CABIN_CAPACITY
- const [CommittedSpace_T](#) DEFAULT_COMMITTED_SPACE
- const [BlockSpace_T](#) DEFAULT_BLOCK_SPACE
- const [Availability_T](#) DEFAULT_NULL_AVAILABILITY
- const [Availability_T](#) DEFAULT_AVAILABILITY
- const [CensorshipFlag_T](#) DEFAULT_CLASS_CENSORSHIPFLAG
- const [BookingLimit_T](#) DEFAULT_CLASS_BOOKING_LIMIT
- const [AuthorizationLevel_T](#) DEFAULT_CLASS_AUTHORIZATION_LEVEL
- const [AuthorizationLevel_T](#) DEFAULT_CLASS_MAX_AUTHORIZATION_LEVEL
- const [AuthorizationLevel_T](#) DEFAULT_CLASS_MIN_AUTHORIZATION_LEVEL
- const [OverbookingRate_T](#) DEFAULT_CLASS_OVERBOOKING_RATE
- const [Fare_T](#) DEFAULT_FARE_VALUE
- const [Revenue_T](#) DEFAULT_REVENUE_VALUE
- const [PriceCurrency_T](#) DEFAULT_CURRENCY
- const [Percentage_T](#) DEFAULT_LOAD_FACTOR_VALUE
- const [DayDuration_T](#) DEFAULT_DAY_DURATION
- const double DEFAULT_EPSILON_VALUE
- const [AirportCode_T](#) AIRPORT_LHR
- const [AirportCode_T](#) AIRPORT_SYD
- const [CityCode_T](#) POS_LHR
- const [DayDuration_T](#) NO_ADVANCE_PURCHASE
- const [SaturdayStay_T](#) SATURDAY_STAY
- const [SaturdayStay_T](#) NO_SATURDAY_STAY
- const [ChangeFees_T](#) CHANGE_FEES
- const [ChangeFees_T](#) NO_CHANGE_FEES
- const [NonRefundable_T](#) NON_REFUNDABLE
- const [NonRefundable_T](#) NO_NON_REFUNDABLE
- const [DayDuration_T](#) NO_STAY_DURATION
- const [CabinCode_T](#) CABIN_Y
- const [AirlineCode_T](#) AIRLINE_CODE_BA
- const [ClassCode_T](#) CLASS_CODE_Y
- const [ClassCode_T](#) CLASS_CODE_Q
- const [AirportCode_T](#) AIRPORT_SIN
- const [AirportCode_T](#) AIRPORT_BKK
- const [CityCode_T](#) POS_SIN
- const [CabinCode_T](#) CABIN_ECO
- const [FrequentFlyer_T](#) FREQUENT_FLYER_MEMBER
- const [EventQueueID_T](#) DEFAULT_EVENT_QUEUE_ID
- const [Count_T](#) DEFAULT_PROGRESS_STATUS
- const [Date_T](#) DEFAULT_EVENT_OLDEST_DATE
- const [DateTime_T](#) DEFAULT_EVENT_OLDEST_DATETIME

- const std::string [DEFAULT_BOM_ROOT_KEY](#)
- const [NbOfFlightDates_T](#) [DEFAULT_NB_OF_FLIGHTDATES](#)
- const unsigned int [DEFAULT_FLIGHT_SPEED](#)
- const [BookingRatio_T](#) [DEFAULT_OND_BOOKING_RATE](#)
- const [Count_T](#) [SECONDS_IN_ONE_DAY](#)
- const [Count_T](#) [MILLISECONDS_IN_ONE_SECOND](#)
- const [Date_T](#) [DEFAULT_DATE](#)
- const [DateTime_T](#) [DEFAULT_DATETIME](#)
- const [Duration_T](#) [DEFAULT_EPSILON_DURATION](#)
- const [RandomSeed_T](#) [DEFAULT_RANDOM_SEED](#)
- const [Duration_T](#) [NULL_BOOST_TIME_DURATION](#)
- const [Fare_T](#) [DEFAULT_CLASS_FARE_VALUE](#)
- const [NbOfAirlines_T](#) [DEFAULT_NB_OF_AIRLINES](#)
- const unsigned int [DEFAULT_NB_OF_DAYS_IN_A_YEAR](#)
- const [Channellabel_T](#) [DEFAULT_CHANNEL](#)
- const unsigned int [DEFAULT_NUMBER_OF_SUBDIVISIONS](#)
- const [AirlineCode_T](#) [DEFAULT_AIRLINE_CODE](#)
- const [AirlineCode_T](#) [DEFAULT_NULL_AIRLINE_CODE](#)
- const [FlightNumber_T](#) [DEFAULT_FLIGHT_NUMBER](#)
- const [GuillotineNumber_T](#) [DEFAULT_GUILLOTINE_NUMBER](#)
- const [Date_T](#) [DEFAULT_DEPARTURE_DATE](#)
- const [AirportCode_T](#) [DEFAULT_AIRPORT_CODE](#)
- const [AirportCode_T](#) [DEFAULT_NULL_AIRPORT_CODE](#)
- const [AirportCode_T](#) [DEFAULT_ORIGIN](#)
- const [AirportCode_T](#) [DEFAULT_DESTINATION](#)
- const [CabinCode_T](#) [DEFAULT_CABIN_CODE](#)
- const [FamilyCode_T](#) [DEFAULT_FARE_FAMILY_CODE](#)
- const [FamilyCode_T](#) [DEFAULT_NULL_FARE_FAMILY_CODE](#)
- const [ClassCode_T](#) [DEFAULT_CLASS_CODE](#)
- const [ClassCode_T](#) [DEFAULT_NULL_CLASS_CODE](#)
- const [BidPrice_T](#) [DEFAULT_BID_PRICE](#)
- const unsigned short [MAXIMAL_NUMBER_OF_LEGS_IN_FLIGHT](#)
- const unsigned short [MAXIMAL_NUMBER_OF_SEGMENTS_IN_OND](#)
- const [Availability_T](#) [MAXIMAL_AVAILABILITY](#)
- const [SeatIndex_T](#) [DEFAULT_SEAT_INDEX](#)
- const std::string [DEFAULT_FARE_FAMILY_VALUE_TYPE](#)
- const std::string [DEFAULT_SEGMENT_CABIN_VALUE_TYPE](#)
- const [DatePeriod_T](#) [BOOST_DEFAULT_DATE_PERIOD](#)
- const [DOW_String_T](#) [DEFAULT_DOW_STRING](#)
- const [DateOffset_T](#) [DEFAULT_DATE_OFFSET](#)
- const [PartySize_T](#) [DEFAULT_PARTY_SIZE](#)
- const [DayDuration_T](#) [DEFAULT_STAY_DURATION](#)
- const [WTP_T](#) [DEFAULT_WTP](#)
- const [CityCode_T](#) [DEFAULT_POS](#)
- const [Date_T](#) [DEFAULT_PREFERRED_DEPARTURE_DATE](#)
- const [Duration_T](#) [DEFAULT_PREFERRED_DEPARTURE_TIME](#)

- const [DateOffset_T](#) DEFAULT_ADVANCE_PURCHASE
- const [Date_T](#) DEFAULT_REQUEST_DATE
- const [Duration_T](#) DEFAULT_REQUEST_TIME
- const [DateTime_T](#) DEFAULT_REQUEST_DATE_TIME
- const [CabinCode_T](#) DEFAULT_PREFERRED_CABIN
- const [ChannelLabel_T](#) CHANNEL_DN
- const [ChannelLabel_T](#) CHANNEL_IN
- const [TripType_T](#) TRIP_TYPE_ONE_WAY
- const [TripType_T](#) TRIP_TYPE_ROUND_TRIP
- const [TripType_T](#) TRIP_TYPE_INBOUND
- const [TripType_T](#) TRIP_TYPE_OUTBOUND
- const [FrequentFlyer_T](#) DEFAULT_FF_TIER
- const [PriceValue_T](#) DEFAULT_VALUE_OF_TIME
- const [Duration_T](#) DEFAULT_MINIMAL_CONNECTION_TIME
- const [Duration_T](#) DEFAULT_MAXIMAL_CONNECTION_TIME
- const [FlightPathCode_T](#) DEFAULT_FLIGHTPATH_CODE
- const [Availability_T](#) DEFAULT_CLASS_AVAILABILITY
- const [AvailabilityStatus_T](#) DEFAULT_AVAILABILITY_STATUS
- const unsigned short DEFAULT_NUMBER_OF_REQUIRED_SEATS
- const [MatchingIndicator_T](#) DEFAULT_MATCHING_INDICATOR
- const [AirlineCode_T](#) DEFAULT_DICO_STUDIED_AIRLINE
- const [Yield_T](#) DEFAULT_YIELD_VALUE
- const [Yield_T](#) DEFAULT_YIELD_MAX_VALUE

33.5.1 Detailed Description

Handle on the StdAir library context.

Author

Anh Quan Nguyen <quannaus@users.sourceforge.net>

Date

20/01/2010 StdAir aims at providing a clean API, and the corresponding C++ implementation, for the basis of Airline IT Business Object Model (BOM), that is, to be used by several other Open Source projects, such as RMOL and OpenTREP.

Install the StdAir library for Airline IT Standard C++ fundamentals.

33.5.2 Typedef Documentation

33.5.2.1 typedef [date_time_element](#)<0, 23> [stdair::hour_t](#)

Type definitions for the date and time elements.

Definition at line 61 of file [BasParserHelperTypes.hpp](#).

33.5.2.2 `typedef date_time_element<0, 59> stdair::minute_t`

Definition at line 62 of file [BasParserHelperTypes.hpp](#).

33.5.2.3 `typedef date_time_element<0, 59> stdair::second_t`

Definition at line 63 of file [BasParserHelperTypes.hpp](#).

33.5.2.4 `typedef date_time_element<1900, 2100> stdair::year_t`

Definition at line 64 of file [BasParserHelperTypes.hpp](#).

33.5.2.5 `typedef date_time_element<1, 12> stdair::month_t`

Definition at line 65 of file [BasParserHelperTypes.hpp](#).

33.5.2.6 `typedef date_time_element<1, 31> stdair::day_t`

Definition at line 66 of file [BasParserHelperTypes.hpp](#).

33.5.2.7 `typedef std::istreambuf_iterator<char> stdair::base_iterator_t`

Definition at line 26 of file [BasParserTypes.hpp](#).

33.5.2.8 `typedef boost::spirit::multi_pass<base_iterator_t> stdair::iterator_t`

Definition at line 27 of file [BasParserTypes.hpp](#).

33.5.2.9 `typedef boost::spirit::qi::int_parser<unsigned int, 10, 1, 1> stdair::int1_p_t`

1-digit-integer parser

Definition at line 35 of file [BasParserTypes.hpp](#).

33.5.2.10 `typedef boost::spirit::qi::uint_parser<int, 10, 2, 2> stdair::uint2_p_t`

2-digit-integer parser

Definition at line 38 of file [BasParserTypes.hpp](#).

33.5.2.11 `typedef boost::spirit::qi::uint_parser<int, 10, 4, 4> stdair::uint4_p_t`

4-digit-integer parser

Definition at line 41 of file [BasParserTypes.hpp](#).

33.5.2.12 `typedef boost::spirit::qi::uint_parser<int, 10, 1, 4> stdair::uint1_4_p_t`

Up-to-4-digit-integer parser

Definition at line 44 of file [BasParserTypes.hpp](#).

33.5.2.13 `typedef boost::spirit::qi::uint_parser<hour_t, 10, 2, 2> stdair::hour_p_t`

Date & time element parsers.

Definition at line 47 of file [BasParserTypes.hpp](#).

33.5.2.14 `typedef boost::spirit::qi::uint_parser<minute_t, 10, 2, 2> stdair::minute_p_t`

Definition at line 48 of file [BasParserTypes.hpp](#).

33.5.2.15 `typedef boost::spirit::qi::uint_parser<second_t, 10, 2, 2> stdair::second_p_t`

Definition at line 49 of file [BasParserTypes.hpp](#).

33.5.2.16 `typedef boost::spirit::qi::uint_parser<year_t, 10, 4, 4> stdair::year_p_t`

Definition at line 50 of file [BasParserTypes.hpp](#).

33.5.2.17 `typedef boost::spirit::qi::uint_parser<month_t, 10, 2, 2> stdair::month_p_t`

Definition at line 51 of file [BasParserTypes.hpp](#).

33.5.2.18 `typedef boost::spirit::qi::uint_parser<day_t, 10, 2, 2> stdair::day_p_t`

Definition at line 52 of file [BasParserTypes.hpp](#).

33.5.2.19 `typedef unsigned short stdair::DictionaryKey_T`

Dictionary key.

Definition at line 17 of file [DictionaryManager.hpp](#).

33.5.2.20 `typedef std::list<AirlineClassList*> stdair::AirlineClassListList_T`

Define the segment-features list.

Definition at line 17 of file [AirlineClassListTypes.hpp](#).

33.5.2.21 `typedef std::map<const MapKey_T, AirlineClassList*>
stdair::AirlineClassListMap_T`

Define the segment-features map.

Definition at line 23 of file [AirlineClassListTypes.hpp](#).

33.5.2.22 `typedef std::pair<MapKey_T, AirlineClassList*>
stdair::AirlineClassListWithKey_T`

Define the list of pair<MapKey_T, AirlineCodeList>.

Definition at line 26 of file [AirlineClassListTypes.hpp](#).

33.5.2.23 `typedef std::list<AirlineClassListWithKey_T>
stdair::AirlineClassListDetailedList_T`

Definition at line 27 of file [AirlineClassListTypes.hpp](#).

33.5.2.24 `typedef std::list<AirlineFeature*> stdair::AirlineFeatureList_T`

Define the airline feature list.

Definition at line 17 of file [AirlineFeatureTypes.hpp](#).

33.5.2.25 `typedef std::map<const MapKey_T, AirlineFeature*>
stdair::AirlineFeatureMap_T`

Define the airline feature map.

Definition at line 23 of file [AirlineFeatureTypes.hpp](#).

33.5.2.26 `typedef std::list<AirportPair*> stdair::AirportPairList_T`

Define the airport-pair list.

Definition at line 17 of file [AirportPairTypes.hpp](#).

33.5.2.27 `typedef std::map<const MapKey_T, AirportPair*> stdair::AirportPairMap_T`

Define the airport-pair map.

Definition at line 23 of file [AirportPairTypes.hpp](#).

33.5.2.28 `typedef std::pair<MapKey_T, AirportPair*> stdair::AirportPairWithKey_T`

Define the list of pair<MapKey_T, AirportPair>.

Definition at line 26 of file [AirportPairTypes.hpp](#).

33.5.2.29 `typedef std::list<AirportPairWithKey_T> stdair::AirportPairDetailedList_T`

Definition at line 27 of file [AirportPairTypes.hpp](#).

33.5.2.30 `typedef std::map<const std::type_info*, BomAbstract*> stdair::HolderMap_T`

Definition at line 48 of file [BomAbstract.hpp](#).

33.5.2.31 `typedef boost::tokenizer< boost::char_separator< char > > stdair::Tokeniser_T`

Boost Tokeniser.

Definition at line 27 of file [BomKeyManager.cpp](#).

33.5.2.32 `typedef std::list<BookingClass*> stdair::BookingClassList_T`

Define the booking class list.

Definition at line 17 of file [BookingClassTypes.hpp](#).

33.5.2.33 `typedef std::map<const MapKey_T, BookingClass*>
stdair::BookingClassMap_T`

Define the booking class map.

Definition at line 23 of file [BookingClassTypes.hpp](#).

```
33.5.2.34  typedef boost::shared_ptr<BookingRequestStruct>
          stdair::BookingRequestPtr_T
```

Define the smart pointer to a booking request.

Definition at line 14 of file [BookingRequestTypes.hpp](#).

```
33.5.2.35  typedef std::string stdair::DemandGeneratorKey_T
```

Define the hash key for the demand generator.

Definition at line 21 of file [BookingRequestTypes.hpp](#).

```
33.5.2.36  typedef std::list<Bucket*> stdair::BucketList_T
```

Define the bucket list.

Definition at line 17 of file [BucketTypes.hpp](#).

```
33.5.2.37  typedef std::map<const MapKey_T, Bucket*> stdair::BucketMap_T
```

Define the bucket map.

Definition at line 23 of file [BucketTypes.hpp](#).

```
33.5.2.38  typedef boost::shared_ptr<CancellationStruct> stdair::CancellationPtr_T
```

Define the smart pointer to a cancellation .

Definition at line 14 of file [CancellationTypes.hpp](#).

```
33.5.2.39  typedef std::list<DatePeriod*> stdair::DatePeriodList_T
```

Define the date-period list.

Definition at line 17 of file [DatePeriodTypes.hpp](#).

```
33.5.2.40  typedef std::map<const MapKey_T, DatePeriod*> stdair::DatePeriodMap_T
```

Define the date-period map.

Definition at line 23 of file [DatePeriodTypes.hpp](#).

```
33.5.2.41  typedef std::pair<MapKey_T, DatePeriod*> stdair::DatePeriodWithKey_T
```

Define the list of pair<MapKey_T, DatePeriod>.

Definition at line 26 of file [DatePeriodTypes.hpp](#).

```
33.5.2.42  typedef std::list<DatePeriodWithKey_T> stdair::DatePeriodDetailedList_T
```

Definition at line 27 of file [DatePeriodTypes.hpp](#).

33.5.2.43 `typedef std::list<EventQueue*> stdair::EventQueueList_T`

Define the [EventQueue](#) list.

Definition at line 17 of file [EventQueueTypes.hpp](#).

33.5.2.44 `typedef std::map<const MapKey_T, EventQueue*>
stdair::EventQueueMap_T`

Define the [EventQueue](#) map.

Definition at line 23 of file [EventQueueTypes.hpp](#).

33.5.2.45 `typedef std::pair<const LongDuration_T, EventStruct>
stdair::EventListElement_T`

Define an element of a event list.

Definition at line 22 of file [EventTypes.hpp](#).

33.5.2.46 `typedef std::map<const LongDuration_T, EventStruct> stdair::EventList_T`

Define a list of events.

Definition at line 32 of file [EventTypes.hpp](#).

33.5.2.47 `typedef std::list<FareFamily*> stdair::FareFamilyList_T`

Define the fare family list.

Definition at line 17 of file [FareFamilyTypes.hpp](#).

33.5.2.48 `typedef std::map<const MapKey_T, FareFamily*> stdair::FareFamilyMap_T`

Define the fare family map.

Definition at line 23 of file [FareFamilyTypes.hpp](#).

33.5.2.49 `typedef std::list<FareFeatures*> stdair::FareFeaturesList_T`

Define the date-period list.

Definition at line 17 of file [FareFeaturesTypes.hpp](#).

33.5.2.50 `typedef std::map<const MapKey_T, FareFeatures*>
stdair::FareFeaturesMap_T`

Define the date-period map.

Definition at line 23 of file [FareFeaturesTypes.hpp](#).

33.5.2.51 `typedef std::pair<MapKey_T, FareFeatures*>
stdair::FareFeaturesWithKey_T`

Define the list of pair<MapKey_T, FareFeatures>.

Definition at line 26 of file [FareFeaturesTypes.hpp](#).

33.5.2.52 `typedef std::list<FareFeaturesWithKey_T>
stdair::FareFeaturesDetailedList_T`

Definition at line 27 of file [FareFeaturesTypes.hpp](#).

33.5.2.53 `typedef std::list<FareOptionStruct> stdair::FareOptionList_T`

Define the booking class list.

Definition at line 18 of file [FareOptionTypes.hpp](#).

33.5.2.54 `typedef std::list<FlightDate*> stdair::FlightDateList_T`

Define the flight-date list.

Definition at line 17 of file [FlightDateTypes.hpp](#).

33.5.2.55 `typedef std::map<const MapKey_T, FlightDate*> stdair::FlightDateMap_T`

Define the flight-date map.

Definition at line 24 of file [FlightDateTypes.hpp](#).

33.5.2.56 `typedef std::list<FlightPeriod*> stdair::FlightPeriodList_T`

Define the flight-period list.

Definition at line 17 of file [FlightPeriodTypes.hpp](#).

33.5.2.57 `typedef std::map<const MapKey_T, FlightPeriod*>
stdair::FlightPeriodMap_T`

Define the flight-period map.

Definition at line 23 of file [FlightPeriodTypes.hpp](#).

33.5.2.58 `typedef std::list<GuillotineBlock*> stdair::GuillotineBlockList_T`

Define the guillotine-block list.

Definition at line 20 of file [GuillotineBlockTypes.hpp](#).

33.5.2.59 `typedef std::map<const MapKey_T, GuillotineBlock*>
stdair::GuillotineBlockMap_T`

Define the guillotine-block map.

Definition at line 27 of file [GuillotineBlockTypes.hpp](#).

33.5.2.60 `typedef std::map<const SegmentCabin*, BlockNumber_T>
stdair::SegmentCabinIndexMap_T`

Define the map between the segment-cabins and the block number.

Definition at line 30 of file [GuillotineBlockTypes.hpp](#).

33.5.2.61 `typedef std::map<const MapKey_T, BlockIndex_T>
stdair::ValueTypeIndexMap_T`

Define the map between the value type of the snapshots and their index.

Definition at line 33 of file [GuillotineBlockTypes.hpp](#).

33.5.2.62 `typedef std::list<Inventory*> stdair::InventoryList_T`

Define the [Inventory](#) list.

Definition at line 17 of file [InventoryTypes.hpp](#).

33.5.2.63 `typedef std::map<const MapKey_T, Inventory*> stdair::InventoryMap_T`

Define the [Inventory](#) map.

Definition at line 23 of file [InventoryTypes.hpp](#).

33.5.2.64 `typedef std::string stdair::MapKey_T`

Key of a STL map.

Definition at line 15 of file [key_types.hpp](#).

33.5.2.65 `typedef std::list<std::string> stdair::KeyList_T`

List of keys.

Definition at line 18 of file [key_types.hpp](#).

33.5.2.66 `typedef std::list<LegCabin*> stdair::LegCabinList_T`

Define the leg-cabin list.

Definition at line 17 of file [LegCabinTypes.hpp](#).

33.5.2.67 `typedef std::map<const MapKey_T, LegCabin*> stdair::LegCabinMap_T`

Define the leg-cabin map.

Definition at line 23 of file [LegCabinTypes.hpp](#).

33.5.2.68 `typedef std::list<LegDate*> stdair::LegDateList_T`

Define the leg-date list.

Definition at line 17 of file [LegDateTypes.hpp](#).

33.5.2.69 `typedef std::map<const MapKey_T, LegDate*> stdair::LegDateMap_T`

Define the leg-date map.

Definition at line 23 of file [LegDateTypes.hpp](#).

33.5.2.70 `typedef std::list<OnDDate*> stdair::OnDDateList_T`

Define the O&D date list.

Definition at line 19 of file [OnDDateTypes.hpp](#).

33.5.2.71 `typedef std::map<const MapKey_T, OnDDate*> stdair::OnDDateMap_T`

Define the OnD date map.

Definition at line 25 of file [OnDDateTypes.hpp](#).

33.5.2.72 `typedef std::pair<std::string, YieldDemandPair_T>
stdair::StringDemandStructPair_T`

Define the yield mean and standard deviation for a certain cabin/class path. This map is mandatory when using the default BOM tree. This map can be empty if yields are charged otherwise (input file, ...)

Definition at line 32 of file [OnDDateTypes.hpp](#).

33.5.2.73 `typedef std::map<std::string, YieldDemandPair_T>
stdair::StringDemandStructMap_T`

Definition at line 33 of file [OnDDateTypes.hpp](#).

33.5.2.74 `typedef std::map<std::string, CabinClassPairList_T>
stdair::StringCabinClassPairListMap_T`

Define the string matching a (cabin,class) path. (i.e, the string is "Y:M;" for a one leg O&D with the cabin Y and the class M; the string is "Y:M;Y:Y;" for a two legs O&D with the cabin Y and the class M for the first leg, and the cabin Y and the class Y for the second leg).

Definition at line 41 of file [OnDDateTypes.hpp](#).

33.5.2.75 `typedef std::pair<std::string, CabinClassPairList_T>
stdair::StringCabinClassPair_T`

Definition at line 42 of file [OnDDateTypes.hpp](#).

33.5.2.76 `typedef std::map<CabinCode_T, WTPDemandPair_T>
stdair::CabinForecastMap_T`

Define the WTP mean and standard deviation for a certain cabin code. This information is needed to forecast O&D demand per cabin.

Definition at line 48 of file [OnDDateTypes.hpp](#).

33.5.2.77 `typedef std::pair<CabinCode_T, WTPDemandPair_T>
stdair::CabinForecastPair_T`

Definition at line 49 of file [OnDDateTypes.hpp](#).

33.5.2.78 `typedef boost::shared_ptr<OptimisationNotificationStruct>
stdair::OptimisationNotificationPtr_T`

Define the smart pointer to a optimisation notification.

Definition at line 14 of file [OptimisationNotificationTypes.hpp](#).

33.5.2.79 `typedef std::list<PosChannel*> stdair::PosChannelList_T`

Define the fare-point_of_sale list.

Definition at line 17 of file [PosChannelTypes.hpp](#).

33.5.2.80 `typedef std::map<const MapKey_T, PosChannel*>
stdair::PosChannelMap_T`

Define the fare-point_of_sale map.

Definition at line 23 of file [PosChannelTypes.hpp](#).

33.5.2.81 `typedef std::pair<MapKey_T, PosChannel*>
stdair::PosChannelWithKey_T`

Define the list of pair<MapKey_T, PosChannel>.

Definition at line 26 of file [PosChannelTypes.hpp](#).

33.5.2.82 `typedef std::list<PosChannelWithKey_T>
stdair::PosChannelDetailedList_T`

Definition at line 27 of file [PosChannelTypes.hpp](#).

33.5.2.83 `typedef boost::shared_ptr<RMEventStruct> stdair::RMEventPtr_T`

Define the smart pointer to a RM event .

Definition at line 16 of file [RMEventTypes.hpp](#).

33.5.2.84 `typedef std::list<RMEventStruct> stdair::RMEventList_T`

Define the list of RM events.

Definition at line 23 of file [RMEventTypes.hpp](#).

33.5.2.85 `typedef std::list<SegmentCabin*> stdair::SegmentCabinList_T`

Define the segment-cabin list.

Definition at line 17 of file [SegmentCabinTypes.hpp](#).

33.5.2.86 `typedef std::map<const MapKey_T, SegmentCabin*>
stdair::SegmentCabinMap_T`

Define the segment-cabin map.

Definition at line 23 of file [SegmentCabinTypes.hpp](#).

33.5.2.87 `typedef std::list<SegmentDate*> stdair::SegmentDateList_T`

Define the segment-date list.

Definition at line 17 of file [SegmentDateTypes.hpp](#).

33.5.2.88 `typedef std::map<const MapKey_T, SegmentDate*>
stdair::SegmentDateMap_T`

Define the segment-date map.

Definition at line 23 of file [SegmentDateTypes.hpp](#).

33.5.2.89 `typedef std::list<SegmentPeriod*> stdair::SegmentPeriodList_T`

Define the segment-period list.

Definition at line 17 of file [SegmentPeriodTypes.hpp](#).

33.5.2.90 `typedef std::map<const MapKey_T, SegmentPeriod*>
stdair::SegmentPeriodMap_T`

Define the segment-period map.

Definition at line 23 of file [SegmentPeriodTypes.hpp](#).

33.5.2.91 `typedef std::pair<MapKey_T, SegmentPeriod*>
stdair::SegmentPeriodWithKey_T`

Define the list of pair<MapKey_T, SegmentPeriod>.

Definition at line 26 of file [SegmentPeriodTypes.hpp](#).

33.5.2.92 `typedef std::list<SegmentPeriodWithKey_T>
stdair::SegmentPeriodDetailedList_T`

Definition at line 27 of file [SegmentPeriodTypes.hpp](#).

33.5.2.93 `typedef boost::shared_ptr<SnapshotStruct> stdair::SnapshotPtr_T`

Define the smart pointer to a snapshot .

Definition at line 14 of file [SnapshotTypes.hpp](#).

33.5.2.94 `typedef std::list<TimePeriod*> stdair::TimePeriodList_T`

Define the time-period list.

Definition at line 17 of file [TimePeriodTypes.hpp](#).

33.5.2.95 `typedef std::map<const MapKey_T, TimePeriod*>
stdair::TimePeriodMap_T`

Define the time-period map.

Definition at line 23 of file [TimePeriodTypes.hpp](#).

33.5.2.96 `typedef std::pair<MapKey_T, TimePeriod*> stdair::TimePeriodWithKey_T`

Define the list of `pair<MapKey_T, TimePeriod>`.

Definition at line 26 of file [TimePeriodTypes.hpp](#).

33.5.2.97 `typedef std::list<TimePeriodWithKey_T> stdair::TimePeriodDetailedList_T`

Definition at line 27 of file [TimePeriodTypes.hpp](#).

33.5.2.98 `typedef std::list<TravelSolutionStruct> stdair::TravelSolutionList_T`

Define the booking class list.

Definition at line 19 of file [TravelSolutionTypes.hpp](#).

33.5.2.99 `typedef KeyList_T stdair::SegmentPath_T`

Define the segment path key.

Definition at line 25 of file [TravelSolutionTypes.hpp](#).

33.5.2.100 `typedef std::list<SegmentPath_T> stdair::SegmentPathList_T`

Define the list of segment paths.

Definition at line 28 of file [TravelSolutionTypes.hpp](#).

33.5.2.101 `typedef std::map<const ClassCode_T, Availability_T>
stdair::ClassAvailabilityMap_T`

Define booking class - availability map.

Definition at line 31 of file [TravelSolutionTypes.hpp](#).

33.5.2.102 `typedef std::list<ClassAvailabilityMap_T>
stdair::ClassAvailabilityMapHolder_T`

Define list of booking class - availability maps.

Definition at line 34 of file [TravelSolutionTypes.hpp](#).

33.5.2.103 `typedef std::map<const ClassCode_T, YieldValue_T>
stdair::ClassYieldMap_T`

Define booking class - yield map.

Definition at line 37 of file [TravelSolutionTypes.hpp](#).

33.5.2.104 `typedef std::list<ClassYieldMap_T> stdair::ClassYieldMapHolder_T`

Define list of booking class - yield maps.

Definition at line 40 of file [TravelSolutionTypes.hpp](#).

33.5.2.105 `typedef std::list<BidPriceVector_T> stdair::BidPriceVectorHolder_T`

Define list of bid price vectors.

Definition at line 43 of file [TravelSolutionTypes.hpp](#).

33.5.2.106 `typedef std::map<const ClassCode_T, const BidPriceVector_T*>
stdair::ClassBpvMap_T`

Define booking class - bid price reference map.

Definition at line 46 of file [TravelSolutionTypes.hpp](#).

33.5.2.107 `typedef std::list<ClassBpvMap_T> stdair::ClassBpvMapHolder_T`

Define list of booking class - bid price reference maps.

Definition at line 49 of file [TravelSolutionTypes.hpp](#).

33.5.2.108 `typedef std::list<VirtualClassStruct> stdair::VirtualClassList_T`

Define the booking class list.

Definition at line 17 of file [VirtualClassTypes.hpp](#).

33.5.2.109 `typedef std::map<const Yield_T, VirtualClassStruct>
stdair::VirtualClassMap_T`

Define the booking class map.

Definition at line 23 of file [VirtualClassTypes.hpp](#).

33.5.2.110 `typedef std::list<YieldFeatures*> stdair::YieldFeaturesList_T`

Define the date-period list.

Definition at line 17 of file [YieldFeaturesTypes.hpp](#).

33.5.2.111 `typedef std::map<const MapKey_T, YieldFeatures*>
stdair::YieldFeaturesMap_T`

Define the date-period map.

Definition at line 23 of file [YieldFeaturesTypes.hpp](#).

33.5.2.112 `typedef std::pair<MapKey_T, YieldFeatures*>
stdair::YieldFeaturesWithKey_T`

Define the list of pair<MapKey_T, YieldFeatures>.

Definition at line 26 of file [YieldFeaturesTypes.hpp](#).

33.5.2.113 `typedef std::list<YieldFeaturesWithKey_T>
stdair::YieldFeaturesDetailedList_T`

Definition at line 27 of file [YieldFeaturesTypes.hpp](#).

33.5.2.114 `typedef std::list<YieldStore*> stdair::YieldStoreList_T`

Define the [Inventory](#) list.

Definition at line 17 of file [YieldStoreTypes.hpp](#).

33.5.2.115 `typedef std::map<const MapKey_T, YieldStore*> stdair::YieldStoreMap_T`

Define the [Inventory](#) map.

Definition at line 23 of file [YieldStoreTypes.hpp](#).

33.5.2.116 `typedef std::string stdair::LocationCode_T`

Location code (3-letter-code, e.g., LON).

Definition at line 16 of file [stdair_basic_types.hpp](#).

33.5.2.117 `typedef unsigned long int stdair::Distance_T`

Define a distance (kilometers).

Definition at line 19 of file [stdair_basic_types.hpp](#).

33.5.2.118 `typedef LocationCode_T stdair::AirportCode_T`

Define the Airport Code type (3-letter-code, e.g., LHR).

Definition at line 22 of file [stdair_basic_types.hpp](#).

33.5.2.119 `typedef LocationCode_T stdair::CityCode_T`

City code

Definition at line 25 of file [stdair_basic_types.hpp](#).

33.5.2.120 `typedef std::string stdair::KeyDescription_T`

Define the key description.

Definition at line 28 of file [stdair_basic_types.hpp](#).

33.5.2.121 `typedef std::string stdair::AirlineCode_T`

Define the Airline Code type (2-letter-code, e.g., BA).

Definition at line 31 of file [stdair_basic_types.hpp](#).

33.5.2.122 `typedef unsigned short stdair::FlightNumber_T`

Define the type for flight numbers.

Definition at line 34 of file [stdair_basic_types.hpp](#).

33.5.2.123 `typedef unsigned short stdair::GuillotineNumber_T`

Define the type for guillotine numbers.

Definition at line 37 of file [stdair_basic_types.hpp](#).

33.5.2.124 `typedef std::string stdair::CabinCode_T`

Define the cabin code (class of service, e.g., first, business, economy).

Definition at line 41 of file [stdair_basic_types.hpp](#).

33.5.2.125 `typedef std::string stdair::FamilyCode_T`

Define the code of the fare family (e.g., 1, 2, 3, etc.).

Definition at line 44 of file [stdair_basic_types.hpp](#).

33.5.2.126 `typedef std::string stdair::ClassCode_T`

Define the booking class code (product segment class, e.g., H, B, K, etc.).

Definition at line 48 of file [stdair_basic_types.hpp](#).

33.5.2.127 `typedef unsigned long stdair::Identity_T`

Define a identity number.

Definition at line 51 of file [stdair_basic_types.hpp](#).

33.5.2.128 `typedef std::string stdair::TripType_T`

Type of trip type (RO=outbound of round-trip, RI=inbound of round-trip, OW=one way).

Definition at line 55 of file [stdair_basic_types.hpp](#).

33.5.2.129 `typedef double stdair::MonetaryValue_T`

Monetary value

Definition at line 58 of file [stdair_basic_types.hpp](#).

33.5.2.130 `typedef double stdair::RealNumber_T`

Real number

Definition at line 61 of file [stdair_basic_types.hpp](#).

33.5.2.131 `typedef double stdair::Percentage_T`

Define a percentage value (between 0 and 100%).

Definition at line 64 of file [stdair_basic_types.hpp](#).

33.5.2.132 `typedef double stdair::PriceValue_T`

Define a price value (e.g., 1000.0 Euros).

Definition at line 67 of file [stdair_basic_types.hpp](#).

33.5.2.133 typedef double stdair::YieldValue_T

Define a yield value (e.g., 1000.0 Euros).

Definition at line 70 of file [stdair_basic_types.hpp](#).

33.5.2.134 typedef std::string stdair::PriceCurrency_T

Define a price currency (e.g., EUR for Euros).

Definition at line 73 of file [stdair_basic_types.hpp](#).

33.5.2.135 typedef double stdair::Revenue_T

Define an amount of revenue.

Definition at line 76 of file [stdair_basic_types.hpp](#).

33.5.2.136 typedef double stdair::Multiplier_T

Define the name of a multiplier.

Definition at line 79 of file [stdair_basic_types.hpp](#).

33.5.2.137 typedef double stdair::NbOfSeats_T

Define the number of seats (it can be non integer, because the overbooking can be applied at booking class or PNR level).

Definition at line 83 of file [stdair_basic_types.hpp](#).

33.5.2.138 typedef unsigned int stdair::Count_T

Count

Definition at line 86 of file [stdair_basic_types.hpp](#).

33.5.2.139 typedef short stdair::PartySize_T

Number of passengers (in a group) for a booking.

Definition at line 89 of file [stdair_basic_types.hpp](#).

33.5.2.140 typedef double stdair::NbOfRequests_T

Define a number of requests.

Definition at line 92 of file [stdair_basic_types.hpp](#).

33.5.2.141 typedef NbOfRequests_T stdair::NbOfBookings_T

Define a number of bookings.

Definition at line 95 of file [stdair_basic_types.hpp](#).

33.5.2.142 typedef NbOfRequests_T stdair::NbOfCancellations_T

Define a number of cancellations.

Define a number of cancellations (travellers).

Definition at line 98 of file [stdair_basic_types.hpp](#).

33.5.2.143 typedef unsigned short stdair::NbOfTravelSolutions_T

Define a number of travel solutions (in a travel solution block).

Definition at line 102 of file [stdair_basic_types.hpp](#).

33.5.2.144 typedef std::string stdair::ClassList_String_T

Define the list of class codes as a string.

Definition at line 105 of file [stdair_basic_types.hpp](#).

33.5.2.145 typedef unsigned short stdair::NbOfSegments_T

Define a number of segment-dates (in a path).

Definition at line 108 of file [stdair_basic_types.hpp](#).

33.5.2.146 typedef unsigned short stdair::NbOfAirlines_T

Define a number of airlines (in a path).

Definition at line 111 of file [stdair_basic_types.hpp](#).

33.5.2.147 typedef double stdair::Availability_T

Define an availability.

Definition at line 114 of file [stdair_basic_types.hpp](#).

33.5.2.148 typedef double stdair::Fare_T

Define the price of a travel solution.

Definition at line 117 of file [stdair_basic_types.hpp](#).

33.5.2.149 typedef bool stdair::Flag_T

Define the censorship flag.

Definition at line 120 of file [stdair_basic_types.hpp](#).

33.5.2.150 typedef unsigned int stdair::UnsignedIndex_T

Define the unsigned index type.

Definition at line 123 of file [stdair_basic_types.hpp](#).

33.5.2.151 typedef std::string stdair::Filename_T

File or directory name.

It may contain paths, relative or absolute (e.g., /foo/bar or C:).

Definition at line 129 of file [stdair_basic_types.hpp](#).

33.5.2.152 typedef std::string stdair::FileAddress_T

Define the file address type (e.g. "a_directory/a_filename").

NOTE: That type should be deprecated.

Definition at line 133 of file [stdair_basic_types.hpp](#).

33.5.2.153 typedef float stdair::ProgressPercentage_T

Progress status (usually, a percentage expressed as a floating point number).

Definition at line 137 of file [stdair_basic_types.hpp](#).

33.5.2.154 typedef boost::posix_time::time_duration stdair::Duration_T

Define the type for durations (e.g., elapsed in-flight time).

Definition at line 17 of file [stdair_date_time_types.hpp](#).

33.5.2.155 typedef boost::gregorian::date stdair::Date_T

Define the type for date (e.g., departure date of a flight).

Definition at line 20 of file [stdair_date_time_types.hpp](#).

33.5.2.156 typedef boost::posix_time::time_duration stdair::Time_T

Time

Definition at line 23 of file [stdair_date_time_types.hpp](#).

33.5.2.157 typedef boost::posix_time::ptime stdair::DateTime_T

Define an accurate time (date+time).

Definition at line 26 of file [stdair_date_time_types.hpp](#).

33.5.2.158 typedef boost::gregorian::date_period stdair::DatePeriod_T

Define the Period (e.g., period during which flights depart).

Definition at line 29 of file [stdair_date_time_types.hpp](#).

33.5.2.159 typedef std::string stdair::DOW_String_T

Define the Day-Of-the-Week as a string.

Definition at line 32 of file [stdair_date_time_types.hpp](#).

33.5.2.160 `typedef boost::gregorian::date_duration stdair::DateOffset_T`

Define the Date Offset (e.g., -1).

Definition at line 35 of file [stdair_date_time_types.hpp](#).

33.5.2.161 `typedef unsigned int stdair::DayDuration_T`

Define a duration in number of days.

Definition at line 38 of file [stdair_date_time_types.hpp](#).

33.5.2.162 `typedef bool stdair::SaturdayStay_T`

Define the Saturday stay status of a travel.

Define the saturday stay of a tickets.

Definition at line 41 of file [stdair_date_time_types.hpp](#).

33.5.2.163 `typedef long int stdair::IntDuration_T`

Time duration in (integer) number of seconds

Definition at line 44 of file [stdair_date_time_types.hpp](#).

33.5.2.164 `typedef unsigned long long int stdair::LongDuration_T`

Time duration in (unsigned long long integer) number of milliseconds

Definition at line 47 of file [stdair_date_time_types.hpp](#).

33.5.2.165 `typedef float stdair::FloatDuration_T`

Duration in (float) number of time units

Definition at line 50 of file [stdair_date_time_types.hpp](#).

33.5.2.166 `typedef soci::session stdair::DBSession_T`

Database session handler.

Definition at line 20 of file [stdair_db.hpp](#).

33.5.2.167 `typedef soci::statement stdair::DBRequestStatement_T`

Database request statement handler.

Definition at line 23 of file [stdair_db.hpp](#).

33.5.2.168 `typedef std::string stdair::DBConnectionName_T`

Define the name of an database connection.

Definition at line 26 of file [stdair_db.hpp](#).

33.5.2.169 typedef bool stdair::ChangeFees_T

Define the availability option allowing the ticket change.

Definition at line 29 of file [stdair_demand_types.hpp](#).

33.5.2.170 typedef bool stdair::NonRefundable_T

Define the refundable availability of a tickets.

Definition at line 32 of file [stdair_demand_types.hpp](#).

33.5.2.171 typedef unsigned int stdair::SaturdayStayRatio_T

Define the average ratio (between 0 and 100) of demand with a saturday stay status equal to TRUE.

Definition at line 39 of file [stdair_demand_types.hpp](#).

33.5.2.172 typedef unsigned int stdair::ChangeFeesRatio_T

Define the average ratio of demand with change fee availability.

Definition at line 43 of file [stdair_demand_types.hpp](#).

33.5.2.173 typedef unsigned int stdair::NonRefundableRatio_T

Define the average ratio of demand with non-refundable availability.

Definition at line 47 of file [stdair_demand_types.hpp](#).

33.5.2.174 typedef std::string stdair::PassengerType_T

Define the passenger characteristics, leisure or business for instance (1-letter-code, e.g., L or B).

Definition at line 51 of file [stdair_demand_types.hpp](#).

33.5.2.175 typedef std::string stdair::DistributionPatternId_T

Define the identifier of a distribution pattern (e.g., 1).

Definition at line 54 of file [stdair_demand_types.hpp](#).

33.5.2.176 typedef std::string stdair::CancellationRateCurveId_T

Define the identifier of a cancellation rate curve (e.g., C1).

Definition at line 57 of file [stdair_demand_types.hpp](#).

33.5.2.177 typedef std::string stdair::AirlinePreferenceld_T

Define the identifier of an airline preference set list (e.g., AP1).

Definition at line 60 of file [stdair_demand_types.hpp](#).

33.5.2.178 `typedef std::pair<Percentage_T, Percentage_T>
stdair::CancellationNoShowRatePair_T`

Define a cancellation & and no-show rate pair.

Definition at line 63 of file [stdair_demand_types.hpp](#).

33.5.2.179 `typedef std::string stdair::CharacteristicsPatternId_T`

Define the identifier of a demand characteristics pattern (e.g. Ch12); for a customer choice model

Definition at line 67 of file [stdair_demand_types.hpp](#).

33.5.2.180 `typedef std::string stdair::CharacteristicsIndex_T`

Define characteristics component index (e.g. W for WTP)

Definition at line 70 of file [stdair_demand_types.hpp](#).

33.5.2.181 `typedef double stdair::WTP_T`

Define a Willingness-To-Pay (WTP) (e.g., 1000.0 Euros).

Definition at line 73 of file [stdair_demand_types.hpp](#).

33.5.2.182 `typedef boost::tuples::tuple<double, WTP_T>
stdair::CharacteristicsWTP_tuple_T`

Define the name of a WTP-component of characteristics pattern.

Definition at line 76 of file [stdair_demand_types.hpp](#).

33.5.2.183 `typedef std::pair<WTP_T, MeanStdDevPair_T>
stdair::WTPDemandPair_T`

Define the <WTP, demand> pair type.

Definition at line 79 of file [stdair_demand_types.hpp](#).

33.5.2.184 `typedef NbOfRequests_T stdair::NbOfNoShows_T`

Define a number of no-shows.

Definition at line 85 of file [stdair_demand_types.hpp](#).

33.5.2.185 `typedef double stdair::MatchingIndicator_T`

Define a indicator of demand to class matching.

Definition at line 88 of file [stdair_demand_types.hpp](#).

33.5.2.186 `typedef std::string stdair::DemandStreamKeyStr_T`

Type definition for the hashed key of the DemandStreamKey object.

Definition at line 91 of file [stdair_demand_types.hpp](#).

33.5.2.187 typedef std::string stdair::ChannelLabel_T

Type of booking channel (D=direct, I=indirect, N=oNline, F=oFfline).

Definition at line 94 of file [stdair_demand_types.hpp](#).

33.5.2.188 typedef std::string stdair::FrequentFlyer_T

Type of frequent flyer (P=Platinum, G=Gold, S=Silver, M=Member, N=None).

Definition at line 97 of file [stdair_demand_types.hpp](#).

33.5.2.189 typedef std::string stdair::RequestStatus_T

Define the Request status for booking (1-letter-code, e.g., B: booked, C: cancelled, R: Rejected).

Definition at line 101 of file [stdair_demand_types.hpp](#).

33.5.2.190 typedef std::map<Identity_T, Identity_T> stdair::BookingTSIDMap_T

Define a map between a BookingID and a TravelSolutionID.

Definition at line 104 of file [stdair_demand_types.hpp](#).

33.5.2.191 typedef std::pair<CabinCode_T, ClassCode_T> stdair::CabinClassPair_T

Define a pair (cabin code, class code) e.g., (economy, K).

Definition at line 107 of file [stdair_demand_types.hpp](#).

33.5.2.192 typedef std::list<CabinClassPair_T> stdair::CabinClassPairList_T

Define a list of pair (cabin code, class code).

Definition at line 110 of file [stdair_demand_types.hpp](#).

33.5.2.193 typedef double stdair::ProportionFactor_T

Define the forecast booking requests proportion.

Definition at line 113 of file [stdair_demand_types.hpp](#).

33.5.2.194 typedef std::list<ProportionFactor_T> stdair::ProportionFactorList_T

Define the list of forecast booking requests proportions.

Definition at line 116 of file [stdair_demand_types.hpp](#).

33.5.2.195 typedef std::string stdair::OnDString_T

Define the O&D string key (e.g. "SQ;11,2010-Feb-08;SIN,BKK").

Definition at line 119 of file [stdair_demand_types.hpp](#).

33.5.2.196 `typedef std::list<OnDString_T> stdair::OnDStringList_T`

Define the list of O&D string key.

Definition at line 122 of file [stdair_demand_types.hpp](#).

33.5.2.197 `typedef std::string stdair::EventName_T`

Define the name of an event.

Definition at line 14 of file [stdair_event_types.hpp](#).

33.5.2.198 `typedef double stdair::NbOfEvents_T`

Define a number of events.

Definition at line 17 of file [stdair_event_types.hpp](#).

33.5.2.199 `typedef std::string stdair::EventQueueID_T`

Define an ID for an [EventQueue](#) object.

Definition at line 20 of file [stdair_event_types.hpp](#).

33.5.2.200 `typedef std::string stdair::EventGeneratorKey_T`

Define a key string of an event generator.

Definition at line 23 of file [stdair_event_types.hpp](#).

33.5.2.201 `typedef double stdair::NbOfFareRules_T`

Define a number of fare rules.

Definition at line 12 of file [stdair_fare_types.hpp](#).

33.5.2.202 `typedef std::string stdair::NetworkID_T`

Define the type for network ID.

Definition at line 23 of file [stdair_inventory_types.hpp](#).

33.5.2.203 `typedef std::vector<AirlineCode_T> stdair::AirlineCodeList_T`

Define a list of airline code.

Definition at line 26 of file [stdair_inventory_types.hpp](#).

33.5.2.204 `typedef std::vector<ClassList_String_T> stdair::ClassList_StringList_T`

Define the list of list of class codes as a string.

Definition at line 29 of file [stdair_inventory_types.hpp](#).

33.5.2.205 `typedef std::vector<ClassCode_T> stdair::ClassCodeList_T`

Define a list of class code.

Definition at line 32 of file [stdair_inventory_types.hpp](#).

33.5.2.206 `typedef unsigned short stdair::SubclassCode_T`

Define the sub-class code (e.g., 0, 1, 2, etc.). The subclass is a sub-structure for the booking class, allowing to have specific rules for some criteria like POS.

Definition at line 37 of file [stdair_inventory_types.hpp](#).

33.5.2.207 `typedef std::string stdair::FlightPathCode_T`

Define the flight path code (code made by a suite of flight numbers).

Definition at line 40 of file [stdair_inventory_types.hpp](#).

33.5.2.208 `typedef std::map<CabinCode_T, ClassList_String_T> stdair::CabinBookingClassMap_T`

Map between the cabin codes and the booking class codes within each cabin.

Definition at line 44 of file [stdair_inventory_types.hpp](#).

33.5.2.209 `typedef double stdair::CabinCapacity_T`

Define the cabin capacity (resource, e.g., 200 seats).

The capacity is expressed as a double to cope with overbooking.

Definition at line 48 of file [stdair_inventory_types.hpp](#).

33.5.2.210 `typedef double stdair::NbOfFlightDates_T`

Define a number of flight dates.

Definition at line 51 of file [stdair_inventory_types.hpp](#).

33.5.2.211 `typedef double stdair::CommittedSpace_T`

Define the committed space of a cabin.

Definition at line 54 of file [stdair_inventory_types.hpp](#).

33.5.2.212 `typedef double stdair::UPR_T`

Define the unsold protection (UPR).

Definition at line 57 of file [stdair_inventory_types.hpp](#).

33.5.2.213 `typedef double stdair::BookingLimit_T`

Define the value of the booking limit.

Define the Booking Limit.

It is a double, as it allows for overbooking.

Definition at line 60 of file [stdair_inventory_types.hpp](#).

33.5.2.214 typedef double stdair::AuthorizationLevel_T

Define the value of the authorization level.

Definition at line 63 of file [stdair_inventory_types.hpp](#).

33.5.2.215 typedef double stdair::CapacityAdjustment_T

Define the value of the adjustment for cabin capacity.

Definition at line 66 of file [stdair_inventory_types.hpp](#).

33.5.2.216 typedef double stdair::BlockSpace_T

Define the number of seat which could not be used for the booking.

Definition at line 69 of file [stdair_inventory_types.hpp](#).

33.5.2.217 typedef bool stdair::AvailabilityStatus_T

Define an availability status (AVS).

Definition at line 72 of file [stdair_inventory_types.hpp](#).

33.5.2.218 typedef std::vector<Availability_T> stdair::BucketAvailabilities_T

Define a list of availabilities.

Definition at line 75 of file [stdair_inventory_types.hpp](#).

33.5.2.219 typedef double stdair::NbOfYields_T

Define a number of yields.

Definition at line 78 of file [stdair_inventory_types.hpp](#).

33.5.2.220 typedef double stdair::NbOfInventoryControlRules_T

Define a number of InventoryControlRules.

Definition at line 81 of file [stdair_inventory_types.hpp](#).

33.5.2.221 typedef bool stdair::CensorshipFlag_T

Define availability of booking limit.

Definition at line 84 of file [stdair_inventory_types.hpp](#).

33.5.2.222 typedef short stdair::DTD_T

Define the type of day-to-departure.

Definition at line 87 of file [stdair_inventory_types.hpp](#).

33.5.2.223 typedef short stdair::DCP_T

Define the type of data collection point.

Definition at line 90 of file [stdair_inventory_types.hpp](#).

33.5.2.224 `typedef std::list<DCP_T> stdair::DCPList_T`

Define the type of data collection point list.

Definition at line 93 of file [stdair_inventory_types.hpp](#).

33.5.2.225 `typedef std::map<DTD_T, RealNumber_T> stdair::DTDFratMap_T`

Define the DTD (days to departure) frat5 coef map.

Definition at line 96 of file [stdair_inventory_types.hpp](#).

33.5.2.226 `typedef std::map<FloatDuration_T, float> stdair::DTDProbMap_T`

Define the DTD (days to departure) probability map.

Definition at line 99 of file [stdair_inventory_types.hpp](#).

33.5.2.227 `typedef std::vector<CensorshipFlag_T> stdair::CensorshipFlagList_T`

Define the list of censorship flags (une list per booking class, one censorship flag per DCP).

Definition at line 103 of file [stdair_inventory_types.hpp](#).

33.5.2.228 `typedef double stdair::BookingRatio_T`

Define the bookingRatio (for instance OnD bookings over whole class bookings).

Definition at line 107 of file [stdair_inventory_types.hpp](#).

33.5.2.229 `typedef double stdair::Yield_T`

Define the yield of a virtual class.

Definition at line 110 of file [stdair_inventory_types.hpp](#).

33.5.2.230 `typedef unsigned int stdair::YieldLevel_T`

Define the yield level (yield as an integer).

Definition at line 113 of file [stdair_inventory_types.hpp](#).

33.5.2.231 `typedef std::map<YieldLevel_T, MeanStdDevPair_T>
stdair::YieldLevelDemandMap_T`

Define the <YieldLevel, demand> demand map.

Definition at line 116 of file [stdair_inventory_types.hpp](#).

33.5.2.232 `typedef std::pair<Yield_T, MeanStdDevPair_T>
stdair::YieldDemandPair_T`

Define the <Yield, demand> pair type.

Definition at line 119 of file [stdair_inventory_types.hpp](#).

33.5.2.233 `typedef double stdair::BidPrice_T`

Define the Bid-Price.

Definition at line 122 of file [stdair_inventory_types.hpp](#).

33.5.2.234 `typedef std::vector<BidPrice_T> stdair::BidPriceVector_T`

Define a Bid-Price Vector.

Definition at line 125 of file [stdair_inventory_types.hpp](#).

33.5.2.235 `typedef unsigned int stdair::SeatIndex_T`

Define the current index of a Bid-Price Vector (for a given [LegCabin](#)).

Definition at line 128 of file [stdair_inventory_types.hpp](#).

33.5.2.236 `typedef std::string stdair::ControlMode_T`

Mode of inventory control.

Definition at line 131 of file [stdair_inventory_types.hpp](#).

33.5.2.237 `typedef double stdair::OverbookingRate_T`

Define the rate of overbooking

Definition at line 134 of file [stdair_inventory_types.hpp](#).

33.5.2.238 `typedef double stdair::ProtectionLevel_T`

Define the Protection Level.

It is a double, as it allows for overbooking.

Definition at line 142 of file [stdair_inventory_types.hpp](#).

33.5.2.239 `typedef std::vector<double> stdair::EmsrValueList_T`

Define the list of EMSR values for the EMSR algorithm.

Definition at line 145 of file [stdair_inventory_types.hpp](#).

33.5.2.240 `typedef std::vector<double> stdair::BookingLimitVector_T`

Define the vector of booking limits.

It is a vector of double.

Definition at line 149 of file [stdair_inventory_types.hpp](#).

33.5.2.241 `typedef std::vector<double> stdair::ProtectionLevelVector_T`

Define the vector of protection levels.

It is a vector of double.

Definition at line 153 of file [stdair_inventory_types.hpp](#).

33.5.2.242 `typedef boost::multi_array<double, 2> stdair::SnapshotBlock_T`

Define a snapshot block.

Definition at line 156 of file [stdair_inventory_types.hpp](#).

33.5.2.243 `typedef SnapshotBlock_T::index_range stdair::SnapshotBlockRange_T`

Define a range for array view.

Definition at line 159 of file [stdair_inventory_types.hpp](#).

33.5.2.244 `typedef SnapshotBlock_T::array_view<1>::type
stdair::SegmentCabinDTDSnapshotView_T`

Define a view for a given DTD.

Definition at line 162 of file [stdair_inventory_types.hpp](#).

33.5.2.245 `typedef SnapshotBlock_T::array_view<2>::type
stdair::SegmentCabinDTDRangeSnapshotView_T`

Define a view for a given range of DTD.

Definition at line 165 of file [stdair_inventory_types.hpp](#).

33.5.2.246 `typedef SnapshotBlock_T::const_array_view<1>::type
stdair::ConstSegmentCabinDTDSnapshotView_T`

Define a const view for a given DTD.

Definition at line 168 of file [stdair_inventory_types.hpp](#).

33.5.2.247 `typedef SnapshotBlock_T::const_array_view<2>::type
stdair::ConstSegmentCabinDTDRangeSnapshotView_T`

Define a const view for a given range of DTD.

Definition at line 171 of file [stdair_inventory_types.hpp](#).

33.5.2.248 `typedef unsigned short stdair::BlockNumber_T`

Define the snapshot block number.

Definition at line 174 of file [stdair_inventory_types.hpp](#).

33.5.2.249 `typedef unsigned short stdair::BlockIndex_T`

Define the index type within a snapshot block.

Definition at line 177 of file [stdair_inventory_types.hpp](#).

33.5.2.250 typedef unsigned int stdair::ReplicationNumber_T

Define the replication number.

Definition at line 24 of file [stdair_maths_types.hpp](#).

33.5.2.251 typedef unsigned long int stdair::ExponentialSeed_T

Define the seed type of an Exponential function.

Definition at line 29 of file [stdair_maths_types.hpp](#).

33.5.2.252 typedef unsigned long int stdair::UniformSeed_T

Define the seed type of an Uniform function.

Definition at line 34 of file [stdair_maths_types.hpp](#).

33.5.2.253 typedef unsigned long int stdair::RandomSeed_T

Seed for the random generation, so that it can be reproducible.

Definition at line 39 of file [stdair_maths_types.hpp](#).

33.5.2.254 typedef boost::minstd_rand stdair::BaseGenerator_T

Random number generator.

Definition at line 44 of file [stdair_maths_types.hpp](#).

33.5.2.255 typedef boost::uniform_real stdair::UniformDistribution_T

Uniform distribution of real numbers (by default, double).

Definition at line 49 of file [stdair_maths_types.hpp](#).

33.5.2.256 typedef boost::variate_generator<BaseGenerator_T&, UniformDistribution_T> stdair::UniformGenerator_T

Uniform random generator.

Definition at line 55 of file [stdair_maths_types.hpp](#).

33.5.2.257 typedef boost::normal_distribution stdair::NormalDistribution_T

Normal distribution of real numbers (by default, double).

Definition at line 60 of file [stdair_maths_types.hpp](#).

33.5.2.258 typedef boost::variate_generator<BaseGenerator_T&, NormalDistribution_T> stdair::NormalGenerator_T

Normal random generator.

Definition at line 66 of file [stdair_maths_types.hpp](#).

33.5.2.259 `typedef boost::exponential_distribution stdair::ExponentialDistribution_T`

Type definiton for the exponential distribution (characteristics).

Definition at line 69 of file [stdair_maths_types.hpp](#).

33.5.2.260 `typedef boost::variate_generator<BaseGenerator_T&, ExponentialDistribution_T> stdair::ExponentialGenerator_T`

Type definition for the exponential distribution random generator.

Definition at line 74 of file [stdair_maths_types.hpp](#).

33.5.2.261 `typedef double stdair::MeanValue_T`

Define a mean value (e.g., 20.2).

Definition at line 79 of file [stdair_maths_types.hpp](#).

33.5.2.262 `typedef double stdair::StdDevValue_T`

Define a standard deviation value (e.g., 1.5).

Definition at line 84 of file [stdair_maths_types.hpp](#).

33.5.2.263 `typedef std::pair<MeanValue_T, StdDevValue_T> stdair::MeanStdDevPair_T`

Define a couple (mean, standart deviation) (e.g., (20.2,1.5)).

Definition at line 89 of file [stdair_maths_types.hpp](#).

33.5.2.264 `typedef float stdair::Probability_T`

Probability.

Definition at line 94 of file [stdair_maths_types.hpp](#).

33.5.2.265 `typedef std::string stdair::ForecasterMode_T`

Mode of the forecaster.

Definition at line 17 of file [stdair_rm_types.hpp](#).

33.5.2.266 `typedef short stdair::HistoricalDataLimit_T`

Limit of similar flight-dates used in the forecaster.

Definition at line 20 of file [stdair_rm_types.hpp](#).

33.5.2.267 `typedef std::string stdair::OptimizerMode_T`

Mode of the forecaster.

Definition at line 23 of file [stdair_rm_types.hpp](#).

33.5.2.268 `typedef NbOfBookings_T stdair::PolicyDemand_T`

Define the demand for a policy.

Definition at line 26 of file [stdair_rm_types.hpp](#).

33.5.2.269 `typedef std::vector<double> stdair::GeneratedDemandVector_T`

Define the vector of generated demand (for MC integration use).

It is a vector of double.

Definition at line 30 of file [stdair_rm_types.hpp](#).

33.5.2.270 `typedef std::vector<GeneratedDemandVector_T>
stdair::GeneratedDemandVectorHolder_T`

Define the holder of the generated demand vectors.

Definition at line 33 of file [stdair_rm_types.hpp](#).

33.5.2.271 `typedef double stdair::SellupProbability_T`

Define the sellup probability.

Definition at line 36 of file [stdair_rm_types.hpp](#).

33.5.2.272 `typedef std::vector<SellupProbability_T>
stdair::SellupProbabilityVector_T`

Define the sellup probability vector.

Definition at line 39 of file [stdair_rm_types.hpp](#).

33.5.2.273 `typedef std::vector<double> stdair::SellupFactorHolder_T`

Define the holder of sellup factors (used for computing Q-eq bookings)

Definition at line 42 of file [stdair_rm_types.hpp](#).

33.5.2.274 `typedef boost::shared_ptr<STDAIR_Service>
stdair::STDAIR_ServicePtr_T`

Pointer on the STDAIR Service handler.

Definition at line 13 of file [stdair_service_types.hpp](#).

33.5.3 Function Documentation**33.5.3.1** `const std::string stdair::DEFAULT_BOM_ROOT_KEY (" -- ROOT -- ")`

Default value for the BOM tree root key (" -- ROOT -- ").

33.5.3.2 `const double stdair::DEFAULT_EPSILON_VALUE (0.0001)`

Default very small value.

33.5.3.3 `const unsigned int stdair::DEFAULT_FLIGHT_SPEED (900)`

Default flight speed (number of kilometers per hour).

33.5.3.4 `const NbOfFlightDates_T stdair::DEFAULT_NB_OF_FLIGHTDATES (0.0)`

Default number of generated flight dates.

33.5.3.5 `const Duration_T stdair::NULL_BOOST_TIME_DURATION (-1, -1, -1)`

Null time duration (in boost::time_duration unit).

33.5.3.6 `const unsigned int stdair::DEFAULT_NB_OF_DAYS_IN_A_YEAR (365)`

Default number of days in a year.

33.5.3.7 `const unsigned int stdair::DEFAULT_NUMBER_OF_SUBDIVISIONS (1000)`

Higher value per thousand

33.5.3.8 `const DayDuration_T stdair::DEFAULT_DAY_DURATION (0)`

Default number of duration days.

33.5.3.9 `const DatePeriod_T stdair::BOOST_DEFAULT_DATE_PERIOD (Date_T(2007, 1, 1), Date_T(2007, 1, 1))`

Default date period (0-length, i.e., it lasts one day).

33.5.3.10 `const DOW_String_T stdair::DEFAULT_DOW_STRING ("0000000")`

Default DOW String (e.g., "0000000").

33.5.3.11 `const DateOffset_T stdair::DEFAULT_DATE_OFFSET (0)`

Default Date Offset (e.g., 0).

33.5.3.12 `const Date_T stdair::DEFAULT_DATE (2010, boost::gregorian::Jan, 1)`

Default date for the General.

33.5.3.13 `const DateTime_T stdair::DEFAULT_DATETIME (DEFAULT_DATE, NULL_BOOST_TIME_DURATION)`

Default date-time.

33.5.3.14 `const Duration_T stdair::DEFAULT_EPSILON_DURATION (0, 0, 0, 1)`

Default epsilon duration (1 nanosecond).

33.5.3.15 `const Count_T stdair::SECONDS_IN_ONE_DAY (86400)`

Number of seconds in one day.

33.5.3.16 `const Count_T stdair::MILLISECONDS_IN_ONE_SECOND (1000)`

Number of milliseconds in one second

33.5.3.17 `const RandomSeed_T stdair::DEFAULT_RANDOM_SEED (120765987)`

Default random seed.

33.5.3.18 `const AirportCode_T stdair::AIRPORT_LHR ("LHR")`

Default origin airport (e.g., "LHR").

33.5.3.19 `const AirportCode_T stdair::AIRPORT_SYD ("SYD")`

Default destination airport (e.g., "SYD").

33.5.3.20 `const CityCode_T stdair::POS_LHR ("LHR")`

London city code (e.g., "LHR").

33.5.3.21 `const Date_T stdair::DATE_20110115 (2011 , boost::gregorian::Jan , 15)`

Date.

33.5.3.22 `const Date_T stdair::DATE_20111231 (2011 , boost::gregorian::Dec , 31)`

33.5.3.23 `const DayDuration_T stdair::NO_ADVANCE_PURCHASE (0)`

Advance purchase 0 day.

33.5.3.24 `const SaturdayStay_T stdair::SATURDAY_STAY (true)`

Default saturdayStay value (true).

33.5.3.25 `const SaturdayStay_T stdair::NO_SATURDAY_STAY (false)`

Default saturdayStay value (false).

33.5.3.26 `const ChangeFees_T stdair::CHANGE_FEES (true)`

Default change fees value (true).

33.5.3.27 `const ChangeFees_T stdair::NO_CHANGE_FEES (false)`

Default change fees value (false).

33.5.3.28 `const NonRefundable_T stdair::NON_REFUNDABLE (true)`

Default non refundable value (true).

33.5.3.29 `const NonRefundable_T stdair::No_NON_REFUNDABLE (false)`

Default refundable value (false).

33.5.3.30 `const SaturdayStay_T stdair::DEFAULT_BOM_TREE_SATURDAY_STAY (true)`

Default saturdayStay value (true).

33.5.3.31 `const ChangeFees_T stdair::DEFAULT_BOM_TREE_CHANGE_FEES (true)`

Default change fees value (true).

33.5.3.32 `const NonRefundable_T stdair::DEFAULT_BOM_TREE_NON_REFUNDABLE (true)`

Default non refundable value (true).

33.5.3.33 `const DayDuration_T stdair::NO_STAY_DURATION (0)`

Stay duration 0 day.

33.5.3.34 `const AirlineCode_T stdair::AIRLINE_CODE_BA ("BA")`

Airline code "BA".

33.5.3.35 `const CabinCode_T stdair::CABIN_Y ("Y")`

Cabin 'Y'.

33.5.3.36 `const ClassCode_T stdair::CLASS_CODE_Y ("Y")`

Class code 'Y'.

33.5.3.37 `const ClassCode_T stdair::CLASS_CODE_Q ("Q")`

Class code 'Q'.

33.5.3.38 `const AirportCode_T stdair::AIRPORT_SIN ("SIN")`

Singapour airport (e.g., "SIN").

33.5.3.39 `const AirportCode_T stdair::AIRPORT_BKK ("BKK")`

Bangkok airport (e.g., "BKK").

33.5.3.40 `const CityCode_T stdair::POS_SIN ("SIN")`

Singapour city code (e.g., "SIN").

33.5.3.41 `const CabinCode_T stdair::CABIN_ECO ("Eco")`

Economic cabin (e.g., "Eco").

33.5.3.42 `const FrequentFlyer_T stdair::FREQUENT_FLYER_MEMBER ("M")`

Frequent flyer tier (e.g., "M" meaning member).

33.5.3.43 `const ClassCode_T stdair::DEFAULT_FAMILY_CODE ("0")`

Default family code value ("X").

33.5.3.44 `const NbOfAirlines_T stdair::DEFAULT_NBOFAIRLINES (0)`

Default number of airlines.

33.5.3.45 `const FlightPathCode_T stdair::DEFAULT_FLIGHTPATH_CODE ("")`

Default flight-path code value ("").

33.5.3.46 `const Distance_T stdair::DEFAULT_DISTANCE_VALUE (0)`

Default distance value (kilometers).

33.5.3.47 `const ClassCode_T stdair::DEFAULT_CLOSED_CLASS_CODE ("CC")`

Default closed class code.

33.5.3.48 `const NbOfBookings_T stdair::DEFAULT_CLASS_NB_OF_BOOKINGS (0)`

Default number of bookings (with counted cancellation) for [BookingClass](#).

33.5.3.49 `const NbOfBookings_T stdair::DEFAULT_CLASS_TOTAL_NB_OF_BOOKINGS (0)`

Default number of booking (without cancellation) demands for [BookingClass](#).

33.5.3.50 `const NbOfBookings_T stdair::DEFAULT_CLASS_UNCONSTRAINED_DEMAND (0)`

Default unconstrained demand for [BookingClass](#).

33.5.3.51 `const NbOfBookings_T stdair::DEFAULT_CLASS_REMAINING_DEMAND_MEAN (0)`

Default remaining future demand mean for [BookingClass](#).

33.5.3.52 `const NbOfBookings_T stdair::DEFAULT_CLASS_REMAINING_DEMAND_STANDARD_DEVIATION (0)`

Default remaining futre demand standard deviation for [BookingClass](#).

33.5.3.53 `const NbOfCancellations_T stdair::DEFAULT_CLASS_NB_OF_CANCELLATIONS (0)`

Default number of cancellations for [BookingClass](#).

33.5.3.54 `const NbOfNoShows_T stdair::DEFAULT_CLASS_NB_OF_NOSHOWS (0)`

Default number of no-shows for [BookingClass](#).

33.5.3.55 `const CabinCapacity_T stdair::DEFAULT_CABIN_CAPACITY (100. 0)`

Default cabin capacity for Leg cabins.

33.5.3.56 `const CommittedSpace_T stdair::DEFAULT_COMMITTED_SPACE (0. 0)`

Default committed space value for Leg cabins.

33.5.3.57 `const BlockSpace_T stdair::DEFAULT_BLOCK_SPACE (0. 0)`

Default committed space value for Leg cabins.

33.5.3.58 `const Availability_T stdair::DEFAULT_NULL_AVAILABILITY (0. 0)`

Default null availability (0.0).

33.5.3.59 `const Availability_T stdair::DEFAULT_AVAILABILITY (9. 0)`

Default availability (9.0).

33.5.3.60 `const Availability_T stdair::MAXIMAL_AVAILABILITY (9999. 0)`

Maximal offered capacity in a cabin.

33.5.3.61 `const CensorshipFlag_T stdair::DEFAULT_CLASS_CENSORSHIPFLAG (false)`

Default boolean for censorship flag given the status of availability for [BookingClass](#).

33.5.3.62 `const BookingLimit_T stdair::DEFAULT_CLASS_BOOKING_LIMIT (9999. 0)`

Default booking limit value for [BookingClass](#).

33.5.3.63 `const AuthorizationLevel_T stdair::DEFAULT_CLASS_AUTHORIZATION_LEVEL (9999. 0)`

Default authorization level for [BookingClass](#).

33.5.3.64 `const AuthorizationLevel_T stdair::DEFAULT_ -
CLASS_MAX_AUTHORIZATION_LEVEL (9999. 0
)`

Default MAX value of authorization level for [BookingClass](#).

33.5.3.65 `const AuthorizationLevel_T stdair::DEFAULT_CLASS_MIN_ -
AUTHORIZATION_LEVEL (0. 0)`

Default MIN value of authorization level for [BookingClass](#).

33.5.3.66 `const OverbookingRate_T stdair::DEFAULT_CLASS_OVERBOOKING_ -
RATE (0. 0)`

Default over-booking rate for [BookingClass](#).

33.5.3.67 `const BookingRatio_T stdair::DEFAULT_OND_BOOKING_RATE (0. 0)`

Default booking rate for OnD bookings over overall class bookings.

33.5.3.68 `const Fare_T stdair::DEFAULT_FARE_VALUE (0. 0)`

Default Fare value.

33.5.3.69 `const Yield_T stdair::DEFAULT_CLASS_YIELD_VALUE (0. 0)`

Default yield value for a virtual class.

33.5.3.70 `const Revenue_T stdair::DEFAULT_REVENUE_VALUE (0. 0)`

Default Revenue value.

33.5.3.71 `const Percentage_T stdair::DEFAULT_LOAD_FACTOR_VALUE (100. 0)`

Default load factor value (100%).

33.5.3.72 `const Yield_T stdair::DEFAULT_YIELD_VALUE (0. 0)`

Default yield value.

33.5.3.73 `const Yield_T stdair::DEFAULT_YIELD_MAX_VALUE (std::numeric_limits<
double >:: max())`

Default yield max value.

33.5.3.74 `const NbOfBookings_T stdair::DEFAULT_YIELD_NB_OF_BOOKINGS (0. 0)`

Default number of bookings for YieldRangeStruct_T.

33.5.3.75 `const Identity_T stdair::DEFAULT_BOOKING_NUMBER (0)`

Default booking number.

33.5.3.76 `const NbOfCancellations_T stdair::DEFAULT_YIELD_NB_OF_CANCELLATIONS (0. 0)`

Default cancellation number for YieldRangeStruct_T.

33.5.3.77 `const NbOfNoShows_T stdair::DEFAULT_YIELD_NB_OF_NOSHOWS (0. 0)`

Default no-shows number for YieldRangeStruct_T.

33.5.3.78 `const Availability_T stdair::DEFAULT_YIELD_AVAILABILITY (0. 0)`

Default availability for YieldRangeStruct_T.

33.5.3.79 `const CensorshipFlag_T stdair::DEFAULT_YIELD_CENSORSHIPFLAG (false)`

Default boolean for booking limit availability for YieldRangeStruct_T.

33.5.3.80 `const BookingLimit_T stdair::DEFAULT_YIELD_BOOKING_LIMIT (0. 0)`

Default booking limit value for YieldRangeStruct_T.

33.5.3.81 `const OverbookingRate_T stdair::DEFAULT_YIELD_OVERBOOKING_RATE (0. 0)`

Default over-booking rate for YieldRangeStruct_T.

33.5.3.82 `const Fare_T stdair::DEFAULT_OND_FARE_VALUE (0. 0)`

Default value of Fare.

33.5.3.83 `const EventQueueID_T stdair::DEFAULT_EVENT_QUEUE_ID ("EQ01")`

Default ID for the event queue.

33.5.3.84 `const Count_T stdair::DEFAULT_PROGRESS_STATUS (0)`

Default progress status.

33.5.3.85 `const Date_T stdair::DEFAULT_EVENT_OLDEST_DATE (2008 , boost::gregorian::Jan , 1)`

Default reference (oldest) date for the events. No event can occur before that date.

33.5.3.86 `const DateTime_T stdair::DEFAULT_EVENT_OLDEST_DATETIME (DEFAULT_EVENT_OLDEST_DATE , NULL_BOOST_TIME_DURATION)`

Default reference (oldest) date-time for the events. No event can occur before that date-time.

33.5.3.87 `const PartySize_T stdair::DEFAULT_PARTY_SIZE (1)`

Default party size in a request.

33.5.3.88 `const DayDuration_T stdair::DEFAULT_STAY_DURATION (7)`

Default duration for a stay.

33.5.3.89 `const WTP_T stdair::DEFAULT_WTP (1000. 0)`

Default Willingness-to-Pay (WTP, as expressed as a monetary unit).

33.5.3.90 `const Date_T stdair::DEFAULT_PREFERRED_DEPARTURE_DATE (
DEFAULT_DEPARTURE_DATE)`

Default departure date.

33.5.3.91 `const Duration_T stdair::DEFAULT_PREFERRED_DEPARTURE_TIME (8 ,
0 , 0)`

Default preferred departure time (08:00).

33.5.3.92 `const DateOffset_T stdair::DEFAULT_ADVANCE_PURCHASE (22)`

Default advance purchase.

33.5.3.93 `const Date_T stdair::DEFAULT_REQUEST_DATE (
DEFAULT_PREFERRED_DEPARTURE_DATE- DEFAULT_ADVANCE_PURCHASE)`

Default request date.

33.5.3.94 `const Duration_T stdair::DEFAULT_REQUEST_TIME (8 , 0 , 0)`

Default preferred departure time (08:00).

33.5.3.95 `const DateTime_T stdair::DEFAULT_REQUEST_DATE_TIME (
DEFAULT_REQUEST_DATE , DEFAULT_REQUEST_TIME)`

Default request date-time.

33.5.3.96 `const CabinCode_T stdair::DEFAULT_PREFERRED_CABIN ("M")`

Default preferred cabin.

33.5.3.97 `const CityCode_T stdair::DEFAULT_POS ("ALL")`

Default point-of-sale.

33.5.3.98 `const ChannelLabel_T stdair::DEFAULT_CHANNEL ("DC")`

Default channel (e.g., "DC" meaning Different Channels).

33.5.3.99 `const ChannelLabel_T stdair::CHANNEL_DN ("DN")`

DN channel (e.g., direct on-line).

33.5.3.100 **const ChannelLabel_T stdair::CHANNEL_IN ("IN")**

IN channel (e.g., indirect on-line).

33.5.3.101 **const TripType_T stdair::TRIP_TYPE_ONE_WAY ("OW")**

Trip type one-way (e.g., "OW").

33.5.3.102 **const TripType_T stdair::TRIP_TYPE_ROUND_TRIP ("RT")**

Trip type round-trip (e.g., "RT").

33.5.3.103 **const TripType_T stdair::TRIP_TYPE_INBOUND ("RI")**

Trip type inbound (e.g., "RI").

33.5.3.104 **const TripType_T stdair::TRIP_TYPE_OUTBOUND ("RO")**

Trip type outbound (e.g., "RO").

33.5.3.105 **const FrequentFlyer_T stdair::DEFAULT_FF_TIER ("N")**

Default frequent flyer tier (non member).

33.5.3.106 **const PriceValue_T stdair::DEFAULT_VALUE_OF_TIME (100.0)**

Default value of time (expressed as a monetary unit per hour).

33.5.3.107 **const Duration_T stdair::DEFAULT_MINIMAL_CONNECTION_TIME (0 , 30 , 0)**

Default Minimal connection time.

33.5.3.108 **const Duration_T stdair::DEFAULT_MAXIMAL_CONNECTION_TIME (24 , 0 , 0)**

Default maximal connection time.

33.5.3.109 **const MatchingIndicator_T stdair::DEFAULT_MATCHING_INDICATOR (0.0)**

Default Matching Indicator value.

33.5.3.110 **const PriceCurrency_T stdair::DEFAULT_CURRENCY ("EUR")**

Default currency (euro).

33.5.3.111 **const AvailabilityStatus_T stdair::DEFAULT_AVAILABILITY_STATUS (false)**

Default availability status for a travel solution.

33.5.3.112 `const AirlineCode_T stdair::DEFAULT_AIRLINE_CODE ("XX")`

Default airline code value ("XX").

33.5.3.113 `const AirlineCode_T stdair::DEFAULT_NULL_AIRLINE_CODE ("")`

Default airline code value ("").

33.5.3.114 `const FlightNumber_T stdair::DEFAULT_FLIGHT_NUMBER (9999)`

Default flight number (9999).

33.5.3.115 `const GuillotineNumber_T stdair::DEFAULT_GUILLOTINE_NUMBER (9999)`

Default guillotine number (9999).

33.5.3.116 `const Date_T stdair::DEFAULT_DEPARTURE_DATE (1900 , boost::gregorian::Jan , 1)`

Default flight departure date (01/01/1900).

33.5.3.117 `const AirportCode_T stdair::DEFAULT_AIRPORT_CODE ("XXX")`

Default airport code value ("XXX").

33.5.3.118 `const AirportCode_T stdair::DEFAULT_NULL_AIRPORT_CODE ("")`

Default airport code value ("").

33.5.3.119 `const AirportCode_T stdair::DEFAULT_ORIGIN ("XXX")`

Default Origin.

33.5.3.120 `const AirportCode_T stdair::DEFAULT_DESTINATION ("XXX")`

Default destination.

33.5.3.121 `const CabinCode_T stdair::DEFAULT_CABIN_CODE ("X")`

Default cabin code.

33.5.3.122 `const FamilyCode_T stdair::DEFAULT_FARE_FAMILY_CODE ("EcoSaver")`

Default fare family Code.

33.5.3.123 `const FamilyCode_T stdair::DEFAULT_NULL_FARE_FAMILY_CODE ("NoFF")`

Default null fare family Code ("NoFF").

33.5.3.124 `const ClassCode_T stdair::DEFAULT_CLASS_CODE ("X")`

Default class code value ("X").

33.5.3.125 `const ClassCode_T stdair::DEFAULT_NULL_CLASS_CODE ("")`

Default null class code value ("").

33.5.3.126 `const BidPrice_T stdair::DEFAULT_BID_PRICE (0.0)`

Default Bid-Price.

33.5.3.127 `const unsigned short stdair::MAXIMAL_NUMBER_OF_LEGS_IN_FLIGHT (7)`

Maximal number of legs linked to a single flight-date.

Note that the number of derived segments is $n*(n+1)/2$ if n is the number of legs.

33.5.3.128 `const unsigned short stdair::MAXIMAL_NUMBER_OF_SEGMENTS_IN_OND (3)`

Maximal number of segments linked to a single O&D (Origin & Destination).

33.5.3.129 `const SeatIndex_T stdair::DEFAULT_SEAT_INDEX (1)`

Default seat index (for a bucket and/or Bid-Price Vector slot).

33.5.3.130 `const std::string stdair::DEFAULT_FARE_FAMILY_VALUE_TYPE ("FF")`

Default value type (within a guillotine block) for fare family.

33.5.3.131 `const std::string stdair::DEFAULT_SEGMENT_CABIN_VALUE_TYPE ("SC")`

Default value type (within a guillotine block) for segment-cabin.

33.5.3.132 `const std::string stdair::DEFAULT_KEY_FLD_DELIMITER (";")`

Default delimiter for string display (e.g delimiter for inventory key and flight-date key).

33.5.3.133 `const std::string stdair::DEFAULT_KEY_SUB_FLD_DELIMITER (", ")`

Default sub delimiter for string display (e.g delimiter for flight number and departure date of a flight-date key).

33.5.3.134 `const boost::char_separator<char> stdair::DEFAULT_KEY_TOKEN_DELIMITER (";", " ")`

Default token for decoding a full string display.

33.5.3.135 `template<int MIN, int MAX> date_time_element<MIN, MAX> stdair::operator* (const date_time_element< MIN, MAX > & o1, const date_time_element< MIN, MAX > & o2) [inline]`

Operator* overload.

Definition at line 47 of file [BasParserHelperTypes.hpp](#).

References [stdair::date_time_element< MIN, MAX >::_value](#).

33.5.3.136 `template<int MIN, int MAX> date_time_element<MIN, MAX> stdair::operator+ (const date_time_element< MIN, MAX > & o1, const date_time_element< MIN, MAX > & o2) [inline]`

Operator+ overload.

Definition at line 55 of file [BasParserHelperTypes.hpp](#).

References [stdair::date_time_element< MIN, MAX >::_value](#).

33.5.3.137 `template void stdair::AirlineClassListKey::serialize< ba::text_oarchive > (ba::text_oarchive & , unsigned int)`

33.5.3.138 `template void stdair::AirlineClassListKey::serialize< ba::text_iarchive > (ba::text_iarchive & , unsigned int)`

33.5.3.139 `void stdair::jsonLegDateExport (bpt::ptree & ioPropertyTree, const FlightDate & iFlightDate)`

Definition at line 67 of file [BomJSONExport.cpp](#).

References [stdair::LegDate::getBoardingDate\(\)](#), [stdair::LegDate::getBoardingPoint\(\)](#), [stdair::LegDate::getBoardingTime\(\)](#), [stdair::LegDate::getCapacity\(\)](#), [stdair::LegDate::getDateOffset\(\)](#), [stdair::LegDate::getDistance\(\)](#), [stdair::LegDate::getElapsedTime\(\)](#), [stdair::LegDate::getOffDate\(\)](#), [stdair::LegDate::getOffPoint\(\)](#), [stdair::LegDate::getOffTime\(\)](#), and [stdair::LegDate::getTimeOffset\(\)](#).

Referenced by [stdair::BomJSONExport::jsonExport\(\)](#).

33.5.3.140 `void stdair::jsonSegmentDateExport (bpt::ptree & ioPropertyTree, const FlightDate & iFlightDate)`

Definition at line 123 of file [BomJSONExport.cpp](#).

33.5.3.141 `void stdair::jsonLegCabinExport (bpt::ptree & ioPropertyTree, const FlightDate & iFlightDate)`

Definition at line 131 of file [BomJSONExport.cpp](#).

References [stdair::LegCabin::getCabinCode\(\)](#).

33.5.3.142 `void stdair::jsonSegmentCabinExport (bpt::ptree & ioPropertyTree, const FlightDate & iFlightDate)`

Definition at line 189 of file [BomJSONExport.cpp](#).

33.5.3.143 void stdair::jsonFareFamilyExport (bpt::ptree & *ioPropertyTree*, const FlightDate & *iFlightDate*)

Definition at line 197 of file [BomJSONExport.cpp](#).

References [stdair::FareFamily::getFamilyCode\(\)](#).

33.5.3.144 void stdair::jsonBucketExport (bpt::ptree & *ioPropertyTree*, const FlightDate & *iFlightDate*)

Definition at line 266 of file [BomJSONExport.cpp](#).

References [stdair::Bucket::getYieldRangeUpperValue\(\)](#).

33.5.3.145 void stdair::jsonBookingClassExport (bpt::ptree & *ioPropertyTree*, const BookingClass & *iBookingClass*, const std::string & *iLeadingString*)

Definition at line 321 of file [BomJSONExport.cpp](#).

References [stdair::BookingClass::getClassCode\(\)](#), and [stdair::BookingClass::getSubclassCode\(\)](#).

Referenced by [jsonBookingClassExport\(\)](#).

33.5.3.146 void stdair::jsonBookingClassExport (bpt::ptree & *ioPropertyTree*, const FlightDate & *iFlightDate*)

Definition at line 363 of file [BomJSONExport.cpp](#).

References [stdair::FareFamily::getFamilyCode\(\)](#), and [jsonBookingClassExport\(\)](#).

33.5.3.147 template void stdair::BomRootKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)

33.5.3.148 template void stdair::BomRootKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)

33.5.3.149 void stdair::intDisplay (std::ostream & *oStream*, const int & *ilnt*)

Definition at line 137 of file [BookingRequestStruct.cpp](#).

33.5.3.150 template void stdair::BucketKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)

33.5.3.151 template void stdair::BucketKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)

33.5.3.152 template void stdair::FareFamilyKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)

33.5.3.153 template void stdair::FareFamilyKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)

33.5.3.154 template void stdair::FlightDateKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)

33.5.3.155 `template void stdair::FlightDateKey::serialize< ba::text_iarchive > (ba::text_iarchive & , unsigned int)`

33.5.3.156 `template void stdair::GuillotineBlockKey::serialize< ba::text_oarchive > (ba::text_oarchive & , unsigned int)`

33.5.3.157 `template void stdair::GuillotineBlockKey::serialize< ba::text_iarchive > (ba::text_iarchive & , unsigned int)`

33.5.3.158 `template void stdair::InventoryKey::serialize< ba::text_oarchive > (ba::text_oarchive & , unsigned int)`

33.5.3.159 `template void stdair::InventoryKey::serialize< ba::text_iarchive > (ba::text_iarchive & , unsigned int)`

33.5.3.160 `template void stdair::OnDDateKey::serialize< ba::text_oarchive > (ba::text_oarchive & , unsigned int)`

33.5.3.161 `template void stdair::OnDDateKey::serialize< ba::text_iarchive > (ba::text_iarchive & , unsigned int)`

33.5.3.162 `const boost::char_separator<char> stdair::TokeniserDashSeparator ("-")`

Dash delimiter for the tokenisation process.

Referenced by [stdair::ParsedKey::getFlightDateKey\(\)](#).

33.5.3.163 `const boost::char_separator<char> stdair::TokeniserTimeSeparator (":")`

Time delimiter for the tokenisation process.

Referenced by [stdair::ParsedKey::getBoardingTime\(\)](#).

33.5.3.164 `template void stdair::SegmentCabinKey::serialize< ba::text_oarchive > (ba::text_oarchive & , unsigned int)`

33.5.3.165 `template void stdair::SegmentCabinKey::serialize< ba::text_iarchive > (ba::text_iarchive & , unsigned int)`

33.5.3.166 `template void stdair::SegmentDateKey::serialize< ba::text_oarchive > (ba::text_oarchive & , unsigned int)`

33.5.3.167 `template void stdair::SegmentDateKey::serialize< ba::text_iarchive > (ba::text_iarchive & , unsigned int)`

33.5.3.168 `template<class Archive , class BOM_OBJECT1 , class BOM_OBJECT2 > void stdair::serialiseHelper (BOM_OBJECT1 & ioObject1, Archive & ioArchive, const unsigned int iFileVersion)`

Definition at line 34 of file [CmdBomSerialiser.cpp](#).

References [stdair::BomHolder< BOM >::_bomList](#), [stdair::BomHolder< BOM >::_bomMap](#), and [stdair::FacBomManager::linkWithParent\(\)](#).

- 33.5.3.169 `template void stdair::BomRoot::serialize< ba::text_oarchive > (ba::text_oarchive & , unsigned int)`
- 33.5.3.170 `template void stdair::BomRoot::serialize< ba::text_iarchive > (ba::text_iarchive & , unsigned int)`
- 33.5.3.171 `template void stdair::Inventory::serialize< ba::text_oarchive > (ba::text_oarchive & , unsigned int)`
- 33.5.3.172 `template void stdair::Inventory::serialize< ba::text_iarchive > (ba::text_iarchive & , unsigned int)`
- 33.5.3.173 `template void stdair::FlightDate::serialize< ba::text_oarchive > (ba::text_oarchive & , unsigned int)`
- 33.5.3.174 `template void stdair::FlightDate::serialize< ba::text_iarchive > (ba::text_iarchive & , unsigned int)`
- 33.5.3.175 `template void stdair::SegmentDate::serialize< ba::text_oarchive > (ba::text_oarchive & , unsigned int)`
- 33.5.3.176 `template void stdair::SegmentDate::serialize< ba::text_iarchive > (ba::text_iarchive & , unsigned int)`
- 33.5.3.177 `template void stdair::SegmentCabin::serialize< ba::text_oarchive > (ba::text_oarchive & , unsigned int)`
- 33.5.3.178 `template void stdair::SegmentCabin::serialize< ba::text_iarchive > (ba::text_iarchive & , unsigned int)`

33.5.4 Variable Documentation

33.5.4.1 `const std::string stdair::DOW_STR`

Initial value:

```
{ "Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun" }
```

Day names (in English).

Representation of Dow-Of-the-Week

Definition at line 49 of file [BasConst.cpp](#).

Referenced by [stdair::DoWStruct::describe\(\)](#).

33.5.4.2 `const CensorshipFlagList_T stdair::DEFAULT_CLASS_CENSORSHIPFLAG_LIST`

Initial value:

```
std::vector<CensorshipFlag_T>()
```

Default list of censorship flag given the status of availability for [BookingClass](#).

Definition at line 220 of file [BasConst.cpp](#).

33.5.4.3 const Date_T stdair::DEFAULT_DICO_STUDIED_DATE

Default DICO studied date.

Definition at line 390 of file [BasConst.cpp](#).

33.5.4.4 const AirlineCodeList_T stdair::DEFAULT_AIRLINE_CODE_LIST

Default airline code list value (empty vector).

Definition at line 400 of file [BasConst.cpp](#).

33.5.4.5 const ClassList_StringList_T stdair::DEFAULT_CLASS_CODE_LIST

Default class code list value (empty vector).

Definition at line 439 of file [BasConst.cpp](#).

33.5.4.6 const BidPriceVector_T stdair::DEFAULT_BID_PRICE_VECTOR = std::vector<BidPrice_T>()

Default Bid-Price Vector.

Default Bid-Price Vector (empty vector).

Definition at line 445 of file [BasConst.cpp](#).

33.5.4.7 const int stdair::DEFAULT_MAX_DTD = 365

Default value for max day-to-departure (365).

Definition at line 466 of file [BasConst.cpp](#).

Referenced by [stdair::GuillotineBlock::initSnapshotBlocks\(\)](#).

33.5.4.8 const DCPList_T stdair::DEFAULT_DCP_LIST = DefaultDCPList::init()

Default data collection point list.

Definition at line 469 of file [BasConst.cpp](#).

33.5.4.9 const DTDFratMap_T stdair::DEFAULT_DTD_FRAT5COEF_MAP

Initial value:

```
DefaultDtdFratMap::init()
```

Default frat5 coef map for demand to come forecaster.

Default frat5 coef map.

Definition at line 487 of file [BasConst.cpp](#).

33.5.4.10 const DTDProbMap_T stdair::DEFAULT_DTD_PROB_MAP

Initial value:

```
DefaultDtdProbMap::init()
```

Default arrival pattern map.

Definition at line 504 of file [BasConst.cpp](#).

33.5.4.11 const OnDStringList_T stdair::DEFAULT_OND_STRING_LIST

Default list of full keys.

Definition at line 529 of file [BasConst.cpp](#).

33.5.4.12 const std::string stdair::DISPLAY_LEVEL_STRING_ARRAY

Array with the indentation spaces needed for all the BOM hierarchical levels.

Definition at line 535 of file [BasConst.cpp](#).

33.5.4.13 const std::string stdair::DEFAULT_KEY_FLD_DELIMITER

Default delimiter for string display (e.g delimiter for inventory key and flight-date key). Typically set to ','.

Referenced by [stdair::SegmentCabin::getFullerKey\(\)](#), [stdair::LegCabin::getFullerKey\(\)](#), and [stdair::ParsedKey::toString\(\)](#).

33.5.4.14 const std::string stdair::DEFAULT_KEY_SUB_FLD_DELIMITER

Default sub delimiter for string display (e.g delimiter for flight number and departure date of a flight-date key). Typically set to ','.

Referenced by [stdair::SegmentDateKey::toString\(\)](#), [stdair::PosChannelKey::toString\(\)](#), [stdair::ParsedKey::toString\(\)](#), [stdair::FlightDateKey::toString\(\)](#), [stdair::AirportPairKey::toString\(\)](#), and [stdair::AirlineClassListKey::toString\(\)](#).

33.5.4.15 const boost::char_separator<char> stdair::DEFAULT_KEY_TOKEN_DELIMITER

Default token for decoding a full string display.

Referenced by [stdair::BomKeyManager::extractKeys\(\)](#).

33.5.4.16 const Distance_T stdair::DEFAULT_DISTANCE_VALUE

Default distance value, in kilometers (0).

Default distance value (kilometers).

Definition at line 30 of file [BasConst_General.hpp](#).

33.5.4.17 const ClassCode_T stdair::DEFAULT_CLOSED_CLASS_CODE

Default closed class code ("CC").

33.5.4.18 const NbOfBookings_T stdair::DEFAULT_CLASS_NB_OF_BOOKINGS

Default number of bookings (with counted cancellation) for [BookingClass](#) (0).

Default number of bookings for [BookingClass](#).

Default number of bookings (0).

Definition at line 27 of file [BasConst_General.hpp](#).

33.5.4.19 const NbOfBookings_T stdair::DEFAULT_CLASS_TOTAL_NB_OF_BOOKINGS

Default number of bookings (without cancellation) for [BookingClass](#) (0).

33.5.4.20 const NbOfBookings_T stdair::DEFAULT_CLASS_UNCONSTRAINED_DEMAND

Default unconstrained demand for [BookingClass](#) (0).

33.5.4.21 const NbOfBookings_T stdair::DEFAULT_CLASS_REMAINING_DEMAND_MEAN

Default remaining future demand mean for [BookingClass](#) (0).

33.5.4.22 const NbOfBookings_T stdair::DEFAULT_CLASS_REMAINING_DEMAND_STANDARD_DEVIATION

Default remaining future demand standard deviation for [BookingClass](#) (0).

33.5.4.23 const NbOfCancellations_T stdair::DEFAULT_CLASS_NB_OF_CANCELLATIONS

Default number of cancellations for [BookingClass](#) (0).

33.5.4.24 const NbOfNoShows_T stdair::DEFAULT_CLASS_NB_OF_NOSHOWS

Default number of no-shows for [BookingClass](#) (0).

33.5.4.25 const CabinCapacity_T stdair::DEFAULT_CABIN_CAPACITY

Default cabin capacity for Leg cabins (0.0).

Default cabin capacity for Leg cabins.

Definition at line 21 of file [BasConst_General.hpp](#).

33.5.4.26 const CommittedSpace_T stdair::DEFAULT_COMMITTED_SPACE

Default committed space value for Leg cabins (0.0).

33.5.4.27 const BlockSpace_T stdair::DEFAULT_BLOCK_SPACE

Default committed space value for Leg cabins (0.0).

33.5.4.28 const Availability_T stdair::DEFAULT_NULL_AVAILABILITY

Default null availability (0.0).

33.5.4.29 const Availability_T stdair::DEFAULT_AVAILABILITY

Default availability (9.0).

33.5.4.30 const CensorshipFlag_T stdair::DEFAULT_CLASS_CENSORSHIPFLAG

Default boolean for censorship flag given the status of availability for [BookingClass](#).

33.5.4.31 const BookingLimit_T stdair::DEFAULT_CLASS_BOOKING_LIMIT

Default booking limit value for [BookingClass](#).

33.5.4.32 const AuthorizationLevel_T stdair::DEFAULT_CLASS_AUTHORIZATION_LEVEL

Default authorization level for [BookingClass](#).

33.5.4.33 const AuthorizationLevel_T stdair::DEFAULT_CLASS_MAX_AUTHORIZATION_LEVEL

Default MAX value of authorization level for [BookingClass](#).

33.5.4.34 const AuthorizationLevel_T stdair::DEFAULT_CLASS_MIN_AUTHORIZATION_LEVEL

Default MIN value of authorization level for [BookingClass](#).

33.5.4.35 const OverbookingRate_T stdair::DEFAULT_CLASS_OVERBOOKING_RATE

Default over-booking rate for [BookingClass](#).

33.5.4.36 const Fare_T stdair::DEFAULT_FARE_VALUE

Default fare.

Default value of Fare.

Definition at line 36 of file [BasConst_General.hpp](#).

33.5.4.37 const Revenue_T stdair::DEFAULT_REVENUE_VALUE

Default revenue value for [BookingClass](#).

Default revenue value.

Definition at line 42 of file [BasConst_General.hpp](#).

33.5.4.38 const PriceCurrency_T stdair::DEFAULT_CURRENCY

Default currency (euro).

Definition at line 39 of file [BasConst_General.hpp](#).

33.5.4.39 const Percentage_T stdair::DEFAULT_LOAD_FACTOR_VALUE

Default load factor value (100%).

33.5.4.40 const DayDuration_T stdair::DEFAULT_DAY_DURATION

Default number of duration days (0).

Default Duration in days (e.g., 0).

Definition at line 26 of file [BasConst_Period_BOM.hpp](#).

33.5.4.41 const double stdair::DEFAULT_EPSILON_VALUE

Default epsilon value between customer requirements and a fare rule.

Default epsilon value (1e-4).

Definition at line 18 of file [BasConst_General.hpp](#).

33.5.4.42 const AirportCode_T stdair::AIRPORT_LHR

London Heathrow airport (e.g., "LHR").

33.5.4.43 const AirportCode_T stdair::AIRPORT_SYD

Sydney airport (e.g., "SYD").

33.5.4.44 const CityCode_T stdair::POS_LHR

London city code (e.g., "LHR").

33.5.4.45 const DayDuration_T stdair::NO_ADVANCE_PURCHASE

Advance purchase 0 day.

33.5.4.46 const SaturdayStay_T stdair::SATURDAY_STAY

Default saturdayStay value (true).

33.5.4.47 const SaturdayStay_T stdair::NO_SATURDAY_STAY

Default saturdayStay value (false).

33.5.4.48 const ChangeFees_T stdair::CHANGE_FEES

Default change fees value (true).

33.5.4.49 const ChangeFees_T stdair::NO_CHANGE_FEES

Default change fees value (false).

33.5.4.50 const NonRefundable_T stdair::NON_REFUNDABLE

Default non refundable value (true).

33.5.4.51 const NonRefundable_T stdair::NO_NON_REFUNDABLE

Default refundable value (false).

33.5.4.52 const DayDuration_T stdair::NO_STAY_DURATION

Stay duration 0 day.

33.5.4.53 const CabinCode_T stdair::CABIN_Y

Cabin 'Y'.

33.5.4.54 const AirlineCode_T stdair::AIRLINE_CODE_BA

Airline code "BA".

33.5.4.55 const ClassCode_T stdair::CLASS_CODE_Y

Class code 'Y'.

33.5.4.56 const ClassCode_T stdair::CLASS_CODE_Q

Class code 'Q'.

33.5.4.57 const AirportCode_T stdair::AIRPORT_SIN

Singapour airport (e.g., "SIN").

33.5.4.58 const AirportCode_T stdair::AIRPORT_BKK

Bangkok airport (e.g., "BKK").

33.5.4.59 const CityCode_T stdair::POS_SIN

Singapour city code (e.g., "SIN").

33.5.4.60 const CabinCode_T stdair::CABIN_ECO

Economic cabin (e.g., "Eco").

33.5.4.61 const FrequentFlyer_T stdair::FREQUENT_FLYER_MEMBER

Frequent flyer tier (e.g., "M" meaning member).

33.5.4.62 const EventQueueID_T stdair::DEFAULT_EVENT_QUEUE_ID

Default ID for the event queue.

33.5.4.63 const Count_T stdair::DEFAULT_PROGRESS_STATUS

Default progress status.

Referenced by [stdair::ProgressStatus::reset\(\)](#).

33.5.4.64 const Date_T stdair::DEFAULT_EVENT_OLDEST_DATE

Default reference (oldest) date for the events. No event can occur before that date.

33.5.4.65 const DateTime_T stdair::DEFAULT_EVENT_OLDEST_DATETIME

Default reference (oldest) date-time for the events. No event can occur before that date-time.

Referenced by [stdair::EventStruct::describe\(\)](#), and [stdair::EventStruct::EventStruct\(\)](#).

33.5.4.66 const std::string stdair::DEFAULT_BOM_ROOT_KEY

Default value for the BOM tree root key (" -- ROOT -- ").

33.5.4.67 const NbOfFlightDates_T stdair::DEFAULT_NB_OF_FLIGHTDATES

Default number of generated flight dates (0).

33.5.4.68 const unsigned int stdair::DEFAULT_FLIGHT_SPEED

Default flight speed (number of kilometers per hour).

33.5.4.69 const BookingRatio_T stdair::DEFAULT_OND_BOOKING_RATE

Default booking rate for OnD bookings over overall class bookings.

33.5.4.70 const Count_T stdair::SECONDS_IN_ONE_DAY

Number of seconds in one day (86400).

33.5.4.71 const Count_T stdair::MILLISECONDS_IN_ONE_SECOND

Number of milliseconds in one second (1000).

33.5.4.72 const Date_T stdair::DEFAULT_DATE

Default date for the General (1-Jan-2010).

33.5.4.73 const DateTime_T stdair::DEFAULT_DATETIME

Default date-time (1-Jan-2010).

33.5.4.74 const Duration_T stdair::DEFAULT_EPSILON_DURATION

Default epsilon duration (1 nanosecond).

33.5.4.75 const RandomSeed_T stdair::DEFAULT_RANDOM_SEED

Default random seed (120765987).

Referenced by [stdair::BookingClass::generateDemandSamples\(\)](#).

33.5.4.76 const Duration_T stdair::NULL_BOOST_TIME_DURATION

Null time duration (in boost::time_duration unit).

Definition at line 23 of file [BasConst_TravelSolution.hpp](#).

33.5.4.77 const Fare_T stdair::DEFAULT_CLASS_FARE_VALUE

Default value of Availability.

33.5.4.78 const NbOfAirlines_T stdair::DEFAULT_NBOFAIRLINES

Default number of airlines (0).

33.5.4.79 const unsigned int stdair::DEFAULT_NB_OF_DAYS_IN_A_YEAR

Default number of days in a year (365).

33.5.4.80 const ChannelLabel_T stdair::DEFAULT_CHANNEL

Default channel.

Default channel (e.g., direct on-line).

Definition at line 48 of file [BasConst_Request.hpp](#).

33.5.4.81 const unsigned int stdair::DEFAULT_NUMBER_OF_SUBDIVISIONS

Higher value per thousand

Referenced by [stdair::DictionaryManager::keyToValue\(\)](#), and [stdair::DictionaryManager::valueToKey\(\)](#).

33.5.4.82 const AirlineCode_T stdair::DEFAULT_AIRLINE_CODE

Default airline code value ("XX").

Referenced by [stdair::BomRetriever::retrieveDummyLegCabin\(\)](#), and [stdair::BomRetriever::retrieveDummySegmentCab](#)

33.5.4.83 const AirlineCode_T stdair::DEFAULT_NULL_AIRLINE_CODE

Default airline code value ("").

33.5.4.84 const FlightNumber_T stdair::DEFAULT_FLIGHT_NUMBER

Default flight number (9999).

Referenced by [stdair::BomRetriever::retrieveDummyLegCabin\(\)](#), and [stdair::BomRetriever::retrieveDummySegmentCab](#)

33.5.4.85 `const GuillotineNumber_T stdair::DEFAULT_GUILLOTINE_NUMBER`

Default guillotine number (9999).

33.5.4.86 `const Date_T stdair::DEFAULT_DEPARTURE_DATE`

Default flight departure date (01/01/1900).

Referenced by [stdair::BomRetriever::retrieveDummyLegCabin\(\)](#), and [stdair::BomRetriever::retrieveDummySegmentCab](#)

33.5.4.87 `const AirportCode_T stdair::DEFAULT_AIRPORT_CODE`

Default airport code value ("XXX").

33.5.4.88 `const AirportCode_T stdair::DEFAULT_NULL_AIRPORT_CODE`

Default airport code value ("").

33.5.4.89 `const AirportCode_T stdair::DEFAULT_ORIGIN`

Default Origin ("XXX").

Referenced by [stdair::BomRetriever::retrieveDummyLegCabin\(\)](#), and [stdair::BomRetriever::retrieveDummySegmentCab](#)

33.5.4.90 `const AirportCode_T stdair::DEFAULT_DESTINATION`

Default Destination ("XXX").

Referenced by [stdair::BomRetriever::retrieveDummySegmentCabin\(\)](#).

33.5.4.91 `const CabinCode_T stdair::DEFAULT_CABIN_CODE`

Default Cabin Code ("X").

Referenced by [stdair::BomRetriever::retrieveDummyLegCabin\(\)](#), and [stdair::BomRetriever::retrieveDummySegmentCab](#)

33.5.4.92 `const FamilyCode_T stdair::DEFAULT_FARE_FAMILY_CODE`

Default Fare Family Code ("EcoSaver").

33.5.4.93 `const FamilyCode_T stdair::DEFAULT_NULL_FARE_FAMILY_CODE`

Default null fare family Code ("NoFF").

33.5.4.94 `const ClassCode_T stdair::DEFAULT_CLASS_CODE`

Default class code value ("X").

33.5.4.95 `const ClassCode_T stdair::DEFAULT_NULL_CLASS_CODE`

Default null class code value ("").

33.5.4.96 const BidPrice_T stdair::DEFAULT_BID_PRICE

Default Bid-Price (0.0).

33.5.4.97 const unsigned short stdair::MAXIMAL_NUMBER_OF_LEGS_IN_FLIGHT

Maximal number of legs linked to a single flight-date (e.g., 7).

Note that the number of derived segments is $n*(n+1)/2$ if n is the number of legs.

33.5.4.98 const unsigned short stdair::MAXIMAL_NUMBER_OF_SEGMENTS_IN_OND

Maximal number of segments linked to a single O&D (Origin & Destination)(e.g., 3).

33.5.4.99 const Availability_T stdair::MAXIMAL_AVAILABILITY

Maximal offered capacity in a cabin.

33.5.4.100 const SeatIndex_T stdair::DEFAULT_SEAT_INDEX

Default seat index (for a bucket and/or Bid-Price Vector slot)(e.g., 1).

33.5.4.101 const std::string stdair::DEFAULT_FARE_FAMILY_VALUE_TYPE

Default value type (within a guillotine block) for fare family.

33.5.4.102 const std::string stdair::DEFAULT_SEGMENT_CABIN_VALUE_TYPE

Default value type (within a guillotine block) for segment-cabin.

33.5.4.103 const DatePeriod_T stdair::BOOST_DEFAULT_DATE_PERIOD

Default date period (0-length, i.e., it lasts one day).

33.5.4.104 const DOW_String_T stdair::DEFAULT_DOW_STRING

Default DOW String (e.g., "1111100").

Referenced by [stdair::DoWStruct::intersection\(\)](#), and [stdair::DoWStruct::shift\(\)](#).

33.5.4.105 const DateOffset_T stdair::DEFAULT_DATE_OFFSET

Default Date Offset (e.g., 0).

33.5.4.106 const PartySize_T stdair::DEFAULT_PARTY_SIZE

Default party size in a request (e.g., 1).

33.5.4.107 const DayDuration_T stdair::DEFAULT_STAY_DURATION

Default duration for a stay (e.g., 7 days).

33.5.4.108 const WTP_T stdair::DEFAULT_WTP

Default Willingness-to-Pay (WTP, as expressed as a monetary unit).

33.5.4.109 const CityCode_T stdair::DEFAULT_POS

Default Point-Of-Sale (POS, e.g., "WORLD").

33.5.4.110 const Date_T stdair::DEFAULT_PREFERRED_DEPARTURE_DATE

Default departure date (e.g., 01-Jan-2011).

33.5.4.111 const Duration_T stdair::DEFAULT_PREFERRED_DEPARTURE_TIME

Default preferred departure time (e.g., 08:00).

33.5.4.112 const DateOffset_T stdair::DEFAULT_ADVANCE_PURCHASE

Default advance purchase (e.g., 22 days).

33.5.4.113 const Date_T stdair::DEFAULT_REQUEST_DATE

Default request date (e.g., 10-Jan-2011).

33.5.4.114 const Duration_T stdair::DEFAULT_REQUEST_TIME

Default preferred departure time (e.g., 08:00).

33.5.4.115 const DateTime_T stdair::DEFAULT_REQUEST_DATE_TIME

Default request date-time (e.g., 08:00).

33.5.4.116 const CabinCode_T stdair::DEFAULT_PREFERRED_CABIN

Default preferred cabin (e.g., 'M').

33.5.4.117 const ChannelLabel_T stdair::CHANNEL_DN

DN channel (e.g., direct on-line).

33.5.4.118 const ChannelLabel_T stdair::CHANNEL_IN

IN channel (e.g., indirect on-line).

33.5.4.119 const TripType_T stdair::TRIP_TYPE_ONE_WAY

Trip type one-way (e.g., "OW").

Referenced by [stdair::BookingRequestStruct::display\(\)](#).

33.5.4.120 const TripType_T stdair::TRIP_TYPE_ROUND_TRIP

Trip type round-trip (e.g., "RT").

Referenced by [stdair::YieldFeatures::isTripTypeValid\(\)](#), and [stdair::FareFeatures::isTripTypeValid\(\)](#).

33.5.4.121 `const TripType_T stdair::TRIP_TYPE_INBOUND`

Trip type inbound (e.g., "RI").

Referenced by [stdair::YieldFeatures::isTripTypeValid\(\)](#), and [stdair::FareFeatures::isTripTypeValid\(\)](#).

33.5.4.122 `const TripType_T stdair::TRIP_TYPE_OUTBOUND`

Trip type outbound (e.g., "RO").

Referenced by [stdair::YieldFeatures::isTripTypeValid\(\)](#), and [stdair::FareFeatures::isTripTypeValid\(\)](#).

33.5.4.123 `const FrequentFlyer_T stdair::DEFAULT_FF_TIER`

Default frequent flyer tier (e.g., non member).

33.5.4.124 `const PriceValue_T stdair::DEFAULT_VALUE_OF_TIME`

Default value of time (expressed as a monetary unit per hour).

33.5.4.125 `const Duration_T stdair::DEFAULT_MINIMAL_CONNECTION_TIME`

Default Minimal connection time.

33.5.4.126 `const Duration_T stdair::DEFAULT_MAXIMAL_CONNECTION_TIME`

Default maximal connection time.

33.5.4.127 `const FlightPathCode_T stdair::DEFAULT_FLIGHTPATH_CODE`

Default flightPathCode value ("").

33.5.4.128 `const Availability_T stdair::DEFAULT_CLASS_AVAILABILITY`

Default value of Availability.

33.5.4.129 `const AvailabilityStatus_T stdair::DEFAULT_AVAILABILITY_STATUS`

Default availability status for a travel solution.

33.5.4.130 `const unsigned short stdair::DEFAULT_NUMBER_OF_REQUIRED_SEATS`

Default number of required seats by the demand.

33.5.4.131 `const MatchingIndicator_T stdair::DEFAULT_MATCHING_INDICATOR`

Default Matching Indicator value between customer requirements and a fare rule.

33.5.4.132 `const AirlineCode_T stdair::DEFAULT_DICO_STUDIED_AIRLINE`

Default DICO studied airline.

33.5.4.133 const Yield_T stdair::DEFAULT_YIELD_VALUE

Default yield value.

33.5.4.134 const Yield_T stdair::DEFAULT_YIELD_MAX_VALUE

Default yield max value.

33.6 stdair::LOG Namespace Reference

Enumerations

- enum [EN_LogLevel](#) {
 [CRITICAL](#) = 0, [ERROR](#), [NOTIFICATION](#), [WARNING](#),
 [DEBUG](#), [VERBOSE](#), [LAST_VALUE](#) }

Variables

- static const std::string [_logLevels](#) [[LAST_VALUE](#)]

33.6.1 Detailed Description

Level of logs.

33.6.2 Enumeration Type Documentation

33.6.2.1 enum stdair::LOG::EN_LogLevel

Enumerator:

CRITICAL
ERROR
NOTIFICATION
WARNING
DEBUG
VERBOSE
LAST_VALUE

Definition at line 18 of file [stdair_log.hpp](#).

33.6.3 Variable Documentation

33.6.3.1 const std::string stdair::LOG::_logLevels[[LAST_VALUE](#)] [[static](#)]

Initial value:

```
{ "C", "E", "N", "W", "D", "V" }
```

Definition at line 28 of file [stdair_log.hpp](#).

Referenced by [stdair::Logger::log\(\)](#), [stdair::BasLogParams::toShortString\(\)](#), and [stdair::BasLogParams::toString\(\)](#).

33.7 stdair_test Namespace Reference

Classes

- struct [BookingClass](#)
- struct [Cabin](#)

33.7.1 Detailed Description

Namespace gathering classes and structures for test purposes

33.8 swift Namespace Reference

The wrapper namespace.

Classes

- class [SKeymap](#)
The readline keymap wrapper.
- class [SReadline](#)
The readline library wrapper.

33.8.1 Detailed Description

The wrapper namespace. The namespace is also used for other library elements.

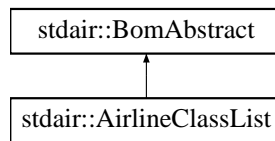
34 Class Documentation

34.1 stdair::AirlineClassList Class Reference

Class representing the actual attributes for a segment-features.

```
#include <stdair/bom/AirlineClassList.hpp>
```

Inheritance diagram for `stdair::AirlineClassList`:



Public Types

- typedef [AirlineClassListKey](#) [Key_T](#)

Public Member Functions

- const [Key_T](#) & [getKey](#) () const
- [BomAbstract](#) *const [getParent](#) () const
- const [AirlineCodeList_T](#) & [getAirlineCodeList](#) () const
- const [ClassList_StringList_T](#) & [getClassCodeList](#) () const
- const [HolderMap_T](#) & [getHolderMap](#) () const
- const [stdair::Yield_T](#) & [getYield](#) () const
- const [stdair::Fare_T](#) & [getFare](#) () const
- void [setYield](#) (const [Yield_T](#) &iYield)
- void [setFare](#) (const [Fare_T](#) &iFare)
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- std::string [toString](#) () const
- const std::string [describeKey](#) () const
- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int iFileVersion)

Protected Member Functions

- [AirlineClassList](#) (const [Key_T](#) &)
- virtual [~AirlineClassList](#) ()

Protected Attributes

- [Key_T_key](#)
- [BomAbstract](#) * [_parent](#)
- [HolderMap_T_holderMap](#)
- [Yield_T_yield](#)
- [Fare_T_fare](#)

Friends

- class [FacBom](#)
- class [FacBomManager](#)
- class [boost::serialization::access](#)

34.1.1 Detailed Description

Class representing the actual attributes for a segment-features.

Definition at line 27 of file [AirlineClassList.hpp](#).

34.1.2 Member Typedef Documentation

34.1.2.1 typedef AirlineClassListKey stdair::AirlineClassList::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 37 of file [AirlineClassList.hpp](#).

34.1.3 Constructor & Destructor Documentation

34.1.3.1 stdair::AirlineClassList::AirlineClassList (const Key_T & iKey) [protected]

Main constructor.

Definition at line 32 of file [AirlineClassList.cpp](#).

34.1.3.2 stdair::AirlineClassList::~~AirlineClassList () [protected, virtual]

Destructor.

Definition at line 37 of file [AirlineClassList.cpp](#).

34.1.4 Member Function Documentation

34.1.4.1 const Key_T& stdair::AirlineClassList::getKey () const [inline]

Get the airline class list key.

Definition at line 43 of file [AirlineClassList.hpp](#).

References [_key](#).

34.1.4.2 BomAbstract* const stdair::AirlineClassList::getParent () const [inline]

Get the parent object.

Definition at line 48 of file [AirlineClassList.hpp](#).

References [_parent](#).

34.1.4.3 const AirlineCodeList_T& stdair::AirlineClassList::getAirlineCodeList () const [inline]

Get the airline code list (part of the primary key).

Definition at line 53 of file [AirlineClassList.hpp](#).

References [_key](#), and [stdair::AirlineClassListKey::getAirlineCodeList\(\)](#).

34.1.4.4 `const ClassList_StringList_T& stdair::AirlineClassList::getClassCodeList () const [inline]`

Get the class code list (part of the primary key).

Definition at line 58 of file [AirlineClassList.hpp](#).

References [_key](#), and [stdair::AirlineClassListKey::getClassCodeList\(\)](#).

34.1.4.5 `const HolderMap_T& stdair::AirlineClassList::getHolderMap () const [inline]`

Get the map of children holders.

Definition at line 63 of file [AirlineClassList.hpp](#).

References [_holderMap](#).

34.1.4.6 `const stdair::Yield_T& stdair::AirlineClassList::getYield () const [inline]`

Get the yield.

Definition at line 68 of file [AirlineClassList.hpp](#).

References [_yield](#).

34.1.4.7 `const stdair::Fare_T& stdair::AirlineClassList::getFare () const [inline]`

Get the fare.

Definition at line 73 of file [AirlineClassList.hpp](#).

References [_fare](#).

34.1.4.8 `void stdair::AirlineClassList::setYield (const Yield_T & iYield) [inline]`

Definition at line 79 of file [AirlineClassList.hpp](#).

References [_yield](#).

34.1.4.9 `void stdair::AirlineClassList::setFare (const Fare_T & iFare) [inline]`

Definition at line 83 of file [AirlineClassList.hpp](#).

References [_fare](#).

34.1.4.10 `void stdair::AirlineClassList::toStream (std::ostream & ioOut) const [inline, virtual]`

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Implements [stdair::BomAbstract](#).

Definition at line 94 of file [AirlineClassList.hpp](#).

References [toString\(\)](#).

34.1.4.11 `void stdair::AirlineClassList::fromStream (std::istream & ioIn)` `[inline, virtual]`

Read a Business Object from an input stream.

Parameters

<i>istream&</i> the input stream.

Implements [stdair::BomAbstract](#).

Definition at line 103 of file [AirlineClassList.hpp](#).

34.1.4.12 `std::string stdair::AirlineClassList::toString () const` `[virtual]`

Get the serialised version of the Business Object.

Implements [stdair::BomAbstract](#).

Definition at line 41 of file [AirlineClassList.cpp](#).

References [_fare](#), [_yield](#), and [describeKey\(\)](#).

Referenced by [toStream\(\)](#).

34.1.4.13 `const std::string stdair::AirlineClassList::describeKey () const` `[inline]`

Get a string describing the key.

Definition at line 114 of file [AirlineClassList.hpp](#).

References [_key](#), and [stdair::AirlineClassListKey::toString\(\)](#).

Referenced by [toString\(\)](#).

34.1.4.14 `template<class Archive > void stdair::AirlineClassList::serialize (Archive & ar, const unsigned int iFileVersion)`

Serialisation.

Definition at line 63 of file [AirlineClassList.cpp](#).

References [_fare](#), [_key](#), and [_yield](#).

34.1.5 Friends And Related Function Documentation

34.1.5.1 `friend class FacBom` `[friend]`

Definition at line 28 of file [AirlineClassList.hpp](#).

34.1.5.2 `friend class FacBomManager` `[friend]`

Definition at line 29 of file [AirlineClassList.hpp](#).

34.1.5.3 friend class boost::serialization::access [friend]

Definition at line 30 of file [AirlineClassList.hpp](#).

34.1.6 Member Data Documentation

34.1.6.1 Key_T stdair::AirlineClassList::_key [protected]

Primary key (flight number and departure date).

Definition at line 164 of file [AirlineClassList.hpp](#).

Referenced by [describeKey\(\)](#), [getAirlineCodeList\(\)](#), [getClassCodeList\(\)](#), [getKey\(\)](#), and [serialize\(\)](#).

34.1.6.2 BomAbstract* stdair::AirlineClassList::_parent [protected]

Pointer on the parent class ([Inventory](#)).

Definition at line 169 of file [AirlineClassList.hpp](#).

Referenced by [getParent\(\)](#).

34.1.6.3 HolderMap_T stdair::AirlineClassList::_holderMap [protected]

Map holding the children ([SegmentDate](#) and [LegDate](#) objects).

Definition at line 174 of file [AirlineClassList.hpp](#).

Referenced by [getHolderMap\(\)](#).

34.1.6.4 Yield_T stdair::AirlineClassList::_yield [protected]

Definition at line 179 of file [AirlineClassList.hpp](#).

Referenced by [getYield\(\)](#), [serialize\(\)](#), [setYield\(\)](#), and [toString\(\)](#).

34.1.6.5 Fare_T stdair::AirlineClassList::_fare [protected]

Definition at line 184 of file [AirlineClassList.hpp](#).

Referenced by [getFare\(\)](#), [serialize\(\)](#), [setFare\(\)](#), and [toString\(\)](#).

The documentation for this class was generated from the following files:

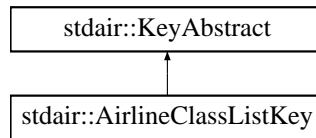
- [stdair/bom/AirlineClassList.hpp](#)
- [stdair/bom/AirlineClassList.cpp](#)

34.2 stdair::AirlineClassListKey Struct Reference

Key of airport-pair.

```
#include <stdair/bom/AirlineClassListKey.hpp>
```

Inheritance diagram for stdair::AirlineClassListKey:



Public Member Functions

- [AirlineClassListKey](#) (const [AirlineCodeList_T](#) &, const [ClassList_StringList_T](#) &)
- [AirlineClassListKey](#) (const [AirlineClassListKey](#) &)
- [~AirlineClassListKey](#) ()
- const [AirlineCodeList_T](#) & [getAirlineCodeList](#) () const
- const [ClassList_StringList_T](#) & [getClassCodeList](#) () const
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- const std::string [toString](#) () const
- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int iFileVersion)

Friends

- class [boost::serialization::access](#)

34.2.1 Detailed Description

Key of airport-pair.

Definition at line 25 of file [AirlineClassListKey.hpp](#).

34.2.2 Constructor & Destructor Documentation

34.2.2.1 `stdair::AirlineClassListKey::AirlineClassListKey (const AirlineCodeList_T & iAirlineCodeList, const ClassList_StringList_T & iClassCodeList)`

Constructor.

Definition at line 24 of file [AirlineClassListKey.cpp](#).

34.2.2.2 `stdair::AirlineClassListKey::AirlineClassListKey (const AirlineClassListKey & iKey)`

Copy constructor.

Definition at line 30 of file [AirlineClassListKey.cpp](#).

34.2.2.3 stdair::AirlineClassListKey::~~AirlineClassListKey ()

Destructor.

Definition at line 36 of file [AirlineClassListKey.cpp](#).

34.2.3 Member Function Documentation

34.2.3.1 const AirlineCodeList_T& stdair::AirlineClassListKey::getAirlineCodeList () const
[inline]

Get the airline code list.

Definition at line 56 of file [AirlineClassListKey.hpp](#).

Referenced by [stdair::AirlineClassList::getAirlineCodeList\(\)](#).

34.2.3.2 const ClassList_StringList_T& stdair::AirlineClassListKey::getClassCodeList ()
const [inline]

Get the class code list.

Definition at line 61 of file [AirlineClassListKey.hpp](#).

Referenced by [stdair::AirlineClassList::getClassCodeList\(\)](#).

34.2.3.3 void stdair::AirlineClassListKey::toStream (std::ostream & ioOut) const
[virtual]

Dump a Business Object Key into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 40 of file [AirlineClassListKey.cpp](#).

References [toString\(\)](#).

34.2.3.4 void stdair::AirlineClassListKey::fromStream (std::istream & ioIn) [virtual]

Read a Business Object Key from an input stream.

Parameters

<i>istream&</i> the input stream.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 45 of file [AirlineClassListKey.cpp](#).

34.2.3.5 `const std::string stdair::AirlineClassListKey::toString () const` [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 49 of file [AirlineClassListKey.cpp](#).

References [stdair::DEFAULT_KEY_SUB_FLD_DELIMITER](#).

Referenced by [stdair::AirlineClassList::describeKey\(\)](#), and [toStream\(\)](#).

34.2.3.6 `template<class Archive > void stdair::AirlineClassListKey::serialize (Archive & ar, const unsigned int iFileVersion)`

Serialisation.

Definition at line 86 of file [AirlineClassListKey.cpp](#).

34.2.4 Friends And Related Function Documentation

34.2.4.1 `friend class boost::serialization::access` [friend]

Definition at line 26 of file [AirlineClassListKey.hpp](#).

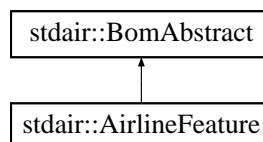
The documentation for this struct was generated from the following files:

- [stdair/bom/AirlineClassListKey.hpp](#)
- [stdair/bom/AirlineClassListKey.cpp](#)

34.3 stdair::AirlineFeature Class Reference

```
#include <stdair/bom/AirlineFeature.hpp>
```

Inheritance diagram for `stdair::AirlineFeature`:



Public Types

- typedef [AirlineFeatureKey](#) Key_T

Public Member Functions

- const [Key_T](#) & [getKey](#) () const
- void [init](#) (const [ForecasterMode_T](#) &, const [HistoricalDataLimit_T](#) &, const [ControlMode_T](#) &)
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- std::string [toString](#) () const
- const std::string [describeKey](#) () const

Protected Member Functions

- [AirlineFeature](#) ()
- [AirlineFeature](#) (const [AirlineFeature](#) &)
- [AirlineFeature](#) (const [Key_T](#) &)
- virtual [~AirlineFeature](#) ()

Protected Attributes

- [Key_T_key](#)
- [ForecasterMode_T_forecasterMode](#)
- [HistoricalDataLimit_T_historicalDataLimit](#)
- [ControlMode_T_controlMode](#)

Friends

- class [FacBom](#)

34.3.1 Detailed Description

Class representing the actual attributes for an airline booking class.

Definition at line 16 of file [AirlineFeature.hpp](#).

34.3.2 Member Typedef Documentation

34.3.2.1 typedef AirlineFeatureKey stdair::AirlineFeature::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 22 of file [AirlineFeature.hpp](#).

34.3.3 Constructor & Destructor Documentation

34.3.3.1 stdair::AirlineFeature::AirlineFeature () [protected]

Default constructors.

34.3.3.2 `stdair::AirlineFeature::AirlineFeature (const AirlineFeature &)` [protected]

34.3.3.3 `stdair::AirlineFeature::AirlineFeature (const Key_T & iKey)` [protected]

Definition at line 13 of file [AirlineFeature.cpp](#).

34.3.3.4 `stdair::AirlineFeature::~~AirlineFeature ()` [protected, virtual]

Destructor.

Definition at line 17 of file [AirlineFeature.cpp](#).

34.3.4 Member Function Documentation

34.3.4.1 `const Key_T& stdair::AirlineFeature::getKey () const` [inline]

Get the airline feature key.

Definition at line 27 of file [AirlineFeature.hpp](#).

References [_key](#).

34.3.4.2 `void stdair::AirlineFeature::init (const ForecasterMode_T & iForecasterMode, const HistoricalDataLimit_T & iHistoricalDataLimit, const ControlMode_T & iControlMode)`

Initialization method.

Definition at line 21 of file [AirlineFeature.cpp](#).

References [_controlMode](#), [_forecasterMode](#), and [_historicalDataLimit](#).

34.3.4.3 `void stdair::AirlineFeature::toStream (std::ostream & ioOut) const` [inline, virtual]

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Implements [stdair::BomAbstract](#).

Definition at line 41 of file [AirlineFeature.hpp](#).

References [toString\(\)](#).

34.3.4.4 `void stdair::AirlineFeature::fromStream (std::istream & ioIn)` [inline, virtual]

Read a Business Object from an input stream.

Parameters

<i>istream&</i> the input stream.

Implements [stdair::BomAbstract](#).

Definition at line 45 of file [AirlineFeature.hpp](#).

34.3.4.5 `std::string stdair::AirlineFeature::toString () const` `[virtual]`

Get the serialised version of the Business Object.

Implements [stdair::BomAbstract](#).

Definition at line 30 of file [AirlineFeature.cpp](#).

References [_controlMode](#), [_forecasterMode](#), [_historicalDataLimit](#), and [describeKey\(\)](#).

Referenced by [toStream\(\)](#).

34.3.4.6 `const std::string stdair::AirlineFeature::describeKey () const` `[inline]`

Get a string describing the key.

Definition at line 51 of file [AirlineFeature.hpp](#).

References [_key](#), and [stdair::AirlineFeatureKey::toString\(\)](#).

Referenced by [toString\(\)](#).

34.3.5 Friends And Related Function Documentation

34.3.5.1 `friend class FacBom` `[friend]`

Definition at line 17 of file [AirlineFeature.hpp](#).

34.3.6 Member Data Documentation

34.3.6.1 `Key_T stdair::AirlineFeature::_key` `[protected]`

The key of both structure and objects.

Definition at line 64 of file [AirlineFeature.hpp](#).

Referenced by [describeKey\(\)](#), and [getKey\(\)](#).

34.3.6.2 `ForecasterMode_T stdair::AirlineFeature::_forecasterMode`
`[protected]`

The type of forecaster.

Definition at line 67 of file [AirlineFeature.hpp](#).

Referenced by [init\(\)](#), and [toString\(\)](#).

34.3.6.3 `HistoricalDataLimit_T stdair::AirlineFeature::_historicalDataLimit`
`[protected]`

The size of the moving average window.

Definition at line 70 of file [AirlineFeature.hpp](#).

Referenced by [init\(\)](#), and [toString\(\)](#).

34.3.6.4 ControlMode_T stdair::AirlineFeature::_controlMode [protected]

The type of inventory control.

Definition at line 73 of file [AirlineFeature.hpp](#).

Referenced by [init\(\)](#), and [toString\(\)](#).

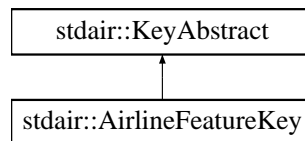
The documentation for this class was generated from the following files:

- [stdair/bom/AirlineFeature.hpp](#)
- [stdair/bom/AirlineFeature.cpp](#)

34.4 stdair::AirlineFeatureKey Struct Reference

```
#include <stdair/bom/AirlineFeatureKey.hpp>
```

Inheritance diagram for stdair::AirlineFeatureKey:



Public Member Functions

- [AirlineFeatureKey](#) (const [AirlineCode_T](#) &iAirlineCode)
- [~AirlineFeatureKey](#) ()
- const [AirlineCode_T](#) & [getAirlineCode](#) () const
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- const std::string [toString](#) () const

34.4.1 Detailed Description

Key of [AirlineFeature](#).

Definition at line 15 of file [AirlineFeatureKey.hpp](#).

34.4.2 Constructor & Destructor Documentation

34.4.2.1 stdair::AirlineFeatureKey::AirlineFeatureKey (const [AirlineCode_T](#) & iAirlineCode)

Constructor.

Definition at line 12 of file [AirlineFeatureKey.cpp](#).

34.4.2.2 stdair::AirlineFeatureKey::~~AirlineFeatureKey ()

Destructor.

Definition at line 17 of file [AirlineFeatureKey.cpp](#).

34.4.3 Member Function Documentation

34.4.3.1 const AirlineCode_T& stdair::AirlineFeatureKey::getAirlineCode () const [inline]

Get the airline code.

Definition at line 27 of file [AirlineFeatureKey.hpp](#).

34.4.3.2 void stdair::AirlineFeatureKey::toStream (std::ostream & ioOut) const [virtual]

Dump a Business Object Key into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 21 of file [AirlineFeatureKey.cpp](#).

References [toString\(\)](#).

34.4.3.3 void stdair::AirlineFeatureKey::fromStream (std::istream & ioIn) [virtual]

Read a Business Object Key from an input stream.

Parameters

<i>istream&</i> the input stream.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 26 of file [AirlineFeatureKey.cpp](#).

34.4.3.4 const std::string stdair::AirlineFeatureKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 30 of file [AirlineFeatureKey.cpp](#).

Referenced by [stdair::AirlineFeature::describeKey\(\)](#), and [toStream\(\)](#).

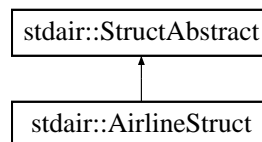
The documentation for this struct was generated from the following files:

- [stdair/bom/AirlineFeatureKey.hpp](#)
- [stdair/bom/AirlineFeatureKey.cpp](#)

34.5 stdair::AirlineStruct Struct Reference

```
#include <stdair/bom/AirlineStruct.hpp>
```

Inheritance diagram for stdair::AirlineStruct:



Public Member Functions

- const [AirlineCode_T](#) & [getAirlineCode](#) () const
- const std::string & [getAirlineName](#) () const
- void [setAirlineCode](#) (const [AirlineCode_T](#) & iAirlineCode)
- void [setAirlineName](#) (const std::string & iAirlineName)
- void [toStream](#) (std::ostream & ioOut) const
- void [fromStream](#) (std::istream & ioIn)
- const std::string [describe](#) () const
- [AirlineStruct](#) (const [AirlineCode_T](#) &, const std::string & iAirlineName)
- [AirlineStruct](#) ()
- [AirlineStruct](#) (const [AirlineStruct](#) &)
- [~AirlineStruct](#) ()

34.5.1 Detailed Description

Structure holding parameters describing an airline.

Definition at line 18 of file [AirlineStruct.hpp](#).

34.5.2 Constructor & Destructor Documentation

34.5.2.1 stdair::AirlineStruct::AirlineStruct (const [AirlineCode_T](#) & iAirlineCode, const std::string & iAirlineName)

Main constructor.

Definition at line 24 of file [AirlineStruct.cpp](#).

34.5.2.2 stdair::AirlineStruct::AirlineStruct ()

Default constructor.

Definition at line 15 of file [AirlineStruct.cpp](#).

34.5.2.3 stdair::AirlineStruct::AirlineStruct (const AirlineStruct & iAirlineStruct)

Default copy constructor.

Definition at line 19 of file [AirlineStruct.cpp](#).

34.5.2.4 stdair::AirlineStruct::~~AirlineStruct ()

Destructor.

Definition at line 30 of file [AirlineStruct.cpp](#).

34.5.3 Member Function Documentation

34.5.3.1 const AirlineCode_T& stdair::AirlineStruct::getAirlineCode () const [inline]

Get the airline code.

Definition at line 22 of file [AirlineStruct.hpp](#).

Referenced by [soci::type_conversion< stdair::AirlineStruct >::to_base\(\)](#), and [stdair::DBManagerForAirlines::updateAirli](#)

34.5.3.2 const std::string& stdair::AirlineStruct::getAirlineName () const [inline]

Get the airline name.

Definition at line 27 of file [AirlineStruct.hpp](#).

Referenced by [soci::type_conversion< stdair::AirlineStruct >::to_base\(\)](#), and [stdair::DBManagerForAirlines::updateAirli](#)

34.5.3.3 void stdair::AirlineStruct::setAirlineCode (const AirlineCode_T & iAirlineCode) [inline]

Set the airline code.

Definition at line 33 of file [AirlineStruct.hpp](#).

Referenced by [soci::type_conversion< stdair::AirlineStruct >::from_base\(\)](#).

34.5.3.4 void stdair::AirlineStruct::setAirlineName (const std::string & iAirlineName) [inline]

Set the airline name.

Definition at line 38 of file [AirlineStruct.hpp](#).

Referenced by [soci::type_conversion< stdair::AirlineStruct >::from_base\(\)](#).

34.5.3.5 void stdair::AirlineStruct::toStream (std::ostream & *ioOut*) const

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Reimplemented from [stdair::StructAbstract](#).

Definition at line 34 of file [AirlineStruct.cpp](#).

References [describe\(\)](#).

34.5.3.6 void stdair::AirlineStruct::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object from an input stream.

Parameters

<i>istream&</i> the input stream.

Reimplemented from [stdair::StructAbstract](#).

Definition at line 39 of file [AirlineStruct.cpp](#).

34.5.3.7 const std::string stdair::AirlineStruct::describe () const [virtual]

Display of the structure.

Implements [stdair::StructAbstract](#).

Definition at line 43 of file [AirlineStruct.cpp](#).

Referenced by [toStream\(\)](#).

The documentation for this struct was generated from the following files:

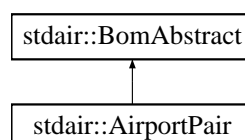
- [stdair/bom/AirlineStruct.hpp](#)
- [stdair/bom/AirlineStruct.cpp](#)

34.6 stdair::AirportPair Class Reference

Class representing the actual attributes for an airport-pair.

```
#include <stdair/bom/AirportPair.hpp>
```

Inheritance diagram for stdair::AirportPair:



Public Types

- typedef [AirportPairKey](#) [Key_T](#)

Public Member Functions

- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- std::string [toString](#) () const
- const std::string [describeKey](#) () const
- const [Key_T](#) & [getKey](#) () const
- const [AirportCode_T](#) & [getBoardingPoint](#) () const
- const [AirportCode_T](#) & [getOffPoint](#) () const
- [BomAbstract](#) *const [getParent](#) () const
- const [HolderMap_T](#) & [getHolderMap](#) () const

Protected Member Functions

- [AirportPair](#) (const [Key_T](#) &)
- virtual [~AirportPair](#) ()

Protected Attributes

- [Key_T _key](#)
- [BomAbstract](#) * [_parent](#)
- [HolderMap_T _holderMap](#)

Friends

- class [FacBom](#)
- class [FacBomManager](#)

34.6.1 Detailed Description

Class representing the actual attributes for an airport-pair.

Definition at line 18 of file [AirportPair.hpp](#).

34.6.2 Member Typedef Documentation

34.6.2.1 typedef [AirportPairKey](#) stdair::AirportPair::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 27 of file [AirportPair.hpp](#).

34.6.3 Constructor & Destructor Documentation

34.6.3.1 stdair::AirportPair::AirportPair (const Key_T & iKey) [protected]

Main constructor.

Definition at line 28 of file [AirportPair.cpp](#).

34.6.3.2 stdair::AirportPair::~~AirportPair () [protected, virtual]

Destructor.

Definition at line 33 of file [AirportPair.cpp](#).

34.6.4 Member Function Documentation

34.6.4.1 void stdair::AirportPair::toStream (std::ostream & ioOut) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Implements [stdair::BomAbstract](#).

Definition at line 36 of file [AirportPair.hpp](#).

References [toString\(\)](#).

34.6.4.2 void stdair::AirportPair::fromStream (std::istream & iIn) [inline, virtual]

Read a Business Object from an input stream.

Parameters

<i>istream&</i> the input stream.

Implements [stdair::BomAbstract](#).

Definition at line 45 of file [AirportPair.hpp](#).

34.6.4.3 std::string stdair::AirportPair::toString () const [virtual]

Get the serialised version of the Business Object.

Implements [stdair::BomAbstract](#).

Definition at line 37 of file [AirportPair.cpp](#).

References [describeKey\(\)](#).

Referenced by [toStream\(\)](#).

34.6.4.4 `const std::string stdair::AirportPair::describeKey () const` `[inline]`

Get a string describing the key.

Definition at line 56 of file [AirportPair.hpp](#).

References [_key](#), and [stdair::AirportPairKey::toString\(\)](#).

Referenced by [toString\(\)](#).

34.6.4.5 `const Key_T& stdair::AirportPair::getKey () const` `[inline]`

Get the primary key (origin airport, destination airport).

Definition at line 65 of file [AirportPair.hpp](#).

References [_key](#).

34.6.4.6 `const AirportCode_T& stdair::AirportPair::getBoardingPoint () const`
`[inline]`

Get the origin airport.

Definition at line 72 of file [AirportPair.hpp](#).

References [_key](#), and [stdair::AirportPairKey::getBoardingPoint\(\)](#).

34.6.4.7 `const AirportCode_T& stdair::AirportPair::getOffPoint () const` `[inline]`

Get the destination airport.

Definition at line 79 of file [AirportPair.hpp](#).

References [_key](#), and [stdair::AirportPairKey::getOffPoint\(\)](#).

34.6.4.8 `BomAbstract* const stdair::AirportPair::getParent () const` `[inline]`

Get a reference on the parent object instance.

Definition at line 86 of file [AirportPair.hpp](#).

References [_parent](#).

34.6.4.9 `const HolderMap_T& stdair::AirportPair::getHolderMap () const` `[inline]`

Get a reference on the children holder.

Definition at line 93 of file [AirportPair.hpp](#).

References [_holderMap](#).

34.6.5 Friends And Related Function Documentation

34.6.5.1 `friend class FacBom` `[friend]`

Definition at line 19 of file [AirportPair.hpp](#).

34.6.5.2 friend class FacBomManager [friend]

Definition at line 20 of file [AirportPair.hpp](#).

34.6.6 Member Data Documentation

34.6.6.1 Key_T stdair::AirportPair::_key [protected]

Primary key (flight number and departure date).

Definition at line 123 of file [AirportPair.hpp](#).

Referenced by [describeKey\(\)](#), [getBoardingPoint\(\)](#), [getKey\(\)](#), and [getOffPoint\(\)](#).

34.6.6.2 BomAbstract* stdair::AirportPair::_parent [protected]

Pointer on the parent class ([Inventory](#)).

Definition at line 128 of file [AirportPair.hpp](#).

Referenced by [getParent\(\)](#).

34.6.6.3 HolderMap_T stdair::AirportPair::_holderMap [protected]

Map holding the children.

Definition at line 133 of file [AirportPair.hpp](#).

Referenced by [getHolderMap\(\)](#).

The documentation for this class was generated from the following files:

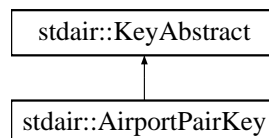
- [stdair/bom/AirportPair.hpp](#)
- [stdair/bom/AirportPair.cpp](#)

34.7 stdair::AirportPairKey Struct Reference

Key of airport-pair.

```
#include <stdair/bom/AirportPairKey.hpp>
```

Inheritance diagram for stdair::AirportPairKey:



Public Member Functions

- [AirportPairKey](#) (const [stdair::AirportCode_T](#) &, const [stdair::AirportCode_T](#) &)

- [AirportPairKey](#) (const [AirportPairKey](#) &)
- [~AirportPairKey](#) ()
- const [stdair::AirportCode_T](#) & [getBoardingPoint](#) () const
- const [stdair::AirportCode_T](#) & [getOffPoint](#) () const
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- const std::string [toString](#) () const

34.7.1 Detailed Description

Key of airport-pair.

Definition at line 16 of file [AirportPairKey.hpp](#).

34.7.2 Constructor & Destructor Documentation

34.7.2.1 `stdair::AirportPairKey::AirportPairKey (const stdair::AirportCode_T & iBoardingPoint, const stdair::AirportCode_T & iOffPoint)`

Main constructor.

Definition at line 22 of file [AirportPairKey.cpp](#).

34.7.2.2 `stdair::AirportPairKey::AirportPairKey (const AirportPairKey & iKey)`

Copy constructor.

Definition at line 28 of file [AirportPairKey.cpp](#).

34.7.2.3 `stdair::AirportPairKey::~~AirportPairKey ()`

Destructor.

Definition at line 34 of file [AirportPairKey.cpp](#).

34.7.3 Member Function Documentation

34.7.3.1 `const stdair::AirportCode_T& stdair::AirportPairKey::getBoardingPoint () const`
[inline]

Get the boarding point.

Definition at line 36 of file [AirportPairKey.hpp](#).

Referenced by [stdair::AirportPair::getBoardingPoint\(\)](#).

34.7.3.2 `const stdair::AirportCode_T& stdair::AirportPairKey::getOffPoint () const`
[inline]

Get the arrival point.

Definition at line 43 of file [AirportPairKey.hpp](#).

Referenced by [stdair::AirportPair::getOffPoint\(\)](#).

34.7.3.3 void stdair::AirportPairKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 38 of file [AirportPairKey.cpp](#).

References [toString\(\)](#).

34.7.3.4 void stdair::AirportPairKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters

<i>istream&</i> the input stream.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 43 of file [AirportPairKey.cpp](#).

34.7.3.5 const std::string stdair::AirportPairKey::toString () const [virtual]

Get the serialised version of the Business Object Key. That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 47 of file [AirportPairKey.cpp](#).

References [stdair::DEFAULT_KEY_SUB_FLD_DELIMITER](#).

Referenced by [stdair::AirportPair::describeKey\(\)](#), [stdair::BomRetriever::retrieveAirportPairFromKeySet\(\)](#), and [toStream\(\)](#).

The documentation for this struct was generated from the following files:

- [stdair/bom/AirportPairKey.hpp](#)
- [stdair/bom/AirportPairKey.cpp](#)

34.8 stdair::BasChronometer Struct Reference

```
#include <stdair/basic/BasChronometer.hpp>
```

Public Member Functions

- [BasChronometer](#) ()

- void [start](#) ()
- std::string [getStart](#) () const
- double [elapsed](#) () const

34.8.1 Detailed Description

Structure allowing measuring the time elapsed between two events.

Definition at line 14 of file [BasChronometer.hpp](#).

34.8.2 Constructor & Destructor Documentation

34.8.2.1 stdair::BasChronometer::BasChronometer ()

Constructor.

Definition at line 12 of file [BasChronometer.cpp](#).

34.8.3 Member Function Documentation

34.8.3.1 void stdair::BasChronometer::start ()

Start the chronometer from the local time

The elapsed time given is the one elapsed since the start is launched.

Definition at line 16 of file [BasChronometer.cpp](#).

34.8.3.2 std::string stdair::BasChronometer::getStart () const [inline]

Get the start time.

Definition at line 24 of file [BasChronometer.hpp](#).

34.8.3.3 double stdair::BasChronometer::elapsed () const

Return the time elapsed since the structure has been instanciated.

That elapsed time is expressed in seconds.

Definition at line 26 of file [BasChronometer.cpp](#).

The documentation for this struct was generated from the following files:

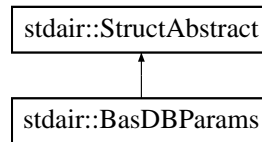
- stdair/basic/[BasChronometer.hpp](#)
- stdair/basic/[BasChronometer.cpp](#)

34.9 stdair::BasDBParams Struct Reference

Structure holding the parameters for connection to a database.

```
#include <stdair/basic/BasDBParams.hpp>
```

Inheritance diagram for stdair::BasDBParams:



Public Member Functions

- const std::string & [getUser](#) () const
- const std::string & [getPassword](#) () const
- const std::string & [getHost](#) () const
- const std::string & [getPort](#) () const
- const std::string & [getDBName](#) () const
- void [setUser](#) (const std::string &iUser)
- void [setPassword](#) (const std::string &iPasswd)
- void [setHost](#) (const std::string &iHost)
- void [setPort](#) (const std::string &iPort)
- void [setDBName](#) (const std::string &iDBName)
- bool [check](#) () const
- const std::string [describe](#) () const
- std::string [toShortString](#) () const
- std::string [toString](#) () const
- [BasDBParams](#) (const std::string &iDBUser, const std::string &iDBPasswd, const std::string &iDBHost, const std::string &iDBPort, const std::string &iDBName)
- [BasDBParams](#) ()
- [BasDBParams](#) (const [BasDBParams](#) &)
- [~BasDBParams](#) ()
- void [toStream](#) (std::ostream &ioOut) const
- virtual void [fromStream](#) (std::istream &ioIn)

34.9.1 Detailed Description

Structure holding the parameters for connection to a database.

Definition at line 19 of file [BasDBParams.hpp](#).

34.9.2 Constructor & Destructor Documentation

- 34.9.2.1 `stdair::BasDBParams::BasDBParams (const std::string & iDBUser, const std::string & iDBPasswd, const std::string & iDBHost, const std::string & iDBPort, const std::string & iDBName)`

Main Constructor.

Definition at line 24 of file [BasDBParams.cpp](#).

34.9.2.2 stdair::BasDBParams::BasDBParams ()

Default Constructor.

Definition at line 13 of file [BasDBParams.cpp](#).

34.9.2.3 stdair::BasDBParams::BasDBParams (const BasDBParams & iDBParams)

Default copy constructor.

Definition at line 17 of file [BasDBParams.cpp](#).

34.9.2.4 stdair::BasDBParams::~~BasDBParams ()

Destructor.

Definition at line 34 of file [BasDBParams.cpp](#).

34.9.3 Member Function Documentation

34.9.3.1 const std::string& stdair::BasDBParams::getUser () const [inline]

Get the database user name.

Definition at line 23 of file [BasDBParams.hpp](#).

34.9.3.2 const std::string& stdair::BasDBParams::getPassword () const [inline]

Get the database user password.

Definition at line 28 of file [BasDBParams.hpp](#).

34.9.3.3 const std::string& stdair::BasDBParams::getHost () const [inline]

Get the database host name.

Definition at line 33 of file [BasDBParams.hpp](#).

34.9.3.4 const std::string& stdair::BasDBParams::getPort () const [inline]

Get the database port number.

Definition at line 38 of file [BasDBParams.hpp](#).

34.9.3.5 const std::string& stdair::BasDBParams::getDBName () const [inline]

Get the database name.

Definition at line 43 of file [BasDBParams.hpp](#).

34.9.3.6 void stdair::BasDBParams::setUser (const std::string & iUser) [inline]

Set the database user name.

Definition at line 50 of file [BasDBParams.hpp](#).

34.9.3.7 void stdair::BasDBParams::setPassword (const std::string & *iPasswd*) [inline]

Set the database password.

Definition at line 55 of file [BasDBParams.hpp](#).

34.9.3.8 void stdair::BasDBParams::setHost (const std::string & *iHost*) [inline]

Set the database host name.

Definition at line 60 of file [BasDBParams.hpp](#).

34.9.3.9 void stdair::BasDBParams::setPort (const std::string & *iPort*) [inline]

Set the database port number.

Definition at line 65 of file [BasDBParams.hpp](#).

34.9.3.10 void stdair::BasDBParams::setDBName (const std::string & *iDBName*)
[inline]

Set the database name.

Definition at line 70 of file [BasDBParams.hpp](#).

34.9.3.11 bool stdair::BasDBParams::check () const

Check that all the parameters are fine.

Definition at line 57 of file [BasDBParams.cpp](#).

34.9.3.12 const std::string stdair::BasDBParams::describe () const [virtual]

Get the serialised version of the DBParams structure.

Implements [stdair::StructAbstract](#).

Definition at line 38 of file [BasDBParams.cpp](#).

References [toString\(\)](#).

34.9.3.13 std::string stdair::BasDBParams::toShortString () const

Get a short display of the DBParams structure.

Definition at line 43 of file [BasDBParams.cpp](#).

34.9.3.14 std::string stdair::BasDBParams::toString () const

Get the serialised version of the DBParams structure.

Definition at line 50 of file [BasDBParams.cpp](#).

Referenced by [describe\(\)](#).

34.9.3.15 `void stdair::StructAbstract::toStream (std::ostream & ioOut) const` `[inline, inherited]`

Dump a Business Object into an output stream.

Parameters

<i>ostream</i> & the output stream.

Reimplemented in [stdair::YieldRange](#), [stdair::AirlineStruct](#), [stdair::BookingRequestStruct](#), [stdair::CancellationStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::TravelSolutionStruct](#), and [stdair::VirtualClassStruct](#).

Definition at line 29 of file [StructAbstract.hpp](#).

References [stdair::StructAbstract::describe\(\)](#).

Referenced by [operator<<\(\)](#).

34.9.3.16 `virtual void stdair::StructAbstract::fromStream (std::istream & ioIn)` `[inline, virtual, inherited]`

Read a Business Object from an input stream.

Parameters

<i>istream</i> & the input stream.

Reimplemented in [stdair::ProgressStatusSet](#), [stdair::YieldRange](#), [stdair::AirlineStruct](#), [stdair::BookingRequestStruct](#), [stdair::CancellationStruct](#), [stdair::EventStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::TravelSolutionStruct](#), and [stdair::VirtualClassStruct](#).

Definition at line 38 of file [StructAbstract.hpp](#).

Referenced by [operator>>\(\)](#).

The documentation for this struct was generated from the following files:

- [stdair/basic/BasDBParams.hpp](#)
- [stdair/basic/BasDBParams.cpp](#)

34.10 stdair::BasFileMgr Struct Reference

```
#include <stdair/basic/BasFileMgr.hpp>
```

Static Public Member Functions

- static bool [doesExistAndIsReadable](#) (const std::string &iFilepath)

34.10.1 Detailed Description

Helper class for operations on files and on the file-system.

Definition at line 13 of file [BasFileMgr.hpp](#).

34.10.2 Member Function Documentation

34.10.2.1 `bool stdair::BasFileMgr::doesExistAndIsReadable (const std::string & iFilePath)`
`[static]`

Definition at line 23 of file [BasFileMgr.cpp](#).

The documentation for this struct was generated from the following files:

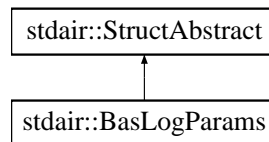
- [stdair/basic/BasFileMgr.hpp](#)
- [stdair/basic/BasFileMgr.cpp](#)

34.11 stdair::BasLogParams Struct Reference

Structure holding parameters for logging.

```
#include <stdair/basic/BasLogParams.hpp>
```

Inheritance diagram for stdair::BasLogParams:



Public Member Functions

- const [LOG::EN_LogLevel](#) & [getLogLevel](#) () const
- std::ostream & [getLogStream](#) () const
- const bool [getForcedInitialisationFlag](#) () const
- void [setForcedInitialisationFlag](#) (const bool iForceMultipleInstance)
- bool [check](#) () const
- const std::string [describe](#) () const
- std::string [toShortString](#) () const
- std::string [toString](#) () const
- [BasLogParams](#) (const [LOG::EN_LogLevel](#) iLogLevel, std::ostream &ioLogOutputStream, const bool iForceMultipleInstance=false)
- [BasLogParams](#) (const [BasLogParams](#) &)
- [~BasLogParams](#) ()
- void [toStream](#) (std::ostream &ioOut) const
- virtual void [fromStream](#) (std::istream &ioIn)

Friends

- class [Logger](#)

34.11.1 Detailed Description

Structure holding parameters for logging.

Definition at line 19 of file [BasLogParams.hpp](#).

34.11.2 Constructor & Destructor Documentation

34.11.2.1 `stdair::BasLogParams::BasLogParams (const LOG::EN_LogLevel iLogLevel,
std::ostream & ioLogOutputStream, const bool iForceMultipleInstance = false)`

Main Constructor.

Parameters

<i>in</i>	<i>const</i>	LOG::EN_LogLevel Level of the log (e.g., DEBUG)
	<i>inout</i>	std::ostream& (STL) Stream to log into.
<i>in</i>	<i>const</i>	bool Whether or not multiple initialisation should be forced.

Definition at line 27 of file [BasLogParams.cpp](#).

34.11.2.2 `stdair::BasLogParams::BasLogParams (const BasLogParams & iLogParams)`

Copy constructor.

Definition at line 21 of file [BasLogParams.cpp](#).

34.11.2.3 `stdair::BasLogParams::~BasLogParams ()`

Destructor.

Definition at line 35 of file [BasLogParams.cpp](#).

34.11.3 Member Function Documentation

34.11.3.1 `const LOG::EN_LogLevel& stdair::BasLogParams::getLogLevel () const
[inline]`

Get the log level.

Definition at line 26 of file [BasLogParams.hpp](#).

34.11.3.2 `std::ostream& stdair::BasLogParams::getLogStream () const [inline]`

Get the log output stream.

Definition at line 33 of file [BasLogParams.hpp](#).

34.11.3.3 `const bool stdair::BasLogParams::getForcedInitialisationFlag () const`
`[inline]`

State whether or not multiple initialisations are to be forced.

Definition at line 40 of file [BasLogParams.hpp](#).

34.11.3.4 `void stdair::BasLogParams::setForcedInitialisationFlag (const bool`
`iForceMultipleInstance) [inline]`

State whether or not multiple initialisations are to be forced.

Definition at line 49 of file [BasLogParams.hpp](#).

34.11.3.5 `bool stdair::BasLogParams::check () const`

Check that all the parameters are fine.

34.11.3.6 `const std::string stdair::BasLogParams::describe () const` `[virtual]`

Get the serialised version of the DBParams structure.

Implements [stdair::StructAbstract](#).

Definition at line 39 of file [BasLogParams.cpp](#).

References [toString\(\)](#).

34.11.3.7 `std::string stdair::BasLogParams::toShortString () const`

Get a short display of the LOGParams structure.

Definition at line 44 of file [BasLogParams.cpp](#).

References [stdair::LOG::_logLevels](#).

34.11.3.8 `std::string stdair::BasLogParams::toString () const`

Get the serialised version of the LOGParams structure.

Definition at line 52 of file [BasLogParams.cpp](#).

References [stdair::LOG::_logLevels](#).

Referenced by [describe\(\)](#).

34.11.3.9 `void stdair::StructAbstract::toStream (std::ostream & ioOut) const` `[inline,`
`inherited]`

Dump a Business Object into an output stream.

Parameters

<code>ostream&</code> the output stream.
--

Reimplemented in [stdair::YieldRange](#), [stdair::AirlineStruct](#), [stdair::BookingRequestStruct](#), [stdair::CancellationStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#),

[stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::TravelSolutionStruct](#), and [stdair::VirtualClassStruct](#).

Definition at line 29 of file [StructAbstract.hpp](#).

References [stdair::StructAbstract::describe\(\)](#).

Referenced by [operator<<\(\)](#).

```
34.11.3.10 virtual void stdair::StructAbstract::fromStream ( std::istream & ioln ) [inline,
virtual, inherited]
```

Read a Business Object from an input stream.

Parameters

<i>istream&</i> the input stream.

Reimplemented in [stdair::ProgressStatusSet](#), [stdair::YieldRange](#), [stdair::AirlineStruct](#), [stdair::BookingRequestStruct](#), [stdair::CancellationStruct](#), [stdair::EventStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::TravelSolutionStruct](#), and [stdair::VirtualClassStruct](#).

Definition at line 38 of file [StructAbstract.hpp](#).

Referenced by [operator>>\(\)](#).

34.11.4 Friends And Related Function Documentation

```
34.11.4.1 friend class Logger [friend]
```

Definition at line 20 of file [BasLogParams.hpp](#).

The documentation for this struct was generated from the following files:

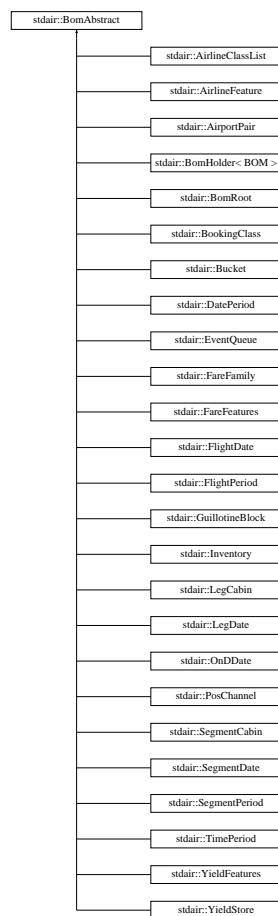
- [stdair/basic/BasLogParams.hpp](#)
- [stdair/basic/BasLogParams.cpp](#)

34.12 stdair::BomAbstract Class Reference

Base class for the Business Object Model (BOM) layer.

```
#include <stdair/bom/BomAbstract.hpp>
```

Inheritance diagram for [stdair::BomAbstract](#):



Public Member Functions

- virtual void [toStream](#) (std::ostream &ioOut) const =0
- virtual void [fromStream](#) (std::istream &ioIn)=0
- virtual std::string [toString](#) () const =0
- virtual [~BomAbstract](#) ()

Protected Member Functions

- [BomAbstract](#) ()
- [BomAbstract](#) (const [BomAbstract](#) &)

34.12.1 Detailed Description

Base class for the Business Object Model (BOM) layer.

Definition at line 23 of file [BomAbstract.hpp](#).

34.12.2 Constructor & Destructor Documentation

34.12.2.1 stdair::BomAbstract::BomAbstract () [inline, protected]

Protected Default Constructor to ensure this class is abstract.

Definition at line 40 of file [BomAbstract.hpp](#).

34.12.2.2 stdair::BomAbstract::BomAbstract (const BomAbstract &) [inline, protected]

Definition at line 41 of file [BomAbstract.hpp](#).

34.12.2.3 virtual stdair::BomAbstract::~~BomAbstract () [inline, virtual]

Destructor.

Definition at line 44 of file [BomAbstract.hpp](#).

34.12.3 Member Function Documentation

34.12.3.1 virtual void stdair::BomAbstract::toStream (std::ostream & ioOut) const [pure virtual]

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Implemented in [stdair::AirlineClassList](#), [stdair::AirlineFeature](#), [stdair::AirportPair](#), [stdair::BomHolder< BOM >](#), [stdair::BomRoot](#), [stdair::BookingClass](#), [stdair::Bucket](#), [stdair::DatePeriod](#), [stdair::EventQueue](#), [stdair::FareFamily](#), [stdair::FareFeatures](#), [stdair::FlightDate](#), [stdair::FlightPeriod](#), [stdair::GuillotineBlock](#), [stdair::Inventory](#), [stdair::LegCabin](#), [stdair::LegDate](#), [stdair::OnDDate](#), [stdair::PosChannel](#), [stdair::SegmentCabin](#), [stdair::SegmentDate](#), [stdair::SegmentPeriod](#), [stdair::TimePeriod](#), [stdair::YieldFeatures](#), and [stdair::YieldStore](#).

Referenced by [operator<<\(\)](#).

34.12.3.2 virtual void stdair::BomAbstract::fromStream (std::istream & ioln) [pure virtual]

Read a Business Object from an input stream.

Parameters

<i>istream&</i> the input stream.

Implemented in [stdair::AirlineClassList](#), [stdair::AirlineFeature](#), [stdair::AirportPair](#), [stdair::BomHolder< BOM >](#), [stdair::BomRoot](#), [stdair::BookingClass](#), [stdair::Bucket](#), [stdair::DatePeriod](#), [stdair::EventQueue](#), [stdair::FareFamily](#), [stdair::FareFeatures](#), [stdair::FlightDate](#), [stdair::FlightPeriod](#), [stdair::GuillotineBlock](#), [stdair::Inventory](#), [stdair::LegCabin](#), [stdair::LegDate](#), [stdair::OnDDate](#), [stdair::PosChannel](#), [stdair::SegmentCabin](#), [stdair::SegmentDate](#), [stdair::SegmentPeriod](#), [stdair::TimePeriod](#),

[stdair::YieldFeatures](#), and [stdair::YieldStore](#).

Referenced by [operator>>\(\)](#).

34.12.3.3 virtual std::string stdair::BomAbstract::toString () const [pure virtual]

Get the serialised version of the Business Object.

Implemented in [stdair::AirlineClassList](#), [stdair::AirlineFeature](#), [stdair::AirportPair](#), [stdair::BomHolder<BOM>](#), [stdair::BomRoot](#), [stdair::BookingClass](#), [stdair::Bucket](#), [stdair::DatePeriod](#), [stdair::EventQueue](#), [stdair::FareFamily](#), [stdair::FareFeatures](#), [stdair::FlightDate](#), [stdair::FlightPeriod](#), [stdair::GuillotineBlock](#), [stdair::Inventory](#), [stdair::LegCabin](#), [stdair::LegDate](#), [stdair::OnDDate](#), [stdair::PosChannel](#), [stdair::SegmentCabin](#), [stdair::SegmentDate](#), [stdair::SegmentPeriod](#), [stdair::TimePeriod](#), [stdair::YieldFeatures](#), and [stdair::YieldStore](#).

The documentation for this class was generated from the following file:

- [stdair/bom/BomAbstract.hpp](#)

34.13 stdair::BomArchive Class Reference

Utility class to archive/restore BOM objects with Boost serialisation.

```
#include <stdair/bom/BomArchive.hpp>
```

Static Public Member Functions

- static void [archive](#) (const [BomRoot](#) &)
- static std::string [archive](#) (const [Inventory](#) &)
- static void [restore](#) (const std::string &iArchive, [Inventory](#) &)
- static void [archive](#) (const [FlightDate](#) &)

34.13.1 Detailed Description

Utility class to archive/restore BOM objects with Boost serialisation.

Definition at line 28 of file [BomArchive.hpp](#).

34.13.2 Member Function Documentation

34.13.2.1 void stdair::BomArchive::archive (const [BomRoot](#) & *iBomRoot*) [static]

Recursively archive (dump in the underlying STL string) the objects of the BOM tree.

Parameters

<i>const</i> BomRoot & Root of the BOM tree to be archived.

Definition at line 32 of file [BomArchive.cpp](#).

34.13.2.2 `std::string stdair::BomArchive::archive (const Inventory & iInventory)`
`[static]`

Recursively archive (dump in the underlying STL string) the objects of the BOM tree.

Parameters

<code>const</code> Inventory & Root of the BOM tree to be archived.

Definition at line 36 of file [BomArchive.cpp](#).

34.13.2.3 `void stdair::BomArchive::restore (const std::string & iArchive, Inventory & iInventory)`
`[static]`

Recursively restore (from the underlying STL string) the objects of the BOM tree.

Parameters

<code>const</code> <code>std::string&</code> String holding the serialised objects.
<code>Inventory&</code> Root of the BOM tree to be restored.

Definition at line 44 of file [BomArchive.cpp](#).

34.13.2.4 `void stdair::BomArchive::archive (const FlightDate & iFlightDate)`
`[static]`

Recursively archive (dump in the underlying STL string) the objects of the BOM tree.

Parameters

<code>const</code> FlightDate & Root of the BOM tree to be archived.
--

Definition at line 52 of file [BomArchive.cpp](#).

The documentation for this class was generated from the following files:

- [stdair/bom/BomArchive.hpp](#)
- [stdair/bom/BomArchive.cpp](#)

34.14 stdair::BomDisplay Class Reference

Utility class to display StdAir objects with a pretty format.

```
#include <stdair/bom/BomDisplay.hpp>
```

Static Public Member Functions

- static `std::string` [csvDisplay](#) (const [EventQueue](#) &)
- static void [list](#) (std::ostream &, const [BomRoot](#) &, const [AirlineCode_T](#) & iAirlineCode="all", const [FlightNumber_T](#) & iFlightNumber=0)

- static void [list](#) (std::ostream &, const [Inventory](#) &, const unsigned short inventoryIndex=0, const [FlightNumber_T](#) & iFlightNumber=0)
- static void [listAirportPairDateRange](#) (std::ostream &, const [BomRoot](#) &)
- static void [csvDisplay](#) (std::ostream &, const [BomRoot](#) &)
- static void [csvDisplay](#) (std::ostream &, const [Inventory](#) &)
- static void [csvDisplay](#) (std::ostream &, const [OnDDate](#) &)
- static void [csvDisplay](#) (std::ostream &, const [FlightDate](#) &)
- static void [csvLegDateDisplay](#) (std::ostream &, const [FlightDate](#) &)
- static void [csvSegmentDateDisplay](#) (std::ostream &, const [FlightDate](#) &)
- static void [csvLegCabinDisplay](#) (std::ostream &, const [FlightDate](#) &)
- static void [csvSegmentCabinDisplay](#) (std::ostream &, const [FlightDate](#) &)
- static void [csvFareFamilyDisplay](#) (std::ostream &, const [FlightDate](#) &)
- static void [csvBucketDisplay](#) (std::ostream &, const [FlightDate](#) &)
- static void [csvBookingClassDisplay](#) (std::ostream &, const [BookingClass](#) &, const std::string & iLeadingString)
- static void [csvBookingClassDisplay](#) (std::ostream &, const [FlightDate](#) &)
- static void [csvDisplay](#) (std::ostream &, const [TravelSolutionList_T](#) &)
- static void [csvDisplay](#) (std::ostream &, const [DatePeriodList_T](#) &)
- static void [csvSimFQTAirRACDisplay](#) (std::ostream &, const [BomRoot](#) &)
- static void [csvAirportPairDisplay](#) (std::ostream &, const [AirportPair](#) &)
- static void [csvDateDisplay](#) (std::ostream &, const [DatePeriod](#) &)
- static void [csvPosChannelDisplay](#) (std::ostream &, const [PosChannel](#) &)
- static void [csvTimeDisplay](#) (std::ostream &, const [TimePeriod](#) &)
- template<typename FEATURE_TYPE >
static void [csvFeatureListDisplay](#) (std::ostream & oStream, const [TimePeriod](#) &)
- template<typename FEATURE_TYPE >
static void [csvFeaturesDisplay](#) (std::ostream & oStream, const FEATURE_TYPE &)
- static void [csvAirlineClassDisplay](#) (std::ostream &, const [AirlineClassList](#) &)

34.14.1 Detailed Description

Utility class to display StdAir objects with a pretty format.

Definition at line 39 of file [BomDisplay.hpp](#).

34.14.2 Member Function Documentation

34.14.2.1 static std::string stdair::BomDisplay::csvDisplay (const [EventQueue](#) &)
[static]

Recursively display (dump in the underlying output log stream) the objects of the BOM tree.

Parameters

const stdair::EventQueue & Root of the BOM tree to be displayed.
--

Returns

std::string Output string in which the BOM tree should be logged/dumped.

34.14.2.2 `static void stdair::BomDisplay::list (std::ostream & , const BomRoot & , const AirlineCode_T & iAirlineCode = "all", const FlightNumber_T & iFlightNumber = 0) [static]`

Display (dump in the underlying output log stream) the list of flight-dates contained within the given BOM tree.

Parameters

<code>std::ostream&</code>	Output stream in which the flight-date keys should be logged/dumped.
<code>const BomRoot&</code>	Root of the BOM tree to be displayed.
<code>const AirlineCode_T &</code>	Airline for which the flight-dates should be displayed. If set to "all" (default), all the inventories will be displayed.
<code>const FlightNumber_T &</code>	Flight number for which all the departure dates should be displayed. If set to 0 (the default), all the flight numbers will be displayed.

34.14.2.3 `static void stdair::BomDisplay::list (std::ostream & , const Inventory & , const unsigned short iInventoryIndex = 0, const FlightNumber_T & iFlightNumber = 0) [static]`

Display (dump in the underlying output log stream) the list of flight-dates contained within the given BOM tree.

Parameters

<code>std::ostream&</code>	Output stream in which the flight-date keys should be logged/dumped.
<code>const Inventory&</code>	Root of the BOM tree to be displayed.
<code>const unsigned short iInventoryIndex</code>	Index, within the list, of the inventory. It is 0 when that inventory is displayed alone.
<code>const FlightNumber_T &</code>	Flight number for which all the departure dates should be displayed. If set to 0 (the default), all the flight numbers will be displayed.

34.14.2.4 `static void stdair::BomDisplay::listAirportPairDateRange (std::ostream & , const BomRoot &) [static]`

Display the list of airports pairs and date ranges (contained within the BOM tree)

Parameters

<code>std::ostream&</code>	Output stream in which the airport pairs and date ranges are logged/dumped.
<code>const BomRoot&</code>	Root of the BOM tree to be displayed.

34.14.2.5 `static void stdair::BomDisplay::csvDisplay (std::ostream & , const BomRoot &)`
`[static]`

Recursively display (dump in the underlying output log stream) the objects of the BOM tree.

Parameters

<code>std::ostream&</code>	Output stream in which the BOM tree should be logged/dumped.
<code>const BomRoot&</code>	Root of the BOM tree to be displayed.

34.14.2.6 `static void stdair::BomDisplay::csvDisplay (std::ostream & , const Inventory &)`
`[static]`

Recursively display (dump in the underlying output log stream) the objects of the BOM tree from the level of the given [Inventory](#).

Parameters

<code>std::ostream&</code>	Output stream in which the BOM tree should be logged/dumped.
<code>const Inventory&</code>	Root of the BOM tree to be displayed.

34.14.2.7 `static void stdair::BomDisplay::csvDisplay (std::ostream & , const OnDDate &)`
`[static]`

Display the O&D date object information.

Parameters

<code>std::ostream&</code>	Output stream in which the BOM tree should be logged/dumped.
<code>const OnDDate&</code>	the BOM to be displayed.

34.14.2.8 `static void stdair::BomDisplay::csvDisplay (std::ostream & , const FlightDate &)`
`[static]`

Recursively display (dump in the underlying output log stream) the objects of the BOM tree from the level of the given [FlightDate](#).

Parameters

<code>std::ostream&</code>	Output stream in which the BOM tree should be logged/dumped.
<code>const FlightDate&</code>	Root of the BOM tree to be displayed.

34.14.2.9 `static void stdair::BomDisplay::csvLegDateDisplay (std::ostream & , const FlightDate &) [static]`

Recursively display (dump in the underlying output log stream) the leg-date level objects of the BOM tree.

Parameters

<i>std::ostream&</i>	Output stream in which the BOM tree should be logged/dumped.
<i>const</i> FlightDate &	Root of the BOM tree to be displayed.

34.14.2.10 `static void stdair::BomDisplay::csvSegmentDateDisplay (std::ostream & , const FlightDate &) [static]`

Recursively display (dump in the underlying output log stream) the segment-date level objects of the BOM tree.

Parameters

<i>std::ostream&</i>	Output stream in which the BOM tree should be logged/dumped.
<i>const</i> FlightDate &	Root of the BOM tree to be displayed.

34.14.2.11 `static void stdair::BomDisplay::csvLegCabinDisplay (std::ostream & , const FlightDate &) [static]`

Recursively display (dump in the underlying output log stream) the leg-cabin level objects of the BOM tree.

Parameters

<i>std::ostream&</i>	Output stream in which the BOM tree should be logged/dumped.
<i>const</i> FlightDate &	Root of the BOM tree to be displayed.

34.14.2.12 `static void stdair::BomDisplay::csvSegmentCabinDisplay (std::ostream & , const FlightDate &) [static]`

Recursively display (dump in the underlying output log stream) the segment-cabin level objects of the BOM tree.

Parameters

<i>std::ostream&</i>	Output stream in which the BOM tree should be logged/dumped.
<i>const</i> FlightDate &	Root of the BOM tree to be displayed.

34.14.2.13 static void stdair::BomDisplay::csvFareFamilyDisplay (std::ostream & , const **FlightDate** &) [static]

Recursively display (dump in the underlying output log stream) the fare families level objects of the BOM tree.

Parameters

<i>std::ostream&</i>	Output stream in which the BOM tree should be logged/dumped.
<i>const</i> FlightDate &	Root of the BOM tree to be displayed.

34.14.2.14 static void stdair::BomDisplay::csvBucketDisplay (std::ostream & , const **FlightDate** &) [static]

Recursively display (dump in the underlying output log stream) the bucket holder level objects of the BOM tree.

Parameters

<i>std::ostream&</i>	Output stream in which the BOM tree should be logged/dumped.
<i>const</i> FlightDate &	Root of the BOM tree to be displayed.

34.14.2.15 static void stdair::BomDisplay::csvBookingClassDisplay (std::ostream & , const **BookingClass** & , const std::string & *iLeadingString*) [static]

Display (dump in the underlying output log stream) the segment-class, without going recursively deeper in the BOM tree.

Parameters

<i>std::ostream&</i>	Output stream in which the BOM tree should be logged/dumped.
<i>const</i> BookingClass &	Root of the BOM tree to be displayed.
<i>const</i> std::string&	Leading string to be displayed.

34.14.2.16 static void stdair::BomDisplay::csvBookingClassDisplay (std::ostream & , const **FlightDate** &) [static]

Recursively display (dump in the underlying output log stream) the segment-class level objects of the BOM tree.

Parameters

<i>std::ostream&</i>	Output stream in which the BOM tree should be logged/dumped.
<i>const</i> FlightDate &	Root of the BOM tree to be displayed.

34.14.2.17 **static void stdair::BomDisplay::csvDisplay (std::ostream & , const TravelSolutionList_T &) [static]**

Display (dump in the underlying output log stream) the full list of travel solution structures.

Parameters

<i>std::ostream&</i>	Output stream in which the list of travel solutions is logged/dumped.
<i>TravelSolutionList_T&</i>	List of travel solutions to display.

34.14.2.18 **static void stdair::BomDisplay::csvDisplay (std::ostream & , const DatePeriodList_T &) [static]**

Display (dump in the underlying output log stream) the full list of date period fare rule sub bom tree.

Parameters

<i>std::ostream&</i>	Output stream in which the list of travel solutions is logged/dumped.
<i>DatePeriodList_T&</i>	List of date period to display.

34.14.2.19 **static void stdair::BomDisplay::csvSimFQTAirRACDisplay (std::ostream & , const BomRoot &) [static]**

Recursively display (dump in the underlying output log stream) the objects of the BOM tree.

Parameters

<i>std::ostream&</i>	Output stream in which the BOM tree should be logged/dumped.
<i>const BomRoot&</i>	Root of the BOM tree to be displayed.

34.14.2.20 **static void stdair::BomDisplay::csvAirportPairDisplay (std::ostream & , const AirportPair &) [static]**

Recursively display (dump in the underlying output log stream) the objects of the BOM tree from the level of the given airport pair.

Parameters

<i>std::ostream&</i>	Output stream in which the BOM tree should be logged/dumped.
<i>const AirportPair&</i>	Root of the BOM tree to be displayed.

34.14.2.21 `static void stdair::BomDisplay::csvDateDisplay (std::ostream & , const DatePeriod &) [static]`

Recursively display (dump in the underlying output log stream) the objects of the BOM tree from the level of the given date range.

Parameters

<i>std::ostream&</i>	Output stream in which the BOM tree should be logged/dumped.
<i>const</i> DatePeriod &	Root of the BOM tree to be displayed.

34.14.2.22 `static void stdair::BomDisplay::csvPosChannelDisplay (std::ostream & , const PosChannel &) [static]`

Recursively display (dump in the underlying output log stream) the objects of the BOM tree from the level of the given point of sale channel.

Parameters

<i>std::ostream&</i>	Output stream in which the BOM tree should be logged/dumped.
<i>const</i> PosChannel &	Root of the BOM tree to be displayed.

34.14.2.23 `static void stdair::BomDisplay::csvTimeDisplay (std::ostream & , const TimePeriod &) [static]`

Recursively display (dump in the underlying output log stream) the objects of the BOM tree from the level of the given time range.

Parameters

<i>std::ostream&</i>	Output stream in which the BOM tree should be logged/dumped.
<i>const</i> TimePeriod &	Root of the BOM tree to be displayed.

34.14.2.24 `template<typename FEATURE_TYPE > static void stdair::BomDisplay::csvFeatureListDisplay (std::ostream & oStream, const TimePeriod &) [static]`

Recursively display (dump in the underlying output log stream) the list of fare/yield features objects of the BOM tree.

Parameters

<i>std::ostream&</i>	Output stream in which the BOM tree should be logged/dumped.
<i>const</i> TimePeriod &	Root of the BOM tree to be displayed.

```
34.14.2.25 template<typename FEATURE_TYPE > static void
stdair::BomDisplay::csvFeaturesDisplay ( std::ostream & oStream, const
FEATURE_TYPE & ) [static]
```

Recursively display (dump in the underlying output log stream) the fare/yield features objects of the BOM tree.

Parameters

<i>std::ostream&</i>	Output stream in which the BOM tree should be logged/dumped.
<i>const</i> FEATURE_TYPE&	Root of the BOM tree to be displayed.

```
34.14.2.26 static void stdair::BomDisplay::csvAirlineClassDisplay ( std::ostream & , const
AirlineClassList & ) [static]
```

Recursively display (dump in the underlying output log stream) the airline class objects of the BOM tree.

Parameters

<i>std::ostream&</i>	Output stream in which the BOM tree should be logged/dumped.
<i>const</i> AirlineClassList &	Root of the BOM tree to be displayed.

The documentation for this class was generated from the following file:

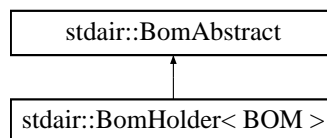
- [stdair/bom/BomDisplay.hpp](#)

34.15 stdair::BomHolder< BOM > Class Template Reference

Class representing the holder of BOM object containers (list and map).

```
#include <stdair/bom/BomHolder.hpp>
```

Inheritance diagram for stdair::BomHolder< BOM >:



Public Types

- typedef [stdair::BomHolderKey](#) Key_T
- typedef std::list< BOM * > [BomList_T](#)
- typedef std::map< const [MapKey_T](#), BOM * > [BomMap_T](#)

Public Member Functions

- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- std::string [toString](#) () const
- const std::string [describeKey](#) () const

Public Attributes

- [Key_T _key](#)
- [BomList_T _bomList](#)
- [BomMap_T _bomMap](#)

Protected Member Functions

- [BomHolder](#) ()
- [BomHolder](#) (const [BomHolder](#) &)
- [BomHolder](#) (const [Key_T](#) &iKey)
- [~BomHolder](#) ()

Friends

- class [FacBom](#)
- class [FacBomManager](#)

34.15.1 Detailed Description

template<typename BOM>class stdair::BomHolder< BOM >

Class representing the holder of BOM object containers (list and map).

Definition at line 24 of file [BomHolder.hpp](#).

34.15.2 Member Typedef Documentation

34.15.2.1 template<typename BOM> typedef stdair::BomHolderKey
stdair::BomHolder< BOM >::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 34 of file [BomHolder.hpp](#).

34.15.2.2 template<typename BOM> typedef std::list<BOM*> stdair::BomHolder< BOM
>::BomList_T

(STL) list of children.

Definition at line 39 of file [BomHolder.hpp](#).

34.15.2.3 `template<typename BOM> typedef std::map<const MapKey_T, BOM*>
stdair::BomHolder< BOM >::BomMap_T`

(STL) map of children.

Definition at line 44 of file [BomHolder.hpp](#).

34.15.3 Constructor & Destructor Documentation

34.15.3.1 `template<typename BOM> stdair::BomHolder< BOM >::BomHolder ()
[protected]`

Constructor.

34.15.3.2 `template<typename BOM> stdair::BomHolder< BOM >::BomHolder (const
BomHolder< BOM > &) [protected]`

Copy constructor.

34.15.3.3 `template<typename BOM> stdair::BomHolder< BOM >::BomHolder (const
Key_T & iKey) [inline, protected]`

Main constructor.

Definition at line 94 of file [BomHolder.hpp](#).

34.15.3.4 `template<typename BOM> stdair::BomHolder< BOM >::~BomHolder ()
[inline, protected]`

Destructor.

Definition at line 99 of file [BomHolder.hpp](#).

34.15.4 Member Function Documentation

34.15.4.1 `template<typename BOM> void stdair::BomHolder< BOM >::toStream (
std::ostream & ioOut) const [inline, virtual]`

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Implements [stdair::BomAbstract](#).

Definition at line 54 of file [BomHolder.hpp](#).

References [stdair::BomHolder< BOM >::toString\(\)](#).

34.15.4.2 `template<typename BOM> void stdair::BomHolder< BOM >::fromStream (std::istream & ioIn) [inline, virtual]`

Read a Business Object from an input stream.

Parameters

<i>istream&</i> the input stream.

Implements [stdair::BomAbstract](#).

Definition at line 63 of file [BomHolder.hpp](#).

34.15.4.3 `template<typename BOM> std::string stdair::BomHolder< BOM >::toString () const [inline, virtual]`

Get the serialised version of the Business Object.

Implements [stdair::BomAbstract](#).

Definition at line 69 of file [BomHolder.hpp](#).

Referenced by [stdair::BomHolder< BOM >::toStream\(\)](#).

34.15.4.4 `template<typename BOM> const std::string stdair::BomHolder< BOM >::describeKey () const [inline]`

Get a string describing the key.

Definition at line 76 of file [BomHolder.hpp](#).

34.15.5 Friends And Related Function Documentation

34.15.5.1 `template<typename BOM> friend class FacBom [friend]`

Friend classes.

Definition at line 26 of file [BomHolder.hpp](#).

34.15.5.2 `template<typename BOM> friend class FacBomManager [friend]`

Definition at line 27 of file [BomHolder.hpp](#).

34.15.6 Member Data Documentation

34.15.6.1 `template<typename BOM> Key_T stdair::BomHolder< BOM >::_key`

Key.

Definition at line 99 of file [BomHolder.hpp](#).

34.15.6.2 `template<typename BOM> BomList_T stdair::BomHolder< BOM >::_bomList`

(STL) list of children.

Definition at line 111 of file [BomHolder.hpp](#).

Referenced by [stdair::FacBomManager::cloneHolder\(\)](#), [stdair::BomManager::getList\(\)](#), [stdair::BomManager::hasList\(\)](#), and [stdair::serialiseHelper\(\)](#).

34.15.6.3 `template<typename BOM> BomMap_T stdair::BomHolder< BOM >::_bomMap`

(STL) map of children.

Definition at line 116 of file [BomHolder.hpp](#).

Referenced by [stdair::FacBomManager::cloneHolder\(\)](#), [stdair::BomManager::getMap\(\)](#), [stdair::BomManager::getObjectPtr\(\)](#), [stdair::BomManager::hasMap\(\)](#), and [stdair::serialiseHelper\(\)](#).

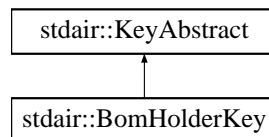
The documentation for this class was generated from the following file:

- [stdair/bom/BomHolder.hpp](#)

34.16 stdair::BomHolderKey Struct Reference

```
#include <stdair/bom/BomHolderKey.hpp>
```

Inheritance diagram for `stdair::BomHolderKey`:



Public Member Functions

- [BomHolderKey](#) ()
- [~BomHolderKey](#) ()
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- const std::string [toString](#) () const
- const std::string [describe](#) () const

34.16.1 Detailed Description

Key of the BOM structure holder.

Definition at line 12 of file [BomHolderKey.hpp](#).

34.16.2 Constructor & Destructor Documentation

34.16.2.1 stdair::BomHolderKey::BomHolderKey ()

Constructor.

Definition at line 13 of file [BomHolderKey.cpp](#).

34.16.2.2 stdair::BomHolderKey::~~BomHolderKey ()

Destructor.

Definition at line 17 of file [BomHolderKey.cpp](#).

34.16.3 Member Function Documentation

34.16.3.1 void stdair::BomHolderKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 21 of file [BomHolderKey.cpp](#).

References [toString\(\)](#).

34.16.3.2 void stdair::BomHolderKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters

<i>istream&</i> the input stream.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 26 of file [BomHolderKey.cpp](#).

34.16.3.3 const std::string stdair::BomHolderKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 30 of file [BomHolderKey.cpp](#).

Referenced by [toStream\(\)](#).

34.16.3.4 `const std::string stdair::BomHolderKey::describe () const`

Display of the key.

The documentation for this struct was generated from the following files:

- [stdair/bom/BomHolderKey.hpp](#)
- [stdair/bom/BomHolderKey.cpp](#)

34.17 stdair::BomJSONExport Class Reference

Utility class to export StdAir objects in a JSON format.

```
#include <stdair/bom/BomJSONExport.hpp>
```

Static Public Member Functions

- static void [jsonExport](#) (std::ostream &, const [FlightDate](#) &)

34.17.1 Detailed Description

Utility class to export StdAir objects in a JSON format.

Definition at line 18 of file [BomJSONExport.hpp](#).

34.17.2 Member Function Documentation

34.17.2.1 `void stdair::BomJSONExport::jsonExport (std::ostream & oStream, const FlightDate & iFlightDate) [static]`

Recursively export (dump in the underlying output log stream) the objects of the BOM tree from the level of the given [FlightDate](#).

Parameters

<i>std::ostream&</i>	Output stream in which the BOM tree should be logged/dumped.
<i>const FlightDate&</i>	Root of the BOM tree to be exported.

Definition at line 449 of file [BomJSONExport.cpp](#).

References [stdair::FlightDate::getAirlineCode\(\)](#), [stdair::FlightDate::getDepartureDate\(\)](#), [stdair::FlightDate::getFlightNumber\(\)](#), and [stdair::jsonLegDateExport\(\)](#).

The documentation for this class was generated from the following files:

- [stdair/bom/BomJSONExport.hpp](#)
- [stdair/bom/BomJSONExport.cpp](#)

34.18 stdair::BomJSONImport Class Reference

Utility class to import StdAir objects in a JSON format.

```
#include <stdair/bom/BomJSONImport.hpp>
```

Static Public Member Functions

- static bool [jsonImportInventoryKey](#) (const std::string &iBomKey, [AirlineCode_T](#) &)
- static bool [jsonImportFlightDateKey](#) (const std::string &iBomKey, [FlightNumber_T](#) &, [Date_T](#) &ioDepartureDate)

34.18.1 Detailed Description

Utility class to import StdAir objects in a JSON format.

Definition at line 18 of file [BomJSONImport.hpp](#).

34.18.2 Member Function Documentation

34.18.2.1 bool stdair::BomJSONImport::jsonImportInventoryKey (const std::string & *iBomKey*, [AirlineCode_T](#) & *ioAirlineCode*) [static]

Extract the airline code from a given JSON-formatted string.

Parameters

<i>std::string&</i>	JSON-formatted string.
<i>AirlineCode_T</i> &	Airline code extracted from the given string.

Returns

bool State whether the extracting has been successful.

Definition at line 26 of file [BomJSONImport.cpp](#).

34.18.2.2 bool stdair::BomJSONImport::jsonImportFlightDateKey (const std::string & *iBomKey*, [FlightNumber_T](#) & *ioFlightNumber*, [Date_T](#) & *ioDepartureDate*) [static]

Build a [FlightDateKey](#) from a given JSON-formatted string.

Parameters

<i>std::string&</i>	JSON-formatted string.
<i>FlightNumber_T</i> &	Flight number extracted from the given string.
<i>Date_T</i> &	Departure date extracted from the given string.

Returns

bool State whether the extracting has been successful.

Definition at line 56 of file [BomJSONImport.cpp](#).

The documentation for this class was generated from the following files:

- [stdair/bom/BomJSONImport.hpp](#)
- [stdair/bom/BomJSONImport.cpp](#)

34.19 stdair::BomKeyManager Class Reference

Utility class to extract key structures from strings.

```
#include <stdair/bom/BomKeyManager.hpp>
```

Static Public Member Functions

- static [ParsedKey](#) [extractKeys](#) (const std::string &iFullKeyStr)
- static [InventoryKey](#) [extractInventoryKey](#) (const std::string &iFullKeyStr)
- static [FlightDateKey](#) [extractFlightDateKey](#) (const std::string &iFullKeyStr)
- static [SegmentDateKey](#) [extractSegmentDateKey](#) (const std::string &iFullKeyStr)

34.19.1 Detailed Description

Utility class to extract key structures from strings.

Definition at line 29 of file [BomKeyManager.hpp](#).

34.19.2 Member Function Documentation**34.19.2.1 [ParsedKey](#) stdair::BomKeyManager::extractKeys (const std::string & iFullKeyStr)**
[static]

Build a [ParsedKey](#) structure from a full key string which includes an inventory key, flight-date key elements, segment-date key elements.

Definition at line 30 of file [BomKeyManager.cpp](#).

References [stdair::ParsedKey::_airlineCode](#), [stdair::ParsedKey::_boardingPoint](#), [stdair::ParsedKey::_boardingTime](#), [stdair::ParsedKey::_departureDate](#), [stdair::ParsedKey::_flightNumber](#), [stdair::ParsedKey::_fullKey](#), [stdair::ParsedKey::_offPoint](#), and [stdair::DEFAULT_KEY_TOKEN_DELIMITER](#).

Referenced by [stdair::TravelSolutionStruct::describe\(\)](#), [stdair::TravelSolutionStruct::display\(\)](#), [extractFlightDateKey\(\)](#), [extractInventoryKey\(\)](#), [extractSegmentDateKey\(\)](#), and [stdair::BomRetriever::retrieveSegmentData](#).

34.19.2.2 InventoryKey stdair::BomKeyManager::extractInventoryKey (const std::string & iFullKeyStr) [static]

Build a [InventoryKey](#) structure from a (full) key string.

The full key string gathers airline code, flight number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

Parameters

<code>const std::string&</code> The full key string.
--

Returns

[InventoryKey](#) The just built [InventoryKey](#) structure.

Definition at line 78 of file [BomKeyManager.cpp](#).

References [extractKeys\(\)](#), and [stdair::ParsedKey::getInventoryKey\(\)](#).

Referenced by [stdair::BomRetriever::retrieveInventoryFromLongKey\(\)](#).

34.19.2.3 FlightDateKey stdair::BomKeyManager::extractFlightDateKey (const std::string & iFullKeyStr) [static]

Build a [FlightDateKey](#) structure from a (full) key string.

The full key string gathers airline code, flight number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

Parameters

<code>const std::string&</code> The full key string.
--

Returns

[FlightDateKey](#) The just built [FlightDateKey](#) structure.

Definition at line 86 of file [BomKeyManager.cpp](#).

References [extractKeys\(\)](#), and [stdair::ParsedKey::getFlightDateKey\(\)](#).

Referenced by [stdair::OnDDateKey::getDate\(\)](#), and [stdair::BomRetriever::retrieveFlightDateFromLongKey\(\)](#).

34.19.2.4 SegmentDateKey stdair::BomKeyManager::extractSegmentDateKey (const std::string & iFullKeyStr) [static]

Build a [SegmentDateKey](#) structure from a (full) key string.

The full key string gathers airline code, flight number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

Parameters

<code>const</code> std::string& The full key string.
--

Returns

[SegmentDateKey](#) The just built [SegmentDateKey](#) structure.

Definition at line 94 of file [BomKeyManager.cpp](#).

References [extractKeys\(\)](#), and [stdair::ParsedKey::getSegmentKey\(\)](#).

Referenced by [stdair::OnDDateKey::getDestination\(\)](#), [stdair::OnDDateKey::getOrigin\(\)](#), and [stdair::BomRetriever::retrieveSegmentDateFromLongKey\(\)](#).

The documentation for this class was generated from the following files:

- [stdair/bom/BomKeyManager.hpp](#)
- [stdair/bom/BomKeyManager.cpp](#)

34.20 stdair::BomManager Class Reference

Utility class for StdAir-based objects.

```
#include <stdair/bom/BomManager.hpp>
```

Public Member Functions

- `template<>`
bool [hasList](#) (const [SegmentDate](#) &ioSegmentDate)
- `template<>`
const [BomHolder](#)< [SegmentDate](#) >::BomList_T & [getList](#) (const [SegmentDate](#) &ioSegmentDate)
- `template<>`
bool [hasMap](#) (const [SegmentDate](#) &ioSegmentDate)

Static Public Member Functions

- `template<typename OBJECT2 , typename OBJECT1 >`
static const [BomHolder](#)< OBJECT2 >::BomList_T & [getList](#) (const OBJECT1 &)
- `template<typename OBJECT2 , typename OBJECT1 >`
static const [BomHolder](#)< OBJECT2 >::BomMap_T & [getMap](#) (const OBJECT1 &)
- `template<typename OBJECT2 , typename OBJECT1 >`
static bool [hasList](#) (const OBJECT1 &)
- `template<typename OBJECT2 , typename OBJECT1 >`
static bool [hasMap](#) (const OBJECT1 &)
- `template<typename PARENT , typename CHILD >`
static PARENT * [getParentPtr](#) (const CHILD &)

- `template<typename PARENT , typename CHILD >`
`static PARENT & getParent (const CHILD &)`
- `template<typename OBJECT2 , typename OBJECT1 >`
`static OBJECT2 * getObjectPtr (const OBJECT1 &, const MapKey_T &)`
- `template<typename OBJECT2 , typename OBJECT1 >`
`static OBJECT2 & getObject (const OBJECT1 &, const MapKey_T &)`

Friends

- class `FacBomManager`

34.20.1 Detailed Description

Utility class for StdAir-based objects.

Most of those methods work for objects specified and instantiated outside StdAir, as long as those objects inherit from StdAir objects.

Definition at line 32 of file `BomManager.hpp`.

34.20.2 Member Function Documentation

34.20.2.1 `template<typename OBJECT2 , typename OBJECT1 > const BomHolder< OBJECT2 >::BomList_T & stdair::BomManager::getList (const OBJECT1 & iObject1)`
`[static]`

Get the container (STL list) of OBJECT2 objects within the OBJECT1 object.

Definition at line 127 of file `BomManager.hpp`.

References `stdair::BomHolder< BOM >::_bomList`.

34.20.2.2 `template<typename OBJECT2 , typename OBJECT1 > const BomHolder< OBJECT2 >::BomMap_T & stdair::BomManager::getMap (const OBJECT1 & iObject1)`
`[static]`

Get the container (STL map) of OBJECT2 objects within the OBJECT1 object.

Definition at line 138 of file `BomManager.hpp`.

References `stdair::BomHolder< BOM >::_bomMap`.

34.20.2.3 `template<typename OBJECT2 , typename OBJECT1 > bool`
`stdair::BomManager::hasList (const OBJECT1 & iObject1)` `[static]`

Check if the list of object2 has been initialised.

Definition at line 157 of file `BomManager.hpp`.

References `stdair::BomHolder< BOM >::_bomList`.

34.20.2.4 `template<typename OBJECT2 , typename OBJECT1 > bool
stdair::BomManager::hasMap (const OBJECT1 & iObject1) [static]`

Check if the map of object2 has been initialised.

Definition at line 177 of file [BomManager.hpp](#).

References [stdair::BomHolder< BOM >::_bomMap](#).

34.20.2.5 `template<typename PARENT , typename CHILD > PARENT *
stdair::BomManager::getParentPtr (const CHILD & iChild) [static]`

Get the PARENT of the given CHILD.

If the types do not match, NULL is returned.

Definition at line 196 of file [BomManager.hpp](#).

34.20.2.6 `template<typename PARENT , typename CHILD > PARENT &
stdair::BomManager::getParent (const CHILD & iChild) [static]`

Get the PARENT of the given CHILD.

Definition at line 206 of file [BomManager.hpp](#).

34.20.2.7 `template<typename OBJECT2 , typename OBJECT1 > OBJECT2 *
stdair::BomManager::getObjectPtr (const OBJECT1 & iObject1, const MapKey_T &
iKey) [static]`

Get the OBJECT2 pointer corresponding to the given string key.

If such a OBJECT2 does not exist, return NULL.

Definition at line 217 of file [BomManager.hpp](#).

References [stdair::BomHolder< BOM >::_bomMap](#).

34.20.2.8 `template<typename OBJECT2 , typename OBJECT1 > OBJECT2 &
stdair::BomManager::getObject (const OBJECT1 & iObject1, const MapKey_T &
iKey) [static]`

Get the OBJECT2 corresponding the the given string key.

Definition at line 259 of file [BomManager.hpp](#).

References [STDAIR_LOG_ERROR](#).

34.20.2.9 `template<> bool stdair::BomManager::hasList (const SegmentDate &
ioSegmentDate) [inline]`

Definition at line 304 of file [BomManager.hpp](#).

34.20.2.10 `template<> const BomHolder<SegmentDate>::BomList_T& stdair::BomManager::getList (const SegmentDate & ioSegmentDate) [inline]`

Definition at line 319 of file [BomManager.hpp](#).

34.20.2.11 `template<> bool stdair::BomManager::hasMap (const SegmentDate & ioSegmentDate) [inline]`

Definition at line 332 of file [BomManager.hpp](#).

34.20.3 Friends And Related Function Documentation

34.20.3.1 `friend class FacBomManager [friend]`

Definition at line 33 of file [BomManager.hpp](#).

The documentation for this class was generated from the following file:

- [stdair/bom/BomManager.hpp](#)

34.21 stdair::BomRetriever Class Reference

Utility class to retrieve StdAir objects.

```
#include <stdair/bom/BomRetriever.hpp>
```

Static Public Member Functions

- static [Inventory](#) * [retrieveInventoryFromLongKey](#) (const [BomRoot](#) &, const std::string &iFullKeyStr)
- static [Inventory](#) * [retrieveInventoryFromKey](#) (const [BomRoot](#) &, const [InventoryKey](#) &)
- static [Inventory](#) * [retrieveInventoryFromKey](#) (const [BomRoot](#) &, const [AirlineCode_T](#) &)
- static [FlightDate](#) * [retrieveFlightDateFromLongKey](#) (const [BomRoot](#) &, const std::string &iFullKeyStr)
- static [FlightDate](#) * [retrieveFlightDateFromKeySet](#) (const [BomRoot](#) &, const [AirlineCode_T](#) &, const [FlightNumber_T](#) &, const [Date_T](#) &iFlightDateDate)
- static [FlightDate](#) * [retrieveFlightDateFromLongKey](#) (const [Inventory](#) &, const std::string &iFullKeyStr)
- static [FlightDate](#) * [retrieveFlightDateFromKey](#) (const [Inventory](#) &, const [FlightDateKey](#) &)
- static [FlightDate](#) * [retrieveFlightDateFromKey](#) (const [Inventory](#) &, const [FlightNumber_T](#) &, const [Date_T](#) &iFlightDateDate)
- static [SegmentDate](#) * [retrieveSegmentDateFromLongKey](#) (const [BomRoot](#) &, const std::string &iFullKeyStr)

- static [SegmentDate](#) * [retrieveSegmentDateFromLongKey](#) (const [Inventory](#) &, const std::string &iFullKeyStr)
- static [SegmentDate](#) * [retrieveSegmentDateFromLongKey](#) (const [FlightDate](#) &, const std::string &iFullKeyStr)
- static [SegmentDate](#) * [retrieveSegmentDateFromKey](#) (const [FlightDate](#) &, const [SegmentDateKey](#) &)
- static [SegmentDate](#) * [retrieveSegmentDateFromKey](#) (const [FlightDate](#) &, const [AirportCode_T](#) &iOrigin, const [AirportCode_T](#) &iDestination)
- static [BookingClass](#) * [retrieveBookingClassFromLongKey](#) (const [Inventory](#) &, const std::string &iFullKeyStr, const [ClassCode_T](#) &)
- static [AirportPair](#) * [retrieveAirportPairFromKeySet](#) (const [BomRoot](#) &, const [stdair::AirportCode_T](#) &, const [stdair::AirportCode_T](#) &)
- static void [retrieveDatePeriodListFromKey](#) (const [AirportPair](#) &, const [stdair::Date_T](#) &, [stdair::DatePeriodList_T](#) &)
- static void [retrieveDatePeriodListFromKeySet](#) (const [BomRoot](#) &, const [stdair::AirportCode_T](#) &, const [stdair::AirportCode_T](#) &, const [stdair::Date_T](#) &, [stdair::DatePeriodList_T](#) &)
- static [stdair::LegCabin](#) & [retrieveDummyLegCabin](#) ([stdair::BomRoot](#) &)
- static [stdair::SegmentCabin](#) & [retrieveDummySegmentCabin](#) ([stdair::BomRoot](#) &)

34.21.1 Detailed Description

Utility class to retrieve StdAir objects.

Definition at line 35 of file [BomRetriever.hpp](#).

34.21.2 Member Function Documentation

34.21.2.1 [Inventory](#) * [stdair::BomRetriever::retrieveInventoryFromLongKey](#) (const [BomRoot](#) & iBomRoot, const std::string & iFullKeyStr) [static]

Retrieve an [Inventory](#) object from a (full) key string.

The full key string gathers airline code, flight number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

Parameters

<i>const</i>	BomRoot &	The root of the BOM tree.
<i>const</i>	std::string&	The full key string.

Returns

[Inventory](#)* The just retrieved [Inventory](#) object.

Definition at line 29 of file [BomRetriever.cpp](#).

References [stdair::BomKeyManager::extractInventoryKey\(\)](#), and [stdair::BomRoot::getInventory\(\)](#).

Referenced by [retrieveFlightDateFromLongKey\(\)](#).

34.21.2.2 **Inventory** * stdair::BomRetriever::retrieveInventoryFromKey (const BomRoot & iBomRoot, const InventoryKey & iKey) [static]

Retrieve an [Inventory](#) object from an [InventoryKey](#) structure.

Parameters

const BomRoot &	The root of the BOM tree.
const InventoryKey &	The key.

Returns

Inventory* The just retrieved [Inventory](#) object.

Definition at line 43 of file [BomRetriever.cpp](#).

References [stdair::BomRoot::getInventory\(\)](#).

Referenced by [retrieveDummyLegCabin\(\)](#), [retrieveDummySegmentCabin\(\)](#), and [retrieveFlightDateFromKeySet\(\)](#).

34.21.2.3 **Inventory** * stdair::BomRetriever::retrieveInventoryFromKey (const BomRoot & iBomRoot, const AirlineCode_T & iAirlineCode) [static]

Retrieve an [Inventory](#) object from an [InventoryKey](#) structure.

Parameters

const BomRoot &	The root of the BOM tree.
const AirlineCode_T&	The key.

Returns

Inventory* The just retrieved [Inventory](#) object.

Definition at line 55 of file [BomRetriever.cpp](#).

References [stdair::BomRoot::getInventory\(\)](#).

34.21.2.4 **FlightDate** * stdair::BomRetriever::retrieveFlightDateFromLongKey (const BomRoot & iBomRoot, const std::string & iFullKeyStr) [static]

Retrieve a [FlightDate](#) object from a (full) key string.

The full key string gathers airline code, flight number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

Parameters

const BomRoot &	The root of the BOM tree.
const std::string&	The full key string.

Returns

FlightDate* The just retrieved [FlightDate](#) object.

Definition at line 68 of file [BomRetriever.cpp](#).

References [stdair::BomKeyManager::extractFlightDateKey\(\)](#), [stdair::Inventory::getFlightDate\(\)](#), and [retrieveInventoryFromLongKey\(\)](#).

Referenced by [retrieveSegmentDateFromLongKey\(\)](#).

```
34.21.2.5 FlightDate * stdair::BomRetriever::retrieveFlightDateFromKeySet ( const
    BomRoot & iBomRoot, const AirlineCode_T & iAirlineCode, const
    FlightNumber_T & iFlightNumber, const Date_T & iFlightDateDate )
    [static]
```

Retrieve a [FlightDate](#) object from a set of keys.

Parameters

<i>const</i>	BomRoot & The root of the BOM tree.
<i>const</i>	AirlineCode_T & The key.
<i>const</i>	FlightNumber_T & Part of the key.
<i>const</i>	Date_T & Part of the key.

Returns

FlightDate* The just retrieved [FlightDate](#) object.

Definition at line 91 of file [BomRetriever.cpp](#).

References [retrieveFlightDateFromKey\(\)](#), and [retrieveInventoryFromKey\(\)](#).

Referenced by [stdair::STDAIR_Service::check\(\)](#), [stdair::STDAIR_Service::csvDisplay\(\)](#), and [stdair::STDAIR_Service::jsonExport\(\)](#).

```
34.21.2.6 FlightDate * stdair::BomRetriever::retrieveFlightDateFromLongKey ( const
    Inventory & iInventory, const std::string & iFullKeyStr ) [static]
```

Retrieve a [FlightDate](#) object from a (full) key string.

The full key string gathers airline code, flight number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

Parameters

<i>const</i>	Inventory & The root of the BOM tree.
<i>const</i>	std::string& The full key string.

Returns

FlightDate* The just retrieved [FlightDate](#) object.

Definition at line 114 of file [BomRetriever.cpp](#).

References [stdair::BomKeyManager::extractFlightDateKey\(\)](#), and [stdair::Inventory::getFlightDate\(\)](#).

34.21.2.7 `FlightDate * stdair::BomRetriever::retrieveFlightDateFromKey (const Inventory & inventory, const FlightDateKey & iKey) [static]`

Retrieve a [FlightDate](#) object from an [FlightDateKey](#) structure.

Parameters

<code>const</code>	Inventory & The root of the BOM tree.
<code>const</code>	FlightDateKey & The key.

Returns

`FlightDate*` The just retrieved [FlightDate](#) object.

Definition at line 129 of file [BomRetriever.cpp](#).

References [stdair::Inventory::getFlightDate\(\)](#).

Referenced by [retrieveDummyLegCabin\(\)](#), [retrieveDummySegmentCabin\(\)](#), [retrieveFlightDateFromKeySet\(\)](#), and [retrieveSegmentDateFromLongKey\(\)](#).

34.21.2.8 `FlightDate * stdair::BomRetriever::retrieveFlightDateFromKey (const Inventory & inventory, const FlightNumber_T & iFlightNumber, const Date_T & iFlightDateDate) [static]`

Retrieve a [FlightDate](#) object from an [FlightDateKey](#) structure.

Parameters

<code>const</code>	Inventory & The root of the BOM tree.
<code>const</code>	<code>FlightNumber_T</code> & Part of the key.
<code>const</code>	<code>Date_T</code> & Part of the key.

Returns

`FlightDate*` The just retrieved [FlightDate](#) object.

Definition at line 141 of file [BomRetriever.cpp](#).

References [stdair::Inventory::getFlightDate\(\)](#).

34.21.2.9 `SegmentDate * stdair::BomRetriever::retrieveSegmentDateFromLongKey (const BomRoot & iBomRoot, const std::string & iFullKeyStr) [static]`

Retrieve a [SegmentDate](#) object from a (full) key string.

The full key string gathers airline code, segment number, origin and destination, cabin and booking class. It corresponds to the output generated by the `toString()` methods of the `XxxKey` structures.

Parameters

<code>const</code>	BomRoot & The root of the BOM tree.
<code>const</code>	<code>std::string</code> & The full key string.

Returns

SegmentDate* The just retrieved [SegmentDate](#) object.

Definition at line 155 of file [BomRetriever.cpp](#).

References [stdair::BomKeyManager::extractSegmentDateKey\(\)](#), [stdair::FlightDate::getSegmentDate\(\)](#), and [retrieveFlightDateFromLongKey\(\)](#).

Referenced by [retrieveBookingClassFromLongKey\(\)](#).

34.21.2.10 [SegmentDate](#) * [stdair::BomRetriever::retrieveSegmentDateFromLongKey](#) ([const Inventory](#) & *iInventory*, [const](#) [std::string](#) & *iFullKeyStr*) [\[static\]](#)

Retrieve a [SegmentDate](#) object from a (full) key string.

The full key string gathers airline code, segment number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

Parameters

const	Inventory & The root of the BOM tree.
const	std::string & The full key string.

Returns

SegmentDate* The just retrieved [SegmentDate](#) object.

Definition at line 178 of file [BomRetriever.cpp](#).

References [stdair::ParsedKey::_airlineCode](#), [stdair::BomKeyManager::extractKeys\(\)](#), [stdair::Inventory::getAirlineCode\(\)](#), [stdair::ParsedKey::getFlightDateKey\(\)](#), [stdair::ParsedKey::getSegmentKey\(\)](#), [retrieveFlightDateFromKey\(\)](#), [retrieveSegmentDateFromKey\(\)](#), [STDAIR_LOG_DEBUG](#), [stdair::SegmentDateKey::toString\(\)](#), and [stdair::FlightDateKey::toString\(\)](#).

34.21.2.11 [SegmentDate](#) * [stdair::BomRetriever::retrieveSegmentDateFromLongKey](#) ([const FlightDate](#) & *iFlightDate*, [const](#) [std::string](#) & *iFullKeyStr*) [\[static\]](#)

Retrieve a [SegmentDate](#) object from a (full) key string.

The full key string gathers airline code, segment number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

Parameters

const	FlightDate & The root of the BOM tree.
const	std::string & The full key string.

Returns

SegmentDate* The just retrieved [SegmentDate](#) object.

Definition at line 210 of file [BomRetriever.cpp](#).

References [stdair::BomKeyManager::extractSegmentDateKey\(\)](#), and [stdair::FlightDate::getSegmentDate\(\)](#).

34.21.2.12 `SegmentDate * stdair::BomRetriever::retrieveSegmentDateFromKey (const FlightDate & iFlightDate, const SegmentDateKey & iKey) [static]`

Retrieve a [SegmentDate](#) object from an [SegmentDateKey](#) structure.

Parameters

<code>const</code>	FlightDate & The root of the BOM tree.
<code>const</code>	SegmentDateKey & The key.

Returns

`SegmentDate*` The just retrieved [SegmentDate](#) object.

Definition at line 225 of file [BomRetriever.cpp](#).

References [stdair::FlightDate::getSegmentDate\(\)](#).

Referenced by [retrieveSegmentDateFromLongKey\(\)](#).

34.21.2.13 `SegmentDate * stdair::BomRetriever::retrieveSegmentDateFromKey (const FlightDate & iFlightDate, const AirportCode_T & iOrigin, const AirportCode_T & iDestination) [static]`

Retrieve a [SegmentDate](#) object from an [SegmentDateKey](#) structure.

Parameters

<code>const</code>	FlightDate & The root of the BOM tree.
<code>const</code>	<code>AirportCode_T</code> & Origin, part of the key.
<code>const</code>	<code>AirportCode_T</code> & Destination, part of the key.

Returns

`SegmentDate*` The just retrieved [SegmentDate](#) object.

Definition at line 237 of file [BomRetriever.cpp](#).

References [stdair::FlightDate::getSegmentDate\(\)](#).

34.21.2.14 `BookingClass * stdair::BomRetriever::retrieveBookingClassFromLongKey (const Inventory & iInventory, const std::string & iFullKeyStr, const ClassCode_T & iClassCode) [static]`

Retrieve a [BookingClass](#) object from a (full) key string.

The full key string gathers airline code, segment number, origin and destination, cabin and booking class. It corresponds to the output generated by the `toString()` methods of the `XxxKey` structures.

Besides being attached to segment-cabin objects (and fare family objects, when they exist), the booking-class objects must also be attached directly to the segment-date.

Hence, if an assertion fails within that method call, chances are that the booking-class

objects have not been attached to the segment-date objects. Check, for instance, the `CmdBomManager::buildSampleBom()` to see how that should be properly done.

Parameters

<i>const</i>	Inventory & The root of the BOM tree.
<i>const</i>	<code>std::string</code> & Part of the full key string.
<i>const</i>	<code>ClassCode_T</code> & Part of the full key string.

Returns

`BookingClass*` The just retrieved [BookingClass](#) object.

Definition at line 251 of file [BomRetriever.cpp](#).

References [retrieveSegmentDateFromLongKey\(\)](#).

34.21.2.15 `AirportPair * stdair::BomRetriever::retrieveAirportPairFromKeySet (const BomRoot & iBomRoot, const stdair::AirportCode_T & iOrigin, const stdair::AirportCode_T & iDestination) [static]`

Retrieve an [AirportPair](#) object from an [AirportPair](#) structure.

Parameters

<i>const</i>	BomRoot & The root of the BOM tree.
<i>const</i>	<code>AirportCode_T</code> & Origin, part of the key.
<i>const</i>	<code>AirportCode_T</code> & Destination, part of the key.

Returns

`AirportPair*` The just retrieved [AirportPair](#) object.

Definition at line 273 of file [BomRetriever.cpp](#).

References [stdair::AirportPairKey::toString\(\)](#).

Referenced by [retrieveDatePeriodListFromKeySet\(\)](#).

34.21.2.16 `void stdair::BomRetriever::retrieveDatePeriodListFromKey (const AirportPair & iAirportPair, const stdair::Date_T & iDepartureDate, stdair::DatePeriodList_T & ioDatePeriodList) [static]`

Retrieve a list of date-period corresponding to a flight date.

Parameters

<i>const</i>	AirportPair & The root of the BOM tree.
<i>const</i>	<code>Date_T</code> & Departure Date of the flight
<i>stdair::DatePeriodList_T</i>	List of DatePeriod to display.

Definition at line 291 of file [BomRetriever.cpp](#).

References [stdair::DatePeriod::isDepartureDateValid\(\)](#).

Referenced by [retrieveDatePeriodListFromKeySet\(\)](#).

```
34.21.2.17 void stdair::BomRetriever::retrieveDatePeriodListFromKeySet ( const
    BomRoot & iBomRoot, const stdair::AirportCode_T & iOrigin, const
    stdair::AirportCode_T & iDestination, const stdair::Date_T & iDepartureDate,
    stdair::DatePeriodList_T & ioDatePeriodList ) [static]
```

Retrieve a list of date-period from a set of keys.

Parameters

<i>const</i>	BomRoot & The root of the BOM tree.
<i>const</i>	AirportCode_T & Part of the AirportPair key: the origin airport
<i>const</i>	AirportCode_T & Part of the AirportPair key: the destination airport.
<i>const</i>	Date_T & Departure date of the flight
<i>stdair::DatePe</i> <i>T&</i>	List of DatePeriod to display.

Definition at line 322 of file [BomRetriever.cpp](#).

References [retrieveAirportPairFromKeySet\(\)](#), and [retrieveDatePeriodListFromKey\(\)](#).

Referenced by [stdair::STDAIR_Service::check\(\)](#), and [stdair::STDAIR_Service::csvDisplay\(\)](#).

```
34.21.2.18 LegCabin & stdair::BomRetriever::retrieveDummyLegCabin ( stdair::BomRoot
    & iBomRoot ) [static]
```

Retrieve the sample leg-cabin of the dummy inventory of "XX".

Parameters

<i>stdair::BomRc</i>	The BOM tree.
----------------------	---------------

Definition at line 345 of file [BomRetriever.cpp](#).

References [stdair::DEFAULT_AIRLINE_CODE](#), [stdair::DEFAULT_CABIN_CODE](#), [stdair::DEFAULT_DEPARTURE_DATE](#), [stdair::DEFAULT_FLIGHT_NUMBER](#), [stdair::DEFAULT_ORIGIN](#), [stdair::LegDate::getLegCabin\(\)](#), [stdair::FlightDate::getLegDate\(\)](#), [retrieveFlightDateFromKey\(\)](#), and [retrieveInventoryFromKey\(\)](#).

```
34.21.2.19 SegmentCabin & stdair::BomRetriever::retrieveDummySegmentCabin (
    stdair::BomRoot & iBomRoot ) [static]
```

Retrieve the sample segment-cabin of the dummy inventory of "XX".

Parameters

<i>stdair::BomRc</i>	The BOM tree.
----------------------	---------------

Definition at line 403 of file [BomRetriever.cpp](#).

References [stdair::DEFAULT_AIRLINE_CODE](#), [stdair::DEFAULT_CABIN_CODE](#), [stdair::DEFAULT_DEPARTURE_DATE](#), [stdair::DEFAULT_DESTINATION](#), [stdair::DEFAULT_FLIGHT_NUMBER](#), [stdair::DEFAULT_ORIGIN](#), [stdair::FlightDate::getSegmentDate\(\)](#), [retrieveFlightDateFromKey\(\)](#), [retrieveInventoryFromKey\(\)](#), and [stdair::SegmentCabinKey::toString\(\)](#).

The documentation for this class was generated from the following files:

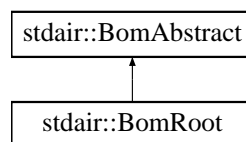
- [stdair/bom/BomRetriever.hpp](#)
- [stdair/bom/BomRetriever.cpp](#)

34.22 stdair::BomRoot Class Reference

Class representing the actual attributes for the Bom root.

```
#include <stdair/bom/BomRoot.hpp>
```

Inheritance diagram for stdair::BomRoot:



Public Types

- typedef [BomRootKey](#) [Key_T](#)

Public Member Functions

- const [Key_T](#) & [getKey](#) () const
- const [HolderMap_T](#) & [getHolderMap](#) () const
- [Inventory](#) * [getInventory](#) (const std::string &iInventoryKeyStr) const
- [Inventory](#) * [getInventory](#) (const [InventoryKey](#) &) const
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- std::string [toString](#) () const
- const std::string [describeKey](#) () const
- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int iFileVersion)

Protected Member Functions

- [BomRoot](#) ()
- [BomRoot](#) (const [BomRoot](#) &)
- [BomRoot](#) (const [Key_T](#) &iKey)
- [~BomRoot](#) ()

Protected Attributes

- [Key_T _key](#)
- [HolderMap_T _holderMap](#)

Friends

- class [FacBom](#)
- class [FacBomManager](#)
- class [boost::serialization::access](#)

34.22.1 Detailed Description

Class representing the actual attributes for the Bom root.

Definition at line 30 of file [BomRoot.hpp](#).

34.22.2 Member Typedef Documentation**34.22.2.1 typedef BomRootKey stdair::BomRoot::Key_T**

Definition allowing to retrieve the associated BOM key type.

Definition at line 39 of file [BomRoot.hpp](#).

34.22.3 Constructor & Destructor Documentation**34.22.3.1 stdair::BomRoot::BomRoot () [protected]**

Default constructor.

Definition at line 17 of file [BomRoot.cpp](#).

34.22.3.2 stdair::BomRoot::BomRoot (const BomRoot & iBomRoot) [protected]

Copy constructor.

Definition at line 22 of file [BomRoot.cpp](#).

34.22.3.3 stdair::BomRoot::BomRoot (const Key_T & iKey) [protected]

Main constructor.

Definition at line 27 of file [BomRoot.cpp](#).

34.22.3.4 stdair::BomRoot::~~BomRoot () [protected]

Destructor.

Definition at line 31 of file [BomRoot.cpp](#).

34.22.4 Member Function Documentation

34.22.4.1 const Key_T& stdair::BomRoot::getKey () const [inline]

Get the inventory key (airline code).

Definition at line 45 of file [BomRoot.hpp](#).

References [_key](#).

34.22.4.2 const HolderMap_T& stdair::BomRoot::getHolderMap () const [inline]

Get the map of children.

Definition at line 50 of file [BomRoot.hpp](#).

References [_holderMap](#).

34.22.4.3 Inventory* stdair::BomRoot::getInventory (const std::string & iInventoryKeyStr) const

Get a pointer on the [Inventory](#) object corresponding to the given key.

Note

The [Inventory](#) object can be inherited from, if needed. In that case, a dynamic_
cast<> may be needed.

Parameters

<i>const</i> std::string& The flight-date key.
--

Returns

Inventory* Found [Inventory](#) object. NULL if not found.

Definition at line 42 of file [BomRoot.cpp](#).

Referenced by [getInventory\(\)](#), [stdair::BomRetriever::retrieveInventoryFromKey\(\)](#), and [stdair::BomRetriever::retrieveInven](#)

34.22.4.4 Inventory* stdair::BomRoot::getInventory (const InventoryKey & iInventoryKey) const

Get a pointer on the [Inventory](#) object corresponding to the given key.

Note

The [Inventory](#) object can be inherited from, if needed. In that case, a dynamic_
cast<> may be needed.

Parameters

<i>const</i> InventoryKey & The flight-date key

Returns

Inventory* Found [Inventory](#) object. NULL if not found.

Definition at line 49 of file [BomRoot.cpp](#).

References [getInventory\(\)](#), and [stdair::InventoryKey::toString\(\)](#).

34.22.4.5 `void stdair::BomRoot::toStream (std::ostream & ioOut) const` `[inline, virtual]`

Dump a Business Object into an output stream.

Parameters

<code>ostream&</code> the output stream.
--

Implements [stdair::BomAbstract](#).

Definition at line 86 of file [BomRoot.hpp](#).

References [toString\(\)](#).

34.22.4.6 `void stdair::BomRoot::fromStream (std::istream & ioin)` `[inline, virtual]`

Read a Business Object from an input stream.

Parameters

<code>istream&</code> the input stream.

Implements [stdair::BomAbstract](#).

Definition at line 95 of file [BomRoot.hpp](#).

34.22.4.7 `std::string stdair::BomRoot::toString () const` `[virtual]`

Get the serialised version of the Business Object.

Implements [stdair::BomAbstract](#).

Definition at line 35 of file [BomRoot.cpp](#).

References [_key](#), and [stdair::BomRootKey::toString\(\)](#).

Referenced by [toStream\(\)](#).

34.22.4.8 `const std::string stdair::BomRoot::describeKey () const` `[inline]`

Get a string describing the key.

Definition at line 106 of file [BomRoot.hpp](#).

References [_key](#), and [stdair::BomRootKey::toString\(\)](#).

34.22.4.9 `template<class Archive > void stdair::BomRoot::serialize (Archive & ar, const unsigned int iFileVersion)`

Serialisation.

That method is used both for serialisation a BOM tree (into a backup file/stream), as well as re-instantiating a BOM tree from a back-up file/stream.

Note

The implementation of that method is to be found in the [CmdBomSerialiser](#) command.

Definition at line 133 of file [CmdBomSerialiser.cpp](#).

References [_key](#).

34.22.5 Friends And Related Function Documentation

34.22.5.1 `friend class FacBom [friend]`

Definition at line 31 of file [BomRoot.hpp](#).

34.22.5.2 `friend class FacBomManager [friend]`

Definition at line 32 of file [BomRoot.hpp](#).

34.22.5.3 `friend class boost::serialization::access [friend]`

Definition at line 33 of file [BomRoot.hpp](#).

34.22.6 Member Data Documentation

34.22.6.1 `Key_T stdair::BomRoot::_key [protected]`

Primary key.

Definition at line 166 of file [BomRoot.hpp](#).

Referenced by [describeKey\(\)](#), [getKey\(\)](#), [serialize\(\)](#), and [toString\(\)](#).

34.22.6.2 `HolderMap_T stdair::BomRoot::_holderMap [protected]`

Map holding the children ([Inventory](#) objects).

Definition at line 171 of file [BomRoot.hpp](#).

Referenced by [getHolderMap\(\)](#).

The documentation for this class was generated from the following files:

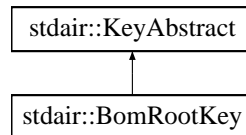
- [stdair/bom/BomRoot.hpp](#)
- [stdair/bom/BomRoot.cpp](#)
- [stdair/command/CmdBomSerialiser.cpp](#)

34.23 stdair::BomRootKey Struct Reference

Key of the BOM structure root.

```
#include <stdair/bom/BomRootKey.hpp>
```

Inheritance diagram for stdair::BomRootKey:



Public Member Functions

- [BomRootKey](#) ()
- [BomRootKey](#) (const std::string &identification)
- [BomRootKey](#) (const [BomRootKey](#) &)
- [~BomRootKey](#) ()
- const std::string & [getID](#) () const
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- const std::string [toString](#) () const
- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int iFileVersion)

Friends

- class [boost::serialization::access](#)

34.23.1 Detailed Description

Key of the BOM structure root.

Definition at line 25 of file [BomRootKey.hpp](#).

34.23.2 Constructor & Destructor Documentation

34.23.2.1 stdair::BomRootKey::BomRootKey ()

Default constructor.

Definition at line 18 of file [BomRootKey.cpp](#).

34.23.2.2 stdair::BomRootKey::BomRootKey (const std::string & *identification*)

Constructor.

Definition at line 28 of file [BomRootKey.cpp](#).

34.23.2.3 stdair::BomRootKey::BomRootKey (const BomRootKey & *iBomRootKey*)

Copy constructor.

Definition at line 23 of file [BomRootKey.cpp](#).

34.23.2.4 stdair::BomRootKey::~~BomRootKey ()

Destructor.

Definition at line 33 of file [BomRootKey.cpp](#).

34.23.3 Member Function Documentation

34.23.3.1 const std::string& stdair::BomRootKey::getID () const [inline]

Get the identification.

Definition at line 56 of file [BomRootKey.hpp](#).

34.23.3.2 void stdair::BomRootKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 37 of file [BomRootKey.cpp](#).

References [toString\(\)](#).

34.23.3.3 void stdair::BomRootKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters

<i>istream&</i> the input stream.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 42 of file [BomRootKey.cpp](#).

34.23.3.4 const std::string stdair::BomRootKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 46 of file [BomRootKey.cpp](#).

Referenced by [stdair::BomRoot::describeKey\(\)](#), [toStream\(\)](#), and [stdair::BomRoot::toString\(\)](#).

34.23.3.5 `template<class Archive > void stdair::BomRootKey::serialize (Archive & ar, const unsigned int iFileVersion)`

Serialisation.

Definition at line 68 of file [BomRootKey.cpp](#).

34.23.4 Friends And Related Function Documentation

34.23.4.1 `friend class boost::serialization::access [friend]`

Definition at line 26 of file [BomRootKey.hpp](#).

The documentation for this struct was generated from the following files:

- [stdair/bom/BomRootKey.hpp](#)
- [stdair/bom/BomRootKey.cpp](#)

34.24 stdair_test::BookingClass Struct Reference

```
#include <test/stdair/StdairTestLib.hpp>
```

Public Member Functions

- [BookingClass](#) (const std::string &iClassCode)
- std::string [toString](#) () const

Public Attributes

- std::string [_classCode](#)

34.24.1 Detailed Description

[BookingClass](#)

Definition at line 16 of file [StdairTestLib.hpp](#).

34.24.2 Constructor & Destructor Documentation

34.24.2.1 stdair_test::BookingClass::BookingClass (const std::string & *iClassCode*)
[inline]

Constructor.

Definition at line 19 of file [StdairTestLib.hpp](#).

34.24.3 Member Function Documentation

34.24.3.1 std::string stdair_test::BookingClass::toString () const [inline]

Display .

Definition at line 24 of file [StdairTestLib.hpp](#).

References [_classCode](#).

34.24.4 Member Data Documentation

34.24.4.1 std::string stdair_test::BookingClass::_classCode

Definition at line 17 of file [StdairTestLib.hpp](#).

Referenced by [stdair_test::Cabin::toString\(\)](#), and [toString\(\)](#).

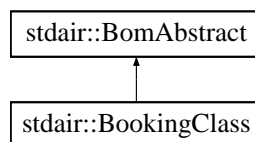
The documentation for this struct was generated from the following file:

- test/stdair/[StdairTestLib.hpp](#)

34.25 stdair::BookingClass Class Reference

```
#include <stdair/bom/BookingClass.hpp>
```

Inheritance diagram for stdair::BookingClass:



Public Types

- typedef [BookingClassKey](#) Key_T

Public Member Functions

- const [Key_T](#) & [getKey](#) () const
- const [ClassCode_T](#) & [getClassCode](#) () const
- [BomAbstract](#) *const [getParent](#) () const
- const [HolderMap_T](#) & [getHolderMap](#) () const
- const [SubclassCode_T](#) & [getSubclassCode](#) () const
- const [AuthorizationLevel_T](#) & [getAuthorizationLevel](#) () const
- const [ProtectionLevel_T](#) & [getProtection](#) () const
- const [ProtectionLevel_T](#) & [getCumulatedProtection](#) () const
- const [BookingLimit_T](#) & [getCumulatedBookingLimit](#) () const
- const [NbOfSeats_T](#) & [getNegotiatedSpace](#) () const
- const [OverbookingRate_T](#) & [getNoShowPercentage](#) () const
- const [OverbookingRate_T](#) & [getCancellationPercentage](#) () const
- const [NbOfBookings_T](#) & [getNbOfBookings](#) () const
- const [NbOfBookings_T](#) & [getNbOfGroupBookings](#) () const
- const [NbOfBookings_T](#) & [getNbOfPendingGroupBookings](#) () const
- const [NbOfBookings_T](#) & [getNbOfStaffBookings](#) () const
- const [NbOfBookings_T](#) & [getNbOfWLBookings](#) () const
- const [NbOfCancellations_T](#) & [getNbOfCancellations](#) () const
- const [NbOfBookings_T](#) & [getETB](#) () const
- const [Availability_T](#) & [getNetClassAvailability](#) () const
- const [Availability_T](#) & [getSegmentAvailability](#) () const
- const [Availability_T](#) & [getNetRevenueAvailability](#) () const
- const [Yield_T](#) & [getYield](#) () const
- const [MeanValue_T](#) & [getMean](#) () const
- const [StdDevValue_T](#) & [getStdDev](#) () const
- const [GeneratedDemandVector_T](#) & [getGeneratedDemandVector](#) () const
- void [setCumulatedProtection](#) (const [ProtectionLevel_T](#) &iPL)
- void [setProtection](#) (const [ProtectionLevel_T](#) &iPL)
- void [setCumulatedBookingLimit](#) (const [BookingLimit_T](#) &iBL)
- void [setAuthorizationLevel](#) (const [AuthorizationLevel_T](#) &iAU)
- void [setSegmentAvailability](#) (const [Availability_T](#) &iAvl)
- void [setYield](#) (const [Yield_T](#) &iYield)
- void [setMean](#) (const [MeanValue_T](#) &iMean)
- void [setStdDev](#) (const [StdDevValue_T](#) &iStdDev)
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- std::string [toString](#) () const
- const std::string [describeKey](#) () const
- void [sell](#) (const [NbOfBookings_T](#) &)
- void [cancel](#) (const [NbOfBookings_T](#) &)
- void [generateDemandSamples](#) (const int &)
- void [generateDemandSamples](#) (const int &, const [RandomSeed_T](#) &)

Protected Member Functions

- [BookingClass](#) (const [Key_T](#) &)
- virtual [~BookingClass](#) ()

Protected Attributes

- [Key_T_key](#)
- [BomAbstract](#) * [_parent](#)
- [HolderMap_T_holderMap](#)
- [SubclassCode_T_subclassCode](#)
- [ProtectionLevel_T_cumulatedProtection](#)
- [ProtectionLevel_T_protection](#)
- [BookingLimit_T_cumulatedBookingLimit](#)
- [AuthorizationLevel_T_au](#)
- [NbOfSeats_T_nego](#)
- [OverbookingRate_T_noShowPercentage](#)
- [OverbookingRate_T_cancellationPercentage](#)
- [NbOfBookings_T_nbOfBookings](#)
- [NbOfBookings_T_groupNbOfBookings](#)
- [NbOfBookings_T_groupPendingNbOfBookings](#)
- [NbOfBookings_T_staffNbOfBookings](#)
- [NbOfBookings_T_wlNbOfBookings](#)
- [NbOfCancellations_T_nbOfCancellations](#)
- [NbOfBookings_T_etb](#)
- [Availability_T_netClassAvailability](#)
- [Availability_T_segmentAvailability](#)
- [Availability_T_netRevenueAvailability](#)
- [Yield_T_yield](#)
- [MeanValue_T_mean](#)
- [StdDevValue_T_stdDev](#)
- [GeneratedDemandVector_T_generatedDemandVector](#)

Friends

- class [FacBom](#)
- class [FacBomManager](#)

34.25.1 Detailed Description

Class representing the actual attributes for an airline booking class.

Definition at line 24 of file [BookingClass.hpp](#).

34.25.2 Member Typedef Documentation

34.25.2.1 typedef BookingClassKey stdair::BookingClass::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 31 of file [BookingClass.hpp](#).

34.25.3 Constructor & Destructor Documentation

34.25.3.1 stdair::BookingClass::BookingClass (const Key_T & iKey) [protected]

Constructor.

Definition at line 27 of file [BookingClass.cpp](#).

34.25.3.2 stdair::BookingClass::~~BookingClass () [protected, virtual]

Destructor.

Definition at line 39 of file [BookingClass.cpp](#).

34.25.4 Member Function Documentation

34.25.4.1 const Key_T& stdair::BookingClass::getKey () const [inline]

Get the booking class key.

Definition at line 36 of file [BookingClass.hpp](#).

References [_key](#).

34.25.4.2 const ClassCode_T& stdair::BookingClass::getClassCode () const [inline]

Get the booking code (part of the primary key).

Definition at line 41 of file [BookingClass.hpp](#).

References [_key](#), and [stdair::BookingClassKey::getClassCode\(\)](#).

Referenced by [stdair::jsonBookingClassExport\(\)](#).

34.25.4.3 BomAbstract* const stdair::BookingClass::getParent () const [inline]

Get the parent object.

Definition at line 46 of file [BookingClass.hpp](#).

References [_parent](#).

34.25.4.4 const HolderMap_T& stdair::BookingClass::getHolderMap () const [inline]

Get the map of children holders.

Definition at line 51 of file [BookingClass.hpp](#).

References [_holderMap](#).

34.25.4.5 `const SubclassCode_T& stdair::BookingClass::getSubclassCode () const`
`[inline]`

Get teh sub-class code.

Definition at line 56 of file [BookingClass.hpp](#).

References [_subclassCode](#).

Referenced by [stdair::jsonBookingClassExport\(\)](#).

34.25.4.6 `const AuthorizationLevel_T& stdair::BookingClass::getAuthorizationLevel ()`
`const [inline]`

Get the authorisation level (AU, i.e., cumulated protection).

Definition at line 61 of file [BookingClass.hpp](#).

References [_au](#).

34.25.4.7 `const ProtectionLevel_T& stdair::BookingClass::getProtection () const`
`[inline]`

Get the protection.

Definition at line 66 of file [BookingClass.hpp](#).

References [_protection](#).

34.25.4.8 `const ProtectionLevel_T& stdair::BookingClass::getCumulatedProtection ()`
`const [inline]`

Get the cumulated protection.

Definition at line 71 of file [BookingClass.hpp](#).

References [_cumulatedProtection](#).

34.25.4.9 `const BookingLimit_T& stdair::BookingClass::getCumulatedBookingLimit ()`
`const [inline]`

Get the cumulated booking limit.

Definition at line 76 of file [BookingClass.hpp](#).

References [_cumulatedBookingLimit](#).

34.25.4.10 `const NbOfSeats_T& stdair::BookingClass::getNegotiatedSpace () const`
`[inline]`

Get the negotiated space.

Definition at line 81 of file [BookingClass.hpp](#).

References [_nego](#).

34.25.4.11 **const OverbookingRate_T&** stdair::BookingClass::getNoShowPercentage ()
const [inline]

Get the no-show rate.

Definition at line 86 of file [BookingClass.hpp](#).

References [_noShowPercentage](#).

34.25.4.12 **const OverbookingRate_T&** stdair::BookingClass::getCancellationPercentage ()
const [inline]

Get the cancellation rate.

Definition at line 91 of file [BookingClass.hpp](#).

References [_cancellationPercentage](#).

34.25.4.13 **const NbOfBookings_T&** stdair::BookingClass::getNbOfBookings () **const**
[inline]

Get the number of bookings.

Definition at line 96 of file [BookingClass.hpp](#).

References [_nbOfBookings](#).

34.25.4.14 **const NbOfBookings_T&** stdair::BookingClass::getNbOfGroupBookings ()
const [inline]

Get the number of group bookings.

Definition at line 101 of file [BookingClass.hpp](#).

References [_groupNbOfBookings](#).

34.25.4.15 **const NbOfBookings_T&** stdair::BookingClass::getNbOfPendingGroupBookings ()
const [inline]

Get the number of pending group bookings.

Definition at line 106 of file [BookingClass.hpp](#).

References [_groupPendingNbOfBookings](#).

34.25.4.16 **const NbOfBookings_T&** stdair::BookingClass::getNbOfStaffBookings () **const**
[inline]

Get the number of staff bookings.

Definition at line 111 of file [BookingClass.hpp](#).

References [_staffNbOfBookings](#).

34.25.4.17 `const NbOfBookings_T& stdair::BookingClass::getNbOfWLBookings () const`
`[inline]`

Get the number of wait-list bookings.

Definition at line 116 of file [BookingClass.hpp](#).

References [_wNbOfBookings](#).

34.25.4.18 `const NbOfCancellations_T& stdair::BookingClass::getNbOfCancellations ()`
`const [inline]`

Get the number of cancellations.

Definition at line 121 of file [BookingClass.hpp](#).

References [_nbOfCancellations](#).

34.25.4.19 `const NbOfBookings_T& stdair::BookingClass::getETB () const` `[inline]`

Get the expected number of passengers to board (ETB).

Definition at line 126 of file [BookingClass.hpp](#).

References [_etb](#).

34.25.4.20 `const Availability_T& stdair::BookingClass::getNetClassAvailability () const`
`[inline]`

Get the net segment class availability.

Definition at line 131 of file [BookingClass.hpp](#).

References [_netClassAvailability](#).

34.25.4.21 `const Availability_T& stdair::BookingClass::getSegmentAvailability () const`
`[inline]`

Get the segment class availability.

Definition at line 136 of file [BookingClass.hpp](#).

References [_segmentAvailability](#).

34.25.4.22 `const Availability_T& stdair::BookingClass::getNetRevenueAvailability () const`
`[inline]`

Net revenue availability.

Definition at line 141 of file [BookingClass.hpp](#).

References [_netRevenueAvailability](#).

34.25.4.23 `const Yield_T& stdair::BookingClass::getYield () const` `[inline]`

Yield.

Definition at line 146 of file [BookingClass.hpp](#).

References [_yield](#).

34.25.4.24 `const MeanValue_T& stdair::BookingClass::getMean () const` `[inline]`

Demand distribution.

Definition at line 149 of file [BookingClass.hpp](#).

References [_mean](#).

34.25.4.25 `const StdDevValue_T& stdair::BookingClass::getStdDev () const`
`[inline]`

Definition at line 150 of file [BookingClass.hpp](#).

References [_stdDev](#).

34.25.4.26 `const GeneratedDemandVector_T&`
`stdair::BookingClass::getGeneratedDemandVector () const`
`[inline]`

Generated demand vector.

Definition at line 153 of file [BookingClass.hpp](#).

References [_generatedDemandVector](#).

Referenced by [stdair::VirtualClassStruct::getGeneratedDemandVector\(\)](#).

34.25.4.27 `void stdair::BookingClass::setCumulatedProtection (const ProtectionLevel_T &`
`iPL)` `[inline]`

Cumulated protection.

Definition at line 160 of file [BookingClass.hpp](#).

References [_cumulatedProtection](#).

34.25.4.28 `void stdair::BookingClass::setProtection (const ProtectionLevel_T & iPL)`
`[inline]`

Protection.

Definition at line 165 of file [BookingClass.hpp](#).

References [_protection](#).

34.25.4.29 `void stdair::BookingClass::setCumulatedBookingLimit (const BookingLimit_T &`
`iBL)` `[inline]`

Cumulated booking limit.

Definition at line 170 of file [BookingClass.hpp](#).

References [_cumulatedBookingLimit](#).

34.25.4.30 void stdair::BookingClass::setAuthorizationLevel (const AuthorizationLevel_T & iAU) [inline]

Authorization level.

Definition at line 175 of file [BookingClass.hpp](#).

References [_au](#).

34.25.4.31 void stdair::BookingClass::setSegmentAvailability (const Availability_T & iAvl) [inline]

Set availability.

Definition at line 180 of file [BookingClass.hpp](#).

References [_segmentAvailability](#).

34.25.4.32 void stdair::BookingClass::setYield (const Yield_T & iYield) [inline]

Yield.

Definition at line 185 of file [BookingClass.hpp](#).

References [_yield](#).

34.25.4.33 void stdair::BookingClass::setMean (const MeanValue_T & iMean) [inline]

Demand distribution.

Definition at line 188 of file [BookingClass.hpp](#).

References [_mean](#).

34.25.4.34 void stdair::BookingClass::setStdDev (const StdDevValue_T & iStdDev) [inline]

Definition at line 189 of file [BookingClass.hpp](#).

References [_stdDev](#).

34.25.4.35 void stdair::BookingClass::toStream (std::ostream & ioOut) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Implements [stdair::BomAbstract](#).

Definition at line 195 of file [BookingClass.hpp](#).

References [toString\(\)](#).

34.25.4.36 void stdair::BookingClass::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters

<i>istream&</i> the input stream.

Implements [stdair::BomAbstract](#).

Definition at line 201 of file [BookingClass.hpp](#).

34.25.4.37 std::string stdair::BookingClass::toString () const [virtual]

Get the serialised version of the Business Object.

Implements [stdair::BomAbstract](#).

Definition at line 43 of file [BookingClass.cpp](#).

References [describeKey\(\)](#).

Referenced by [toStream\(\)](#).

34.25.4.38 const std::string stdair::BookingClass::describeKey () const [inline]

Get a string describing the key.

Definition at line 208 of file [BookingClass.hpp](#).

References [_key](#), and [stdair::BookingClassKey::toString\(\)](#).

Referenced by [toString\(\)](#).

34.25.4.39 void stdair::BookingClass::sell (const NbOfBookings_T & *iNbOfBookings*)

Register a sale.

Definition at line 50 of file [BookingClass.cpp](#).

References [_nbOfBookings](#).

34.25.4.40 void stdair::BookingClass::cancel (const NbOfBookings_T & *iNbOfCancellations*)

Register a cancellation.

Definition at line 55 of file [BookingClass.cpp](#).

References [_nbOfBookings](#), and [_nbOfCancellations](#).

34.25.4.41 void stdair::BookingClass::generateDemandSamples (const int & *K*)

Generate demand samples for Monte-Carlo method with the default random seed.

Definition at line 61 of file [BookingClass.cpp](#).

References [_generatedDemandVector](#), [_mean](#), [_stdDev](#), [stdair::DEFAULT_RANDOM_](#)

[SEED](#), and [stdair::RandomGeneration::generateNormal\(\)](#).

34.25.4.42 void [stdair::BookingClass::generateDemandSamples](#) (const int & *K*, const [RandomSeed_T](#) & *iSeed*)

Generate demand samples for Monte-Carlo method with the given random seed.

Definition at line 73 of file [BookingClass.cpp](#).

References [_generatedDemandVector](#), [_mean](#), [_stdDev](#), and [stdair::RandomGeneration::generateNormal\(\)](#).

34.25.5 Friends And Related Function Documentation

34.25.5.1 friend class [FacBom](#) [[friend](#)]

Definition at line 25 of file [BookingClass.hpp](#).

34.25.5.2 friend class [FacBomManager](#) [[friend](#)]

Definition at line 26 of file [BookingClass.hpp](#).

34.25.6 Member Data Documentation

34.25.6.1 [Key_T](#) [stdair::BookingClass::_key](#) [[protected](#)]

Primary key (booking class code).

Definition at line 245 of file [BookingClass.hpp](#).

Referenced by [describeKey\(\)](#), [getClassCode\(\)](#), and [getKey\(\)](#).

34.25.6.2 [BomAbstract*](#) [stdair::BookingClass::_parent](#) [[protected](#)]

Pointer on the parent class ([SegmentCabin](#)).

Definition at line 248 of file [BookingClass.hpp](#).

Referenced by [getParent\(\)](#).

34.25.6.3 [HolderMap_T](#) [stdair::BookingClass::_holderMap](#) [[protected](#)]

Map holding the children ([SegmentDate](#) and [LegDate](#) objects).

Definition at line 251 of file [BookingClass.hpp](#).

Referenced by [getHolderMap\(\)](#).

34.25.6.4 [SubclassCode_T](#) [stdair::BookingClass::_subclassCode](#) [[protected](#)]

Sub-class code.

Definition at line 254 of file [BookingClass.hpp](#).

Referenced by [getSubclassCode\(\)](#).

34.25.6.5 `ProtectionLevel_T stdair::BookingClass::_cumulatedProtection`
[protected]

Cumulated protection.

Definition at line 257 of file [BookingClass.hpp](#).

Referenced by [getCumulatedProtection\(\)](#), and [setCumulatedProtection\(\)](#).

34.25.6.6 `ProtectionLevel_T stdair::BookingClass::_protection` [protected]

Protection.

Definition at line 260 of file [BookingClass.hpp](#).

Referenced by [getProtection\(\)](#), and [setProtection\(\)](#).

34.25.6.7 `BookingLimit_T stdair::BookingClass::_cumulatedBookingLimit`
[protected]

Cumulated booking limit.

Definition at line 263 of file [BookingClass.hpp](#).

Referenced by [getCumulatedBookingLimit\(\)](#), and [setCumulatedBookingLimit\(\)](#).

34.25.6.8 `AuthorizationLevel_T stdair::BookingClass::_au` [protected]

Authorization level.

Definition at line 266 of file [BookingClass.hpp](#).

Referenced by [getAuthorizationLevel\(\)](#), and [setAuthorizationLevel\(\)](#).

34.25.6.9 `NbOfSeats_T stdair::BookingClass::_nego` [protected]

Negotiated space.

Definition at line 269 of file [BookingClass.hpp](#).

Referenced by [getNegotiatedSpace\(\)](#).

34.25.6.10 `OverbookingRate_T stdair::BookingClass::_noShowPercentage`
[protected]

Overbooking rate.

Definition at line 272 of file [BookingClass.hpp](#).

Referenced by [getNoShowPercentage\(\)](#).

34.25.6.11 `OverbookingRate_T stdair::BookingClass::_cancellationPercentage`
[protected]

Cancellation rate.

Definition at line 275 of file [BookingClass.hpp](#).

Referenced by [getCancellationPercentage\(\)](#).

34.25.6.12 NbOfBookings_T stdair::BookingClass::_nbOfBookings
[protected]

Number of bookings.

Definition at line 278 of file [BookingClass.hpp](#).

Referenced by [cancel\(\)](#), [getNbOfBookings\(\)](#), and [sell\(\)](#).

34.25.6.13 NbOfBookings_T stdair::BookingClass::_groupNbOfBookings
[protected]

Number of group bookings.

Definition at line 281 of file [BookingClass.hpp](#).

Referenced by [getNbOfGroupBookings\(\)](#).

34.25.6.14 NbOfBookings_T stdair::BookingClass::_groupPendingNbOfBookings
[protected]

Number of pending group bookings.

Definition at line 284 of file [BookingClass.hpp](#).

Referenced by [getNbOfPendingGroupBookings\(\)](#).

34.25.6.15 NbOfBookings_T stdair::BookingClass::_staffNbOfBookings
[protected]

Number of staff bookings.

Definition at line 287 of file [BookingClass.hpp](#).

Referenced by [getNbOfStaffBookings\(\)](#).

34.25.6.16 NbOfBookings_T stdair::BookingClass::_wlnNbOfBookings
[protected]

Number of wait-list bookings.

Definition at line 290 of file [BookingClass.hpp](#).

Referenced by [getNbOfWLBookings\(\)](#).

34.25.6.17 NbOfCancellations_T stdair::BookingClass::_nbOfCancellations
[protected]

Number of cancellations.

Definition at line 293 of file [BookingClass.hpp](#).

Referenced by [cancel\(\)](#), and [getNbOfCancellations\(\)](#).

34.25.6.18 NbOfBookings_T stdair::BookingClass::_etb [protected]

Expected to board (ETB).

Definition at line 296 of file [BookingClass.hpp](#).

Referenced by [getETB\(\)](#).

34.25.6.19 Availability_T stdair::BookingClass::_netClassAvailability [protected]

Net segment class availability.

Definition at line 299 of file [BookingClass.hpp](#).

Referenced by [getNetClassAvailability\(\)](#).

34.25.6.20 Availability_T stdair::BookingClass::_segmentAvailability [protected]

Segment class availability.

Definition at line 302 of file [BookingClass.hpp](#).

Referenced by [getSegmentAvailability\(\)](#), and [setSegmentAvailability\(\)](#).

34.25.6.21 Availability_T stdair::BookingClass::_netRevenueAvailability [protected]

Net revenue availability.

Definition at line 305 of file [BookingClass.hpp](#).

Referenced by [getNetRevenueAvailability\(\)](#).

34.25.6.22 Yield_T stdair::BookingClass::_yield [protected]

Yield.

Definition at line 308 of file [BookingClass.hpp](#).

Referenced by [getYield\(\)](#), and [setYield\(\)](#).

34.25.6.23 MeanValue_T stdair::BookingClass::_mean [protected]

Remaining demand distribution forecast.

Definition at line 311 of file [BookingClass.hpp](#).

Referenced by [generateDemandSamples\(\)](#), [getMean\(\)](#), and [setMean\(\)](#).

34.25.6.24 StdDevValue_T stdair::BookingClass::_stdDev [protected]

Definition at line 312 of file [BookingClass.hpp](#).

Referenced by [generateDemandSamples\(\)](#), [getStdDev\(\)](#), and [setStdDev\(\)](#).

34.25.6.25 GeneratedDemandVector_T stdair::BookingClass::_generatedDemandVector [protected]

Vector of number of demand samples drawn from the demand distribution.

Definition at line 315 of file [BookingClass.hpp](#).

Referenced by [generateDemandSamples\(\)](#), and [getGeneratedDemandVector\(\)](#).

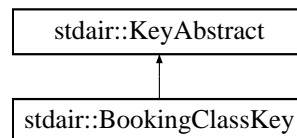
The documentation for this class was generated from the following files:

- [stdair/bom/BookingClass.hpp](#)
- [stdair/bom/BookingClass.cpp](#)

34.26 stdair::BookingClassKey Struct Reference

```
#include <stdair/bom/BookingClassKey.hpp>
```

Inheritance diagram for stdair::BookingClassKey:



Public Member Functions

- [BookingClassKey](#) (const [ClassCode_T](#) &iClassCode)
- [BookingClassKey](#) (const [BookingClassKey](#) &)
- [~BookingClassKey](#) ()
- const [ClassCode_T](#) & [getClassCode](#) () const
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- const std::string [toString](#) () const

34.26.1 Detailed Description

Key of a given leg-cabin, made of a cabin code.

Definition at line 16 of file [BookingClassKey.hpp](#).

34.26.2 Constructor & Destructor Documentation

34.26.2.1 stdair::BookingClassKey::BookingClassKey (const [ClassCode_T](#) & iClassCode)

Constructor.

Definition at line 24 of file [BookingClassKey.cpp](#).

34.26.2.2 stdair::BookingClassKey::BookingClassKey (const [BookingClassKey](#) & iKey)

Default copy constructor.

Definition at line 19 of file [BookingClassKey.cpp](#).

34.26.2.3 stdair::BookingClassKey::~~BookingClassKey ()

Destructor.

Definition at line 29 of file [BookingClassKey.cpp](#).

34.26.3 Member Function Documentation

34.26.3.1 const ClassCode_T& stdair::BookingClassKey::getClassCode () const [inline]

Get the class code.

Definition at line 34 of file [BookingClassKey.hpp](#).

Referenced by [stdair::BookingClass::getClassCode\(\)](#).

34.26.3.2 void stdair::BookingClassKey::toStream (std::ostream & ioOut) const [virtual]

Dump a Business Object Key into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 33 of file [BookingClassKey.cpp](#).

References [toString\(\)](#).

34.26.3.3 void stdair::BookingClassKey::fromStream (std::istream & ioIn) [virtual]

Read a Business Object Key from an input stream.

Parameters

<i>istream&</i> the input stream.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 38 of file [BookingClassKey.cpp](#).

34.26.3.4 const std::string stdair::BookingClassKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-cabin.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 42 of file [BookingClassKey.cpp](#).

Referenced by [stdair::BookingClass::describeKey\(\)](#), and [toStream\(\)](#).

The documentation for this struct was generated from the following files:

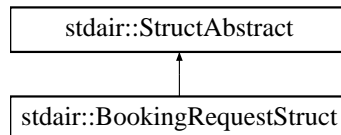
- [stdair/bom/BookingClassKey.hpp](#)
- [stdair/bom/BookingClassKey.cpp](#)

34.27 stdair::BookingRequestStruct Struct Reference

Structure holding the elements of a booking request.

```
#include <stdair/bom/BookingRequestStruct.hpp>
```

Inheritance diagram for stdair::BookingRequestStruct:



Public Member Functions

- const [DemandGeneratorKey_T](#) & [getDemandGeneratorKey](#) () const
- const [AirportCode_T](#) & [getOrigin](#) () const
- const [AirportCode_T](#) & [getDestination](#) () const
- const [CityCode_T](#) & [getPOS](#) () const
- const [Date_T](#) & [getPreferredDepartureDate](#) () const
- const [Duration_T](#) & [getPreferredDepartureTime](#) () const
- const [DateTime_T](#) & [getRequestDateTime](#) () const
- const [CabinCode_T](#) & [getPreferredCabin](#) () const
- const [NbOfSeats_T](#) & [getPartySize](#) () const
- const [Channellabel_T](#) & [getBookingChannel](#) () const
- const [TripType_T](#) & [getTripType](#) () const
- const [DayDuration_T](#) & [getStayDuration](#) () const
- const [FrequentFlyer_T](#) & [getFrequentFlyerType](#) () const
- const [WTP_T](#) & [getWTP](#) () const
- const [PriceValue_T](#) & [getValueOfTime](#) () const
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- const std::string [describe](#) () const
- const std::string [display](#) () const
- [BookingRequestStruct](#) (const [DemandGeneratorKey_T](#) &iGeneratorKey, const [AirportCode_T](#) &iOrigin, const [AirportCode_T](#) &iDestination, const [CityCode_T](#) &iPOS, const

- [Date_T](#) &iDepartureDate, const [DateTime_T](#) &iRequestDateTime, const [CabinCode_T](#) &iPreferredCabin, const [NbOfSeats_T](#) &iPartySize, const [ChannelLabel_T](#) &iChannel, const [TripType_T](#) &iTripType, const [DayDuration_T](#) &iStayDuration, const [FrequentFlyer_T](#) &iFrequentFlyerType, const [Duration_T](#) &iPreferredDepartureTime, const [WTP_T](#) &iWTP, const [PriceValue_T](#) &iValueOfTime)
- [BookingRequestStruct](#) (const [AirportCode_T](#) &iOrigin, const [AirportCode_T](#) &iDestination, const [CityCode_T](#) &iPOS, const [Date_T](#) &iDepartureDate, const [DateTime_T](#) &iRequestDateTime, const [CabinCode_T](#) &iPreferredCabin, const [NbOfSeats_T](#) &iPartySize, const [ChannelLabel_T](#) &iChannel, const [TripType_T](#) &iTripType, const [DayDuration_T](#) &iStayDuration, const [FrequentFlyer_T](#) &iFrequentFlyerType, const [Duration_T](#) &iPreferredDepartureTime, const [WTP_T](#) &iWTP, const [PriceValue_T](#) &iValueOfTime)
- [BookingRequestStruct](#) (const [BookingRequestStruct](#) &)
- [~BookingRequestStruct](#) ()

34.27.1 Detailed Description

Structure holding the elements of a booking request.

Definition at line 21 of file [BookingRequestStruct.hpp](#).

34.27.2 Constructor & Destructor Documentation

34.27.2.1 `stdair::BookingRequestStruct::BookingRequestStruct (const DemandGeneratorKey_T &iGeneratorKey, const AirportCode_T &iOrigin, const AirportCode_T &iDestination, const CityCode_T &iPOS, const Date_T &iDepartureDate, const DateTime_T &iRequestDateTime, const CabinCode_T &iPreferredCabin, const NbOfSeats_T &iPartySize, const ChannelLabel_T &iChannel, const TripType_T &iTripType, const DayDuration_T &iStayDuration, const FrequentFlyer_T &iFrequentFlyerType, const Duration_T &iPreferredDepartureTime, const WTP_T &iWTP, const PriceValue_T &iValueOfTime)`

Default constructor.

Definition at line 57 of file [BookingRequestStruct.cpp](#).

34.27.2.2 `stdair::BookingRequestStruct::BookingRequestStruct (const AirportCode_T &iOrigin, const AirportCode_T &iDestination, const CityCode_T &iPOS, const Date_T &iDepartureDate, const DateTime_T &iRequestDateTime, const CabinCode_T &iPreferredCabin, const NbOfSeats_T &iPartySize, const ChannelLabel_T &iChannel, const TripType_T &iTripType, const DayDuration_T &iStayDuration, const FrequentFlyer_T &iFrequentFlyerType, const Duration_T &iPreferredDepartureTime, const WTP_T &iWTP, const PriceValue_T &iValueOfTime)`

Constructor without the demand generator key, used for batches.

Definition at line 85 of file [BookingRequestStruct.cpp](#).

34.27.2.3 `stdair::BookingRequestStruct::BookingRequestStruct (const
BookingRequestStruct & iBookingRequest)`

Copy constructor.

Definition at line 37 of file [BookingRequestStruct.cpp](#).

34.27.2.4 `stdair::BookingRequestStruct::~~BookingRequestStruct ()`

Destructor.

Definition at line 111 of file [BookingRequestStruct.cpp](#).

34.27.3 Member Function Documentation

34.27.3.1 `const DemandGeneratorKey_T&
stdair::BookingRequestStruct::getDemandGeneratorKey ()
const [inline]`

Get the demand generator key.

Definition at line 25 of file [BookingRequestStruct.hpp](#).

34.27.3.2 `const AirportCode_T& stdair::BookingRequestStruct::getOrigin () const
[inline]`

Get the requested origin.

Definition at line 30 of file [BookingRequestStruct.hpp](#).

34.27.3.3 `const AirportCode_T& stdair::BookingRequestStruct::getDestination () const
[inline]`

Get the requested destination.

Definition at line 35 of file [BookingRequestStruct.hpp](#).

34.27.3.4 `const CityCode_T& stdair::BookingRequestStruct::getPOS () const
[inline]`

Get the point-of-sale.

Definition at line 40 of file [BookingRequestStruct.hpp](#).

34.27.3.5 `const Date_T& stdair::BookingRequestStruct::getPreferredDepartureDate () const
[inline]`

Get the requested departure date.

Definition at line 45 of file [BookingRequestStruct.hpp](#).

34.27.3.6 **const Duration_T& stdair::BookingRequestStruct::getPreferredDepartureTime ()**
const [inline]

Get the preferred departure time.

Definition at line 50 of file [BookingRequestStruct.hpp](#).

34.27.3.7 **const DateTime_T& stdair::BookingRequestStruct::getRequestDateTime () const**
[inline]

Get the request datetime.

Definition at line 55 of file [BookingRequestStruct.hpp](#).

34.27.3.8 **const CabinCode_T& stdair::BookingRequestStruct::getPreferredCabin () const**
[inline]

Get the preferred cabin.

Definition at line 60 of file [BookingRequestStruct.hpp](#).

34.27.3.9 **const NbOfSeats_T& stdair::BookingRequestStruct::getPartySize () const**
[inline]

Get the party size.

Definition at line 65 of file [BookingRequestStruct.hpp](#).

34.27.3.10 **const ChannelLabel_T& stdair::BookingRequestStruct::getBookingChannel ()**
const [inline]

Get the reservation channel.

Definition at line 70 of file [BookingRequestStruct.hpp](#).

34.27.3.11 **const TripType_T& stdair::BookingRequestStruct::getTripType () const**
[inline]

Get the trip type.

Definition at line 75 of file [BookingRequestStruct.hpp](#).

34.27.3.12 **const DayDuration_T& stdair::BookingRequestStruct::getStayDuration () const**
[inline]

Get the duration of stay.

Definition at line 80 of file [BookingRequestStruct.hpp](#).

34.27.3.13 **const FrequentFlyer_T& stdair::BookingRequestStruct::getFrequentFlyerType ()**
const [inline]

Get the frequent flyer type.

Definition at line 85 of file [BookingRequestStruct.hpp](#).

34.27.3.14 `const WTP_T& stdair::BookingRequestStruct::getWTP () const` `[inline]`

Get the willingness-to-pay.

Definition at line 90 of file [BookingRequestStruct.hpp](#).

34.27.3.15 `const PriceValue_T& stdair::BookingRequestStruct::getValueOfTime () const`
`[inline]`

Get the value of time.

Definition at line 95 of file [BookingRequestStruct.hpp](#).

34.27.3.16 `void stdair::BookingRequestStruct::toStream (std::ostream & ioOut) const`

Dump a Business Object into an output stream.

Parameters

<code>ostream&</code> the output stream.
--

Reimplemented from [stdair::StructAbstract](#).

Definition at line 115 of file [BookingRequestStruct.cpp](#).

References [describe\(\)](#).

34.27.3.17 `void stdair::BookingRequestStruct::fromStream (std::istream & ioIn)`
`[virtual]`

Read a Business Object from an input stream.

Parameters

<code>istream&</code> the input stream.

Reimplemented from [stdair::StructAbstract](#).

Definition at line 120 of file [BookingRequestStruct.cpp](#).

34.27.3.18 `const std::string stdair::BookingRequestStruct::describe () const` `[virtual]`

Display of the structure.

Implements [stdair::StructAbstract](#).

Definition at line 124 of file [BookingRequestStruct.cpp](#).

Referenced by [toStream\(\)](#).

34.27.3.19 `const std::string stdair::BookingRequestStruct::display () const`

Display of the structure.

- #id,
- request_date (YYMMDD),

- request_time (HHMMSS),
- POS (three-letter code),
- Channel (two-letter code):
 - 'D' for direct or 'I' for indirect,
 - 'N' for oNline or 'F' for oFfline,
- Origin (three-letter code),
- Destination (three-letter code),
- Preferred departure date (YYMMDD),
- Preferred departure time (HHMM),
- Min departure time (HHMM),
- Max departure time (HHMM),
- Preferred arrival date (YYMMDD),
- Preferred arrival time (HHMM),
- Preferred cabin:
 - 'F' for first,
 - 'C' for club/business,
 - 'W' for economy plus,
 - 'M' for economy,
- Trip type:
 - 'OW' for a one-way trip,
 - 'RO' for the outbound part of a rount-trip,
 - 'RI' for the inbound part of a rount-trip,
- Duration of stay (expressed as a number of days),
- Frequent flyer tier:
 - 'G' for gold,
 - 'S' for silver,
 - 'K' for basic,
 - 'N' for none,
- Willingness-to-pay (WTP, expressed as a monetary unit, e.g., EUR),
- Disutility per stop (expressed as a monetary unit, e.g., EUR),
- Value of time (EUR per hour),

Returns

const std::string The output of the booking request structure.

Definition at line 147 of file [BookingRequestStruct.cpp](#).

References [stdair::TRIP_TYPE_ONE_WAY](#).

The documentation for this struct was generated from the following files:

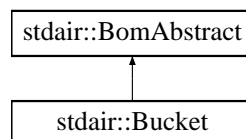
- [stdair/bom/BookingRequestStruct.hpp](#)
- [stdair/bom/BookingRequestStruct.cpp](#)

34.28 stdair::Bucket Class Reference

Class representing the actual attributes for an airline booking class.

```
#include <stdair/bom/Bucket.hpp>
```

Inheritance diagram for stdair::Bucket:

**Public Types**

- typedef [BucketKey](#) Key_T

Public Member Functions

- const [Key_T](#) & [getKey](#) () const
- [BomAbstract](#) *const [getParent](#) () const
- const [HolderMap_T](#) & [getHolderMap](#) () const
- const [SeatIndex_T](#) & [getSeatIndex](#) () const
- const [Yield_T](#) & [getYieldRangeUpperValue](#) () const
- const [CabinCapacity_T](#) & [getAvailability](#) () const
- const [NbOfSeats_T](#) & [getSoldSeats](#) () const
- void [setYieldRangeUpperValue](#) (const [Yield_T](#) &iYield)
- void [setAvailability](#) (const [CabinCapacity_T](#) &iAvl)
- void [setSoldSeats](#) (const [NbOfSeats_T](#) &iSoldSeats)
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- std::string [toString](#) () const
- const std::string [describeKey](#) () const
- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int iFileVersion)

Protected Member Functions

- [Bucket](#) (const [Key_T](#) &)
- virtual [~Bucket](#) ()

Protected Attributes

- [Key_T_key](#)
- [BomAbstract](#) * [_parent](#)
- [HolderMap_T_holderMap](#)
- [Yield_T_yieldRangeUpperValue](#)
- [CabinCapacity_T_availability](#)
- [NbOfSeats_T_soldSeats](#)

Friends

- class [FacBom](#)
- class [FacBomManager](#)
- class [boost::serialization::access](#)

34.28.1 Detailed Description

Class representing the actual attributes for an airline booking class.

Definition at line 29 of file [Bucket.hpp](#).

34.28.2 Member Typedef Documentation**34.28.2.1 typedef [BucketKey](#) [stdair::Bucket::Key_T](#)**

Definition allowing to retrieve the associated BOM key type.

Definition at line 39 of file [Bucket.hpp](#).

34.28.3 Constructor & Destructor Documentation**34.28.3.1 [stdair::Bucket::Bucket](#) ([const Key_T](#) & *iKey*) [protected]**

Default constructor.

Definition at line 24 of file [Bucket.cpp](#).

34.28.3.2 [stdair::Bucket::~Bucket](#) () [protected, virtual]

Destructor.

Definition at line 28 of file [Bucket.cpp](#).

34.28.4 Member Function Documentation

34.28.4.1 `const Key_T& stdair::Bucket::getKey () const` `[inline]`

Get the primary key of the bucket.

Definition at line 46 of file [Bucket.hpp](#).

References [_key](#).

34.28.4.2 `BomAbstract* const stdair::Bucket::getParent () const` `[inline]`

Get the parent object.

Definition at line 53 of file [Bucket.hpp](#).

References [_parent](#).

34.28.4.3 `const HolderMap_T& stdair::Bucket::getHolderMap () const` `[inline]`

Get the map of children holders.

Definition at line 58 of file [Bucket.hpp](#).

References [_holderMap](#).

34.28.4.4 `const SeatIndex_T& stdair::Bucket::getSeatIndex () const` `[inline]`

Get the seat index (part of the primary key).

Definition at line 63 of file [Bucket.hpp](#).

References [_key](#), and [stdair::BucketKey::getSeatIndex\(\)](#).

34.28.4.5 `const Yield_T& stdair::Bucket::getYieldRangeUpperValue () const` `[inline]`

Get the upper yield range.

Definition at line 68 of file [Bucket.hpp](#).

References [_yieldRangeUpperValue](#).

Referenced by [stdair::jsonBucketExport\(\)](#).

34.28.4.6 `const CabinCapacity_T& stdair::Bucket::getAvailability () const` `[inline]`

Get the availability.

Definition at line 73 of file [Bucket.hpp](#).

References [_availability](#).

34.28.4.7 `const NbOfSeats_T& stdair::Bucket::getSoldSeats () const` `[inline]`

Get the number of seats already sold.

Definition at line 78 of file [Bucket.hpp](#).

References [_soldSeats](#).

34.28.4.8 `void stdair::Bucket::setYieldRangeUpperValue (const Yield_T & iYield)`
`[inline]`

Set the upper yield range.

Definition at line 85 of file [Bucket.hpp](#).

References [_yieldRangeUpperValue](#).

34.28.4.9 `void stdair::Bucket::setAvailability (const CabinCapacity_T & iAvl)`
`[inline]`

Set the availability.

Definition at line 90 of file [Bucket.hpp](#).

References [_availability](#).

34.28.4.10 `void stdair::Bucket::setSoldSeats (const NbOfSeats_T & iSoldSeats)`
`[inline]`

Set the number of seats already sold.

Definition at line 95 of file [Bucket.hpp](#).

References [_soldSeats](#).

34.28.4.11 `void stdair::Bucket::toStream (std::ostream & ioOut) const` `[inline, virtual]`

Dump a Business Object into an output stream.

Parameters

<code>ostream&</code> the output stream.
--

Implements [stdair::BomAbstract](#).

Definition at line 107 of file [Bucket.hpp](#).

References [toString\(\)](#).

34.28.4.12 `void stdair::Bucket::fromStream (std::istream & iIn)` `[inline, virtual]`

Read a Business Object from an input stream.

Parameters

<code>istream&</code> the input stream.

Implements [stdair::BomAbstract](#).

Definition at line 116 of file [Bucket.hpp](#).

34.28.4.13 `std::string stdair::Bucket::toString () const` [virtual]

Get the serialised version of the Business Object.

Implements [stdair::BomAbstract](#).

Definition at line 32 of file [Bucket.cpp](#).

References [describeKey\(\)](#).

Referenced by [toStream\(\)](#).

34.28.4.14 `const std::string stdair::Bucket::describeKey () const` [inline]

Get a string describing the key.

Definition at line 127 of file [Bucket.hpp](#).

References [_key](#), and [stdair::BucketKey::toString\(\)](#).

Referenced by [toString\(\)](#).

34.28.4.15 `template<class Archive > void stdair::Bucket::serialize (Archive & ar, const unsigned int iFileVersion)`

Serialisation.

Definition at line 54 of file [Bucket.cpp](#).

References [_key](#).

34.28.5 Friends And Related Function Documentation

34.28.5.1 `friend class FacBom` [friend]

Definition at line 30 of file [Bucket.hpp](#).

34.28.5.2 `friend class FacBomManager` [friend]

Definition at line 31 of file [Bucket.hpp](#).

34.28.5.3 `friend class boost::serialization::access` [friend]

Definition at line 32 of file [Bucket.hpp](#).

34.28.6 Member Data Documentation

34.28.6.1 `Key_T stdair::Bucket::_key` [protected]

Primary key (upper yield range).

Definition at line 178 of file [Bucket.hpp](#).

Referenced by [describeKey\(\)](#), [getKey\(\)](#), [getSeatIndex\(\)](#), and [serialize\(\)](#).

34.28.6.2 BomAbstract* stdair::Bucket::_parent [protected]

Pointer on the parent class ([LegCabin](#)).

Definition at line 183 of file [Bucket.hpp](#).

Referenced by [getParent\(\)](#).

34.28.6.3 HolderMap_T stdair::Bucket::_holderMap [protected]

Map holding the children (empty for now).

Definition at line 188 of file [Bucket.hpp](#).

Referenced by [getHolderMap\(\)](#).

34.28.6.4 Yield_T stdair::Bucket::_yieldRangeUpperValue [protected]

Upper yield range.

Definition at line 196 of file [Bucket.hpp](#).

Referenced by [getYieldRangeUpperValue\(\)](#), and [setYieldRangeUpperValue\(\)](#).

34.28.6.5 CabinCapacity_T stdair::Bucket::_availability [protected]

Availability.

Definition at line 201 of file [Bucket.hpp](#).

Referenced by [getAvailability\(\)](#), and [setAvailability\(\)](#).

34.28.6.6 NbOfSeats_T stdair::Bucket::_soldSeats [protected]

Number of seats already sold.

Definition at line 206 of file [Bucket.hpp](#).

Referenced by [getSoldSeats\(\)](#), and [setSoldSeats\(\)](#).

The documentation for this class was generated from the following files:

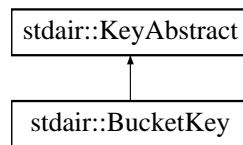
- [stdair/bom/Bucket.hpp](#)
- [stdair/bom/Bucket.cpp](#)

34.29 stdair::BucketKey Struct Reference

Key of booking-class.

```
#include <stdair/bom/BucketKey.hpp>
```

Inheritance diagram for stdair::BucketKey:



Public Member Functions

- [BucketKey](#) (const [SeatIndex_T](#) &)
- [BucketKey](#) (const [BucketKey](#) &)
- [~BucketKey](#) ()
- const [SeatIndex_T](#) & [getSeatIndex](#) () const
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- const std::string [toString](#) () const
- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int iFileVersion)

Friends

- class [boost::serialization::access](#)

34.29.1 Detailed Description

Key of booking-class.

Definition at line 26 of file [BucketKey.hpp](#).

34.29.2 Constructor & Destructor Documentation

34.29.2.1 stdair::BucketKey::BucketKey (const [SeatIndex_T](#) & *iSeatIndex*)

Main constructor.

Definition at line 22 of file [BucketKey.cpp](#).

34.29.2.2 stdair::BucketKey::BucketKey (const [BucketKey](#) & *iBucketKey*)

Copy constructor.

Definition at line 27 of file [BucketKey.cpp](#).

34.29.2.3 stdair::BucketKey::~~BucketKey ()

Destructor.

Definition at line 32 of file [BucketKey.cpp](#).

34.29.3 Member Function Documentation

34.29.3.1 `const SeatIndex_T& stdair::BucketKey::getSeatIndex () const` `[inline]`

Get the seat index.

Definition at line 54 of file [BucketKey.hpp](#).

Referenced by [stdair::Bucket::getSeatIndex\(\)](#).

34.29.3.2 `void stdair::BucketKey::toStream (std::ostream & ioOut) const` `[virtual]`

Dump a Business Object Key into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 36 of file [BucketKey.cpp](#).

References [toString\(\)](#).

34.29.3.3 `void stdair::BucketKey::fromStream (std::istream & iIn)` `[virtual]`

Read a Business Object Key from an input stream.

Parameters

<i>istream&</i> the input stream.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 41 of file [BucketKey.cpp](#).

34.29.3.4 `const std::string stdair::BucketKey::toString () const` `[virtual]`

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-cabin.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 45 of file [BucketKey.cpp](#).

Referenced by [stdair::Bucket::describeKey\(\)](#), and [toStream\(\)](#).

34.29.3.5 `template<class Archive > void stdair::BucketKey::serialize (Archive & ar, const unsigned int iFileVersion)`

Serialisation.

Definition at line 67 of file [BucketKey.cpp](#).

34.29.4 Friends And Related Function Documentation

34.29.4.1 friend class boost::serialization::access [friend]

Definition at line 27 of file [BucketKey.hpp](#).

The documentation for this struct was generated from the following files:

- [stdair/bom/BucketKey.hpp](#)
- [stdair/bom/BucketKey.cpp](#)

34.30 stdair_test::Cabin Struct Reference

```
#include <test/stdair/StdairTestLib.hpp>
```

Public Types

- typedef [BookingClass](#) child

Public Member Functions

- [Cabin](#) (const [BookingClass](#) &iBkgClass)
- std::string [toString](#) () const

Public Attributes

- [BookingClass _bookingClass](#)

34.30.1 Detailed Description

[Cabin](#)

Definition at line 32 of file [StdairTestLib.hpp](#).

34.30.2 Member Typedef Documentation

34.30.2.1 typedef [BookingClass](#) stdair_test::Cabin::child

Child type.

Definition at line 46 of file [StdairTestLib.hpp](#).

34.30.3 Constructor & Destructor Documentation

34.30.3.1 `stdair_test::Cabin::Cabin (const BookingClass & iBkgClass)` `[inline]`

Definition at line 34 of file [StdairTestLib.hpp](#).

34.30.4 Member Function Documentation

34.30.4.1 `std::string stdair_test::Cabin::toString () const` `[inline]`

Display .

Definition at line 39 of file [StdairTestLib.hpp](#).

References [_bookingClass](#), and [stdair_test::BookingClass::_classCode](#).

34.30.5 Member Data Documentation

34.30.5.1 **BookingClass** `stdair_test::Cabin::_bookingClass`

Definition at line 33 of file [StdairTestLib.hpp](#).

Referenced by [toString\(\)](#).

The documentation for this struct was generated from the following file:

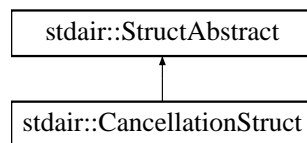
- [test/stdair/StdairTestLib.hpp](#)

34.31 stdair::CancellationStruct Struct Reference

Structure holding the elements of a travel solution.

```
#include <stdair/bom/CancellationStruct.hpp>
```

Inheritance diagram for `stdair::CancellationStruct`:



Public Member Functions

- `const SegmentPath_T & getSegmentPath () const`
- `const ClassList_String_T & getClassList () const`
- `const PartySize_T & getPartySize () const`
- `const DateTime_T & getCancellationDateTime () const`
- `void toStream (std::ostream &ioOut) const`

- void [fromStream](#) (std::istream &ioln)
- const std::string [describe](#) () const
- const std::string [display](#) () const
- [CancellationStruct](#) (const [SegmentPath_T](#) &, const [ClassList_String_T](#) &, const [PartySize_T](#) &, const [DateTime_T](#) &)
- [~CancellationStruct](#) ()

34.31.1 Detailed Description

Structure holding the elements of a travel solution.

Definition at line 22 of file [CancellationStruct.hpp](#).

34.31.2 Constructor & Destructor Documentation

34.31.2.1 `stdair::CancellationStruct::CancellationStruct (const SegmentPath_T &
iSegPath, const ClassList_String_T & iList, const PartySize_T & iSize, const
DateTime_T & iDateTime)`

Default constructor.

Definition at line 13 of file [CancellationStruct.cpp](#).

34.31.2.2 `stdair::CancellationStruct::~~CancellationStruct ()`

Destructor.

Definition at line 22 of file [CancellationStruct.cpp](#).

34.31.3 Member Function Documentation

34.31.3.1 `const SegmentPath_T& stdair::CancellationStruct::getSegmentPath () const
[inline]`

Get the segment path.

Definition at line 26 of file [CancellationStruct.hpp](#).

34.31.3.2 `const ClassList_String_T& stdair::CancellationStruct::getClassList () const
[inline]`

Get the class list.

Definition at line 31 of file [CancellationStruct.hpp](#).

34.31.3.3 `const PartySize_T& stdair::CancellationStruct::getPartySize () const
[inline]`

Get the party size.

Definition at line 36 of file [CancellationStruct.hpp](#).

34.31.3.4 `const DateTime_T& stdair::CancellationStruct::getCancellationDateTime () const`
[inline]

Get the datetime.

Definition at line 41 of file [CancellationStruct.hpp](#).

34.31.3.5 `void stdair::CancellationStruct::toStream (std::ostream & ioOut) const`

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Reimplemented from [stdair::StructAbstract](#).

Definition at line 26 of file [CancellationStruct.cpp](#).

References [describe\(\)](#).

34.31.3.6 `void stdair::CancellationStruct::fromStream (std::istream & ioin)` [virtual]

Read a Business Object from an input stream.

Parameters

<i>istream&</i> the input stream.

Reimplemented from [stdair::StructAbstract](#).

Definition at line 31 of file [CancellationStruct.cpp](#).

34.31.3.7 `const std::string stdair::CancellationStruct::describe () const` [virtual]

Display of the structure.

Implements [stdair::StructAbstract](#).

Definition at line 35 of file [CancellationStruct.cpp](#).

Referenced by [toStream\(\)](#).

34.31.3.8 `const std::string stdair::CancellationStruct::display () const`

Display of the structure.

Definition at line 54 of file [CancellationStruct.cpp](#).

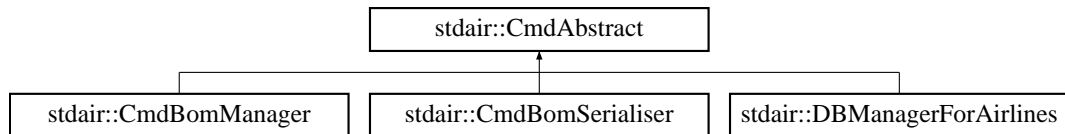
The documentation for this struct was generated from the following files:

- [stdair/bom/CancellationStruct.hpp](#)
- [stdair/bom/CancellationStruct.cpp](#)

34.32 stdair::CmdAbstract Class Reference

```
#include <stdair/command/CmdAbstract.hpp>
```

Inheritance diagram for stdair::CmdAbstract:



34.32.1 Detailed Description

Base class for the Command layer.

Definition at line 11 of file [CmdAbstract.hpp](#).

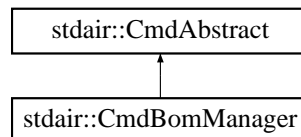
The documentation for this class was generated from the following file:

- [stdair/command/CmdAbstract.hpp](#)

34.33 stdair::CmdBomManager Class Reference

```
#include <stdair/command/CmdBomManager.hpp>
```

Inheritance diagram for stdair::CmdBomManager:



Friends

- class [STDAIR_Service](#)

34.33.1 Detailed Description

Class wrapping utility functions for handling the BOM tree objects.

Definition at line 25 of file [CmdBomManager.hpp](#).

34.33.2 Friends And Related Function Documentation

34.33.2.1 friend class STDAIR_Service [friend]

Definition at line 27 of file [CmdBomManager.hpp](#).

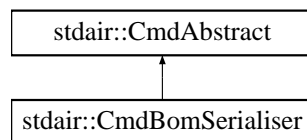
The documentation for this class was generated from the following file:

- [stdair/command/CmdBomManager.hpp](#)

34.34 stdair::CmdBomSerialiser Class Reference

```
#include <stdair/command/CmdBomSerialiser.hpp>
```

Inheritance diagram for stdair::CmdBomSerialiser:



34.34.1 Detailed Description

Class wrapping utility functions for handling the BOM tree objects.

Definition at line 25 of file [CmdBomSerialiser.hpp](#).

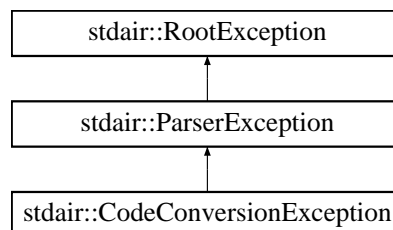
The documentation for this class was generated from the following file:

- [stdair/command/CmdBomSerialiser.hpp](#)

34.35 stdair::CodeConversionException Class Reference

```
#include <stdair/stdair_exceptions.hpp>
```

Inheritance diagram for stdair::CodeConversionException:



Public Member Functions

- [CodeConversionException](#) (const std::string &iWhat)
- const char * [what](#) () const throw ()

Protected Attributes

- `std::string _what`

34.35.1 Detailed Description

Code conversion.

Definition at line 133 of file [stdair_exceptions.hpp](#).

34.35.2 Constructor & Destructor Documentation

34.35.2.1 `stdair::CodeConversionException::CodeConversionException (const std::string & iWhat) [inline]`

Constructor.

Definition at line 136 of file [stdair_exceptions.hpp](#).

34.35.3 Member Function Documentation

34.35.3.1 `const char* stdair::RootException::what () const throw () [inline, inherited]`

Give the details of the exception.

Definition at line 38 of file [stdair_exceptions.hpp](#).

References [stdair::RootException::_what](#).

34.35.4 Member Data Documentation

34.35.4.1 `std::string stdair::RootException::_what [protected, inherited]`

Details for the exception.

Definition at line 46 of file [stdair_exceptions.hpp](#).

Referenced by [stdair::RootException::what\(\)](#).

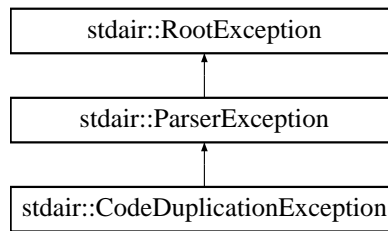
The documentation for this class was generated from the following file:

- [stdair/stdair_exceptions.hpp](#)

34.36 stdair::CodeDuplicationException Class Reference

```
#include <stdair/stdair_exceptions.hpp>
```

Inheritance diagram for `stdair::CodeDuplicationException`:



Public Member Functions

- [CodeDuplicationException](#) (const std::string &iWhat)
- const char * [what](#) () const throw ()

Protected Attributes

- std::string [_what](#)

34.36.1 Detailed Description

Code duplication.

Definition at line 141 of file [stdair_exceptions.hpp](#).

34.36.2 Constructor & Destructor Documentation

34.36.2.1 stdair::CodeDuplicationException::CodeDuplicationException (const std::string &iWhat) [inline]

Constructor.

Definition at line 144 of file [stdair_exceptions.hpp](#).

34.36.3 Member Function Documentation

34.36.3.1 const char* stdair::RootException::what () const throw () [inline, inherited]

Give the details of the exception.

Definition at line 38 of file [stdair_exceptions.hpp](#).

References [stdair::RootException::_what](#).

34.36.4 Member Data Documentation

34.36.4.1 `std::string stdair::RootException::_what` [protected, inherited]

Details for the exception.

Definition at line 46 of file [stdair_exceptions.hpp](#).

Referenced by [stdair::RootException::what\(\)](#).

The documentation for this class was generated from the following file:

- [stdair/stdair_exceptions.hpp](#)

34.37 COMMAND Struct Reference

```
#include <stdair/ui/cmdline/readline_autocomp.hpp>
```

Public Attributes

- `char const * name`
- `pt2Func * func`
- `char * doc`

34.37.1 Detailed Description

A structure which contains information on the commands this program can understand.

Definition at line 41 of file [readline_autocomp.hpp](#).

34.37.2 Member Data Documentation

34.37.2.1 `char const* COMMAND::name`

User printable name of the function.

Definition at line 45 of file [readline_autocomp.hpp](#).

Referenced by [com_help\(\)](#), and [find_command\(\)](#).

34.37.2.2 `pt2Func* COMMAND::func`

Function to call to do the job.

Definition at line 50 of file [readline_autocomp.hpp](#).

Referenced by [execute_line\(\)](#).

34.37.2.3 `char* COMMAND::doc`

Documentation for this function.

Definition at line 55 of file [readline_autocomp.hpp](#).

The documentation for this struct was generated from the following file:

- [stdair/ui/cmdline/readline_autocomp.hpp](#)

34.38 `stdair::ContinuousAttributeLite< T >` Struct Template Reference

Class modeling the distribution of values that can be taken by a continuous attribute.

```
#include <stdair/basic/ContinuousAttributeLite.hpp>
```

Public Types

- `typedef std::map< T, stdair::Probability_T > ContinuousDistribution_T`

Public Member Functions

- `const T getValue (const stdair::Probability_T &iCumulativeProbability) const`
- `const stdair::Probability_T getRemainingProportion (const T &iValue) const`
- `const double getDerivativeValue (const T iKey) const`
- `const T getUpperBound (const T iKey) const`
- `const std::string displayCumulativeDistribution () const`
- `ContinuousAttributeLite (const ContinuousDistribution_T &iValueMap)`
- `ContinuousAttributeLite (const ContinuousAttributeLite &iCAL)`
- `ContinuousAttributeLite & operator= (const ContinuousAttributeLite &iCAL)`
- `virtual ~ContinuousAttributeLite ()`

34.38.1 Detailed Description

```
template<typename T>struct stdair::ContinuousAttributeLite< T >
```

Class modeling the distribution of values that can be taken by a continuous attribute.

Definition at line 26 of file [ContinuousAttributeLite.hpp](#).

34.38.2 Member Typedef Documentation

34.38.2.1 `template<typename T > typedef std::map<T, stdair::Probability_T>
stdair::ContinuousAttributeLite< T >::ContinuousDistribution_T`

Type for the probability mass function.

Definition at line 32 of file [ContinuousAttributeLite.hpp](#).

34.38.3 Constructor & Destructor Documentation

```
34.38.3.1 template<typename T > stdair::ContinuousAttributeLite< T
>::ContinuousAttributeLite ( const ContinuousDistribution_T & iValueMap
) [inline]
```

Constructor.

Definition at line 204 of file [ContinuousAttributeLite.hpp](#).

```
34.38.3.2 template<typename T > stdair::ContinuousAttributeLite< T
>::ContinuousAttributeLite ( const ContinuousAttributeLite< T > & iCAL
) [inline]
```

Copy constructor.

Definition at line 212 of file [ContinuousAttributeLite.hpp](#).

```
34.38.3.3 template<typename T > virtual stdair::ContinuousAttributeLite< T
>::~~ContinuousAttributeLite ( ) [inline, virtual]
```

Destructor.

Definition at line 231 of file [ContinuousAttributeLite.hpp](#).

34.38.4 Member Function Documentation

```
34.38.4.1 template<typename T > const T stdair::ContinuousAttributeLite< T
>::getValue ( const stdair::Probability_T & iCumulativeProbability ) const
[inline]
```

Get value from inverse cumulative distribution.

Definition at line 39 of file [ContinuousAttributeLite.hpp](#).

References [stdair::DictionaryManager::keyToValue\(\)](#), and [stdair::DictionaryManager::valueToKey\(\)](#).

```
34.38.4.2 template<typename T > const stdair::Probability_T
stdair::ContinuousAttributeLite< T >::getRemainingProportion ( const T &
iValue ) const [inline]
```

Get remaining proportion from cumulative distribution.

Definition at line 84 of file [ContinuousAttributeLite.hpp](#).

References [stdair::DictionaryManager::keyToValue\(\)](#).

```
34.38.4.3 template<typename T > const double stdair::ContinuousAttributeLite< T
>::getDerivativeValue ( const T iKey ) const [inline]
```

Get the value of the derivative function in a key point.

Definition at line 131 of file [ContinuousAttributeLite.hpp](#).

References [stdair::DictionaryManager::keyToValue\(\)](#).

34.39 stdair::date_time_element< MIN, MAX > Struct Template Reference 351

34.38.4.4 `template<typename T> const T stdair::ContinuousAttributeLite< T
>::getUpperBound (const T iKey) const [inline]`

Get the upper bound.

Definition at line 163 of file [ContinuousAttributeLite.hpp](#).

34.38.4.5 `template<typename T> const std::string stdair::ContinuousAttributeLite< T
>::displayCumulativeDistribution () const [inline]`

Display cumulative distribution.

Definition at line 182 of file [ContinuousAttributeLite.hpp](#).

References [stdair::DictionaryManager::keyToValue\(\)](#).

34.38.4.6 `template<typename T> ContinuousAttributeLite&
stdair::ContinuousAttributeLite< T>::operator= (const
ContinuousAttributeLite< T> & iCAL) [inline]`

Copy operator.

Definition at line 221 of file [ContinuousAttributeLite.hpp](#).

The documentation for this struct was generated from the following file:

- [stdair/basic/ContinuousAttributeLite.hpp](#)

34.39 stdair::date_time_element< MIN, MAX > Struct Template Reference

```
#include <stdair/basic/BasParserHelperTypes.hpp>
```

Public Member Functions

- [date_time_element](#) ()
- [date_time_element](#) (const [date_time_element](#) &t)
- [date_time_element](#) (int i)
- void [check](#) () const

Public Attributes

- unsigned int [_value](#)

34.39.1 Detailed Description

```
template<int MIN = 0, int MAX = 0>struct stdair::date_time_element< MIN, MAX >
```

Date & time element parser.

Definition at line 23 of file [BasParserHelperTypes.hpp](#).

34.39.2 Constructor & Destructor Documentation

34.39.2.1 `template<int MIN = 0, int MAX = 0> stdair::date_time_element< MIN, MAX >::date_time_element () [inline]`

Default constructor.

Definition at line 28 of file [BasParserHelperTypes.hpp](#).

34.39.2.2 `template<int MIN = 0, int MAX = 0> stdair::date_time_element< MIN, MAX >::date_time_element (const date_time_element< MIN, MAX > & t) [inline]`

Default copy constructor.

Definition at line 30 of file [BasParserHelperTypes.hpp](#).

34.39.2.3 `template<int MIN = 0, int MAX = 0> stdair::date_time_element< MIN, MAX >::date_time_element (int i) [inline]`

Constructor.

Definition at line 32 of file [BasParserHelperTypes.hpp](#).

34.39.3 Member Function Documentation

34.39.3.1 `template<int MIN = 0, int MAX = 0> void stdair::date_time_element< MIN, MAX >::check () const [inline]`

Checker.

Definition at line 34 of file [BasParserHelperTypes.hpp](#).

References [stdair::date_time_element< MIN, MAX >::_value](#).

34.39.4 Member Data Documentation

34.39.4.1 `template<int MIN = 0, int MAX = 0> unsigned int stdair::date_time_element< MIN, MAX >::_value`

Definition at line 24 of file [BasParserHelperTypes.hpp](#).

Referenced by [stdair::date_time_element< MIN, MAX >::check\(\)](#), [stdair::operator*\(\)](#), and [stdair::operator+\(\)](#).

The documentation for this struct was generated from the following file:

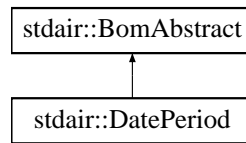
- [stdair/basic/BasParserHelperTypes.hpp](#)

34.40 stdair::DatePeriod Class Reference

Class representing the actual attributes for a fare date-period.

```
#include <stdair/bom/DatePeriod.hpp>
```

Inheritance diagram for stdair::DatePeriod:



Public Types

- typedef [DatePeriodKey](#) [Key_T](#)

Public Member Functions

- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- std::string [toString](#) () const
- const std::string [describeKey](#) () const
- const [Key_T](#) & [getKey](#) () const
- [BomAbstract](#) *const [getParent](#) () const
- const [HolderMap_T](#) & [getHolderMap](#) () const
- const [DatePeriod_T](#) & [getDatePeriod](#) () const
- bool [isDepartureDateValid](#) (const [Date_T](#) &) const

Protected Member Functions

- [DatePeriod](#) (const [Key_T](#) &)
- virtual [~DatePeriod](#) ()

Protected Attributes

- [Key_T](#) [_key](#)
- [BomAbstract](#) * [_parent](#)
- [HolderMap_T](#) [_holderMap](#)

Friends

- class [FacBom](#)
- class [FacBomManager](#)

34.40.1 Detailed Description

Class representing the actual attributes for a fare date-period.

Definition at line 18 of file [DatePeriod.hpp](#).

34.40.2 Member Typedef Documentation

34.40.2.1 typedef DatePeriodKey stdair::DatePeriod::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 27 of file [DatePeriod.hpp](#).

34.40.3 Constructor & Destructor Documentation

34.40.3.1 stdair::DatePeriod::DatePeriod (const Key_T & iKey) [protected]

Main constructor.

Definition at line 28 of file [DatePeriod.cpp](#).

34.40.3.2 stdair::DatePeriod::~~DatePeriod () [protected, virtual]

Destructor.

Definition at line 33 of file [DatePeriod.cpp](#).

34.40.4 Member Function Documentation

34.40.4.1 void stdair::DatePeriod::toStream (std::ostream & ioOut) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Implements [stdair::BomAbstract](#).

Definition at line 36 of file [DatePeriod.hpp](#).

References [toString\(\)](#).

34.40.4.2 void stdair::DatePeriod::fromStream (std::istream & ioIn) [inline, virtual]

Read a Business Object from an input stream.

Parameters

<i>istream&</i> the input stream.

Implements [stdair::BomAbstract](#).

Definition at line 45 of file [DatePeriod.hpp](#).

34.40.4.3 `std::string stdair::DatePeriod::toString () const` [virtual]

Get the serialised version of the Business Object.

Implements [stdair::BomAbstract](#).

Definition at line 37 of file [DatePeriod.cpp](#).

References [describeKey\(\)](#).

Referenced by [toStream\(\)](#).

34.40.4.4 `const std::string stdair::DatePeriod::describeKey () const` [inline]

Get a string describing the key.

Definition at line 56 of file [DatePeriod.hpp](#).

References [_key](#), and [stdair::DatePeriodKey::toString\(\)](#).

Referenced by [toString\(\)](#).

34.40.4.5 `const Key_T& stdair::DatePeriod::getKey () const` [inline]

Get the primary key (date period).

Definition at line 65 of file [DatePeriod.hpp](#).

References [_key](#).

34.40.4.6 `BomAbstract* const stdair::DatePeriod::getParent () const` [inline]

Get a reference on the parent object instance.

Definition at line 72 of file [DatePeriod.hpp](#).

References [_parent](#).

34.40.4.7 `const HolderMap_T& stdair::DatePeriod::getHolderMap () const` [inline]

Get a reference on the children holder.

Definition at line 79 of file [DatePeriod.hpp](#).

References [_holderMap](#).

34.40.4.8 `const DatePeriod_T& stdair::DatePeriod::getDatePeriod () const` [inline]

Get the date period.

Definition at line 86 of file [DatePeriod.hpp](#).

References [_key](#), and [stdair::DatePeriodKey::getDatePeriod\(\)](#).

Referenced by [isDepartureDateValid\(\)](#).

34.40.4.9 `bool stdair::DatePeriod::isDepartureDateValid (const Date_T & iFlightDate) const`

Check if the given departure date is included in the departure period of the segment path.

Definition at line 45 of file [DatePeriod.cpp](#).

References [getDatePeriod\(\)](#).

Referenced by [stdair::BomRetriever::retrieveDatePeriodListFromKey\(\)](#).

34.40.5 Friends And Related Function Documentation

34.40.5.1 `friend class FacBom [friend]`

Definition at line 19 of file [DatePeriod.hpp](#).

34.40.5.2 `friend class FacBomManager [friend]`

Definition at line 20 of file [DatePeriod.hpp](#).

34.40.6 Member Data Documentation

34.40.6.1 `Key_T stdair::DatePeriod::_key [protected]`

Primary key (date period).

Definition at line 125 of file [DatePeriod.hpp](#).

Referenced by [describeKey\(\)](#), [getDatePeriod\(\)](#), and [getKey\(\)](#).

34.40.6.2 `BomAbstract* stdair::DatePeriod::_parent [protected]`

Pointer on the parent class.

Definition at line 130 of file [DatePeriod.hpp](#).

Referenced by [getParent\(\)](#).

34.40.6.3 `HolderMap_T stdair::DatePeriod::_holderMap [protected]`

Map holding the children.

Definition at line 135 of file [DatePeriod.hpp](#).

Referenced by [getHolderMap\(\)](#).

The documentation for this class was generated from the following files:

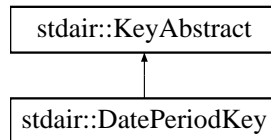
- [stdair/bom/DatePeriod.hpp](#)
- [stdair/bom/DatePeriod.cpp](#)

34.41 stdair::DatePeriodKey Struct Reference

Key of date-period.

```
#include <stdair/bom/DatePeriodKey.hpp>
```

Inheritance diagram for stdair::DatePeriodKey:



Public Member Functions

- [DatePeriodKey](#) (const [DatePeriod_T](#) &)
- [DatePeriodKey](#) (const [DatePeriodKey](#) &)
- [~DatePeriodKey](#) ()
- const [DatePeriod_T](#) & [getDatePeriod](#) () const
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- const std::string [toString](#) () const

34.41.1 Detailed Description

Key of date-period.

Definition at line 14 of file [DatePeriodKey.hpp](#).

34.41.2 Constructor & Destructor Documentation

34.41.2.1 stdair::DatePeriodKey::DatePeriodKey (const [DatePeriod_T](#) & *iDatePeriod*)

Main Constructor.

Definition at line 22 of file [DatePeriodKey.cpp](#).

34.41.2.2 stdair::DatePeriodKey::DatePeriodKey (const [DatePeriodKey](#) & *iKey*)

Copy constructor.

Definition at line 27 of file [DatePeriodKey.cpp](#).

34.41.2.3 stdair::DatePeriodKey::~~DatePeriodKey ()

Destructor.

Definition at line 32 of file [DatePeriodKey.cpp](#).

34.41.3 Member Function Documentation

34.41.3.1 `const DatePeriod_T& stdair::DatePeriodKey::getDatePeriod () const`
[inline]

Get the date period.

Definition at line 32 of file [DatePeriodKey.hpp](#).

Referenced by [stdair::DatePeriod::getDatePeriod\(\)](#).

34.41.3.2 `void stdair::DatePeriodKey::toStream (std::ostream & ioOut) const` [virtual]

Dump a Business Object Key into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 36 of file [DatePeriodKey.cpp](#).

References [toString\(\)](#).

34.41.3.3 `void stdair::DatePeriodKey::fromStream (std::istream & ioIn)` [virtual]

Read a Business Object Key from an input stream.

Parameters

<i>istream&</i> the input stream.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 41 of file [DatePeriodKey.cpp](#).

34.41.3.4 `const std::string stdair::DatePeriodKey::toString () const` [virtual]

Get the serialised version of the Business Object Key. That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 45 of file [DatePeriodKey.cpp](#).

Referenced by [stdair::DatePeriod::describeKey\(\)](#), and [toStream\(\)](#).

The documentation for this struct was generated from the following files:

- [stdair/bom/DatePeriodKey.hpp](#)
- [stdair/bom/DatePeriodKey.cpp](#)

34.42 stdair::DbAbstract Class Reference

```
#include <stdair/dbadaptor/DbAbstract.hpp>
```

Public Member Functions

- virtual [~DbAbstract](#) ()
- virtual void [toStream](#) (std::ostream &ioOut) const
- virtual void [fromStream](#) (std::istream &ioIn)

Protected Member Functions

- [DbAbstract](#) ()

34.42.1 Detailed Description

Base class for the Database Adaptor (DBA) layer.

Definition at line 13 of file [DbAbstract.hpp](#).

34.42.2 Constructor & Destructor Documentation

34.42.2.1 virtual stdair::DbAbstract::~~DbAbstract () [inline, virtual]

Destructor.

Definition at line 17 of file [DbAbstract.hpp](#).

34.42.2.2 stdair::DbAbstract::DbAbstract () [inline, protected]

Protected Default Constructor to ensure this class is abstract.

Definition at line 29 of file [DbAbstract.hpp](#).

34.42.3 Member Function Documentation

34.42.3.1 virtual void stdair::DbAbstract::toStream (std::ostream & *ioOut*) const
[inline, virtual]

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i>	the output stream.
---------------------	--------------------

Definition at line 21 of file [DbAbstract.hpp](#).

Referenced by [operator<<\(\)](#).

34.42.3.2 virtual void stdair::DbAbstract::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters

<i>istream&</i> the input stream.

Definition at line 25 of file [DbAbstract.hpp](#).

Referenced by [operator>>\(\)](#).

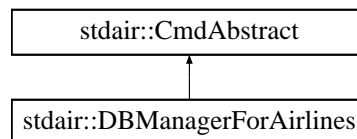
The documentation for this class was generated from the following file:

- [stdair/dbadaptor/DbAbstract.hpp](#)

34.43 stdair::DBManagerForAirlines Class Reference

```
#include <stdair/command/DBManagerForAirlines.hpp>
```

Inheritance diagram for stdair::DBManagerForAirlines:



Static Public Member Functions

- static void [updateAirlineInDB](#) (DBSession_T &, const [AirlineStruct](#) &)
- static bool [retrieveAirline](#) (DBSession_T &, const [AirlineCode_T](#) &, [AirlineStruct](#) &)
- static void [prepareSelectStatement](#) (DBSession_T &, [DBRequestStatement_T](#) &, [AirlineStruct](#) &)
- static bool [iterateOnStatement](#) ([DBRequestStatement_T](#) &, [AirlineStruct](#) &)

34.43.1 Detailed Description

Class building the Business Object Model (BOM) from data retrieved from the database.

Definition at line 18 of file [DBManagerForAirlines.hpp](#).

34.43.2 Member Function Documentation

34.43.2.1 void stdair::DBManagerForAirlines::updateAirlineInDB (DBSession_T & ioSociSession, const AirlineStruct & iAirline) [static]

Update the fields of the database row corresponding to the given BOM object. DBSession_T & [AirlineStruct](#) & .

Definition at line 99 of file [DBManagerForAirlines.cpp](#).

References [stdair::AirlineStruct::getAirlineCode\(\)](#), and [stdair::AirlineStruct::getAirlineName\(\)](#).

34.43.2.2 bool stdair::DBManagerForAirlines::retrieveAirline (DBSession_T & ioSociSession, const AirlineCode_T & iAirlineCode, AirlineStruct & ioAirline) [static]

Retrieve, from the (MySQL) database, the row corresponding to the given BOM code, and fill the given BOM object with that retrieved data. DBSession_T & const AirlineCode_T & [AirlineStruct](#) & .

Definition at line 134 of file [DBManagerForAirlines.cpp](#).

References [iterateOnStatement\(\)](#).

34.43.2.3 void stdair::DBManagerForAirlines::prepareSelectStatement (DBSession_T & ioSociSession, DBRequestStatement_T & ioSelectStatement, AirlineStruct & ioAirline) [static]

Prepare (parse and put in cache) the SQL statement. DBSession_T & DBRequestStatement_T & [AirlineStruct](#) & .

Definition at line 26 of file [DBManagerForAirlines.cpp](#).

34.43.2.4 bool stdair::DBManagerForAirlines::iterateOnStatement (DBRequestStatement_T & ioStatement, AirlineStruct & ioAirline) [static]

Iterate on the SQL statement.

The SQL has to be already prepared. DBRequestStatement_T & [AirlineStruct](#) & .

Definition at line 82 of file [DBManagerForAirlines.cpp](#).

Referenced by [retrieveAirline\(\)](#).

The documentation for this class was generated from the following files:

- [stdair/command/DBManagerForAirlines.hpp](#)
- [stdair/command/DBManagerForAirlines.cpp](#)

34.44 stdair::DBSessionManager Class Reference

```
#include <stdair/service/DBSessionManager.hpp>
```

Public Member Functions

- [DBSession_T & getDBSession \(\)](#) const

Static Public Member Functions

- static [DBSessionManager](#) & [instance](#) ()

Friends

- class [FacSupervisor](#)
- class [STDAIR_Service](#)

34.44.1 Detailed Description

Class holding the database session.

Note that the database access is handled by the SOCI library.

Definition at line 17 of file [DBSessionManager.hpp](#).

34.44.2 Member Function Documentation

34.44.2.1 [DBSessionManager](#) & [stdair::DBSessionManager::instance](#) () [static]

Return the static [DBSessionManager](#) instance.

Definition at line 82 of file [DBSessionManager.cpp](#).

34.44.2.2 [DBSession_T](#) & [stdair::DBSessionManager::getDBSession](#) () const

Retrieve the database session handler, held by the static instance of [DBSessionManager](#).

Definition at line 92 of file [DBSessionManager.cpp](#).

34.44.3 Friends And Related Function Documentation

34.44.3.1 friend class [FacSupervisor](#) [friend]

Definition at line 19 of file [DBSessionManager.hpp](#).

34.44.3.2 friend class [STDAIR_Service](#) [friend]

Definition at line 20 of file [DBSessionManager.hpp](#).

The documentation for this class was generated from the following files:

- [stdair/service/DBSessionManager.hpp](#)
- [stdair/service/DBSessionManager.cpp](#)

34.45 stdair::DefaultDCPList Struct Reference

```
#include <stdair/basic/BasConst_Inventory.hpp>
```

Static Public Member Functions

- static [DCPList_T](#) init ()

34.45.1 Detailed Description

Definition at line 94 of file [BasConst_Inventory.hpp](#).

34.45.2 Member Function Documentation

34.45.2.1 [DCPList_T](#) stdair::DefaultDCPList::init () [static]

Definition at line 470 of file [BasConst.cpp](#).

The documentation for this struct was generated from the following files:

- stdair/basic/[BasConst_Inventory.hpp](#)
- stdair/basic/[BasConst.cpp](#)

34.46 stdair::DefaultDtdFratMap Struct Reference

```
#include <stdair/basic/BasConst_Inventory.hpp>
```

Static Public Member Functions

- static [DTDFratMap_T](#) init ()

34.46.1 Detailed Description

Definition at line 98 of file [BasConst_Inventory.hpp](#).

34.46.2 Member Function Documentation

34.46.2.1 [DTDFratMap_T](#) stdair::DefaultDtdFratMap::init () [static]

Definition at line 489 of file [BasConst.cpp](#).

The documentation for this struct was generated from the following files:

- stdair/basic/[BasConst_Inventory.hpp](#)
- stdair/basic/[BasConst.cpp](#)

34.47 stdair::DefaultDtdProbMap Struct Reference

```
#include <stdair/basic/BasConst_Inventory.hpp>
```

Static Public Member Functions

- static [DTDProbMap_T](#) [init](#) ()

34.47.1 Detailed Description

Definition at line 102 of file [BasConst_Inventory.hpp](#).

34.47.2 Member Function Documentation

34.47.2.1 [DTDProbMap_T](#) [stdair::DefaultDtdProbMap::init](#) () [static]

Definition at line 506 of file [BasConst.cpp](#).

The documentation for this struct was generated from the following files:

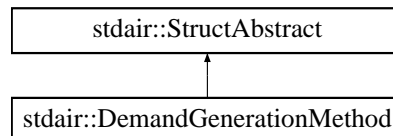
- [stdair/basic/BasConst_Inventory.hpp](#)
- [stdair/basic/BasConst.cpp](#)

34.48 stdair::DemandGenerationMethod Struct Reference

Enumeration of demand (booking request) generation methods.

```
#include <stdair/basic/DemandGenerationMethod.hpp>
```

Inheritance diagram for [stdair::DemandGenerationMethod](#):



Public Types

- enum [EN_DemandGenerationMethod](#) { [POI_PRO](#) = 0, [STA_ORD](#), [LAST_VALUE](#) }

Public Member Functions

- [EN_DemandGenerationMethod](#) [getMethod](#) () const
- char [getMethodAsChar](#) () const
- std::string [getMethodAsString](#) () const
- const std::string [describe](#) () const
- bool [operator==](#) (const [EN_DemandGenerationMethod](#) &) const
- [DemandGenerationMethod](#) (const [EN_DemandGenerationMethod](#) &)

- [DemandGenerationMethod](#) (const char iMethod)
- [DemandGenerationMethod](#) (const std::string &iMethod)
- [DemandGenerationMethod](#) (const [DemandGenerationMethod](#) &)
- void [toStream](#) (std::ostream &ioOut) const
- virtual void [fromStream](#) (std::istream &ioIn)

Static Public Member Functions

- static const std::string & [getLabel](#) (const [EN_DemandGenerationMethod](#) &)
- static [EN_DemandGenerationMethod](#) [getMethod](#) (const char)
- static char [getMethodLabel](#) (const [EN_DemandGenerationMethod](#) &)
- static std::string [getMethodLabelAsString](#) (const [EN_DemandGenerationMethod](#) &)
- static std::string [describeLabels](#) ()

34.48.1 Detailed Description

Enumeration of demand (booking request) generation methods.

Definition at line 17 of file [DemandGenerationMethod.hpp](#).

34.48.2 Member Enumeration Documentation

34.48.2.1 enum stdair::DemandGenerationMethod::EN_DemandGenerationMethod

Enumerator:

POI_PRO
STA_ORD
LAST_VALUE

Definition at line 19 of file [DemandGenerationMethod.hpp](#).

34.48.3 Constructor & Destructor Documentation

34.48.3.1 stdair::DemandGenerationMethod::DemandGenerationMethod (const [EN_DemandGenerationMethod](#) & *iDemandGenerationMethod*)

Main constructor.

Definition at line 34 of file [DemandGenerationMethod.cpp](#).

34.48.3.2 stdair::DemandGenerationMethod::DemandGenerationMethod (const char *iMethod*)

Alternative constructor.

Definition at line 62 of file [DemandGenerationMethod.cpp](#).

34.48.3.3 `stdair::DemandGenerationMethod::DemandGenerationMethod (const std::string & iMethod)`

Alternative constructor.

Definition at line 68 of file [DemandGenerationMethod.cpp](#).

References [getMethod\(\)](#).

34.48.3.4 `stdair::DemandGenerationMethod::DemandGenerationMethod (const DemandGenerationMethod & iDemandGenerationMethod)`

Default copy constructor.

Definition at line 28 of file [DemandGenerationMethod.cpp](#).

34.48.4 Member Function Documentation

34.48.4.1 `const std::string & stdair::DemandGenerationMethod::getLabel (const EN_DemandGenerationMethod & iMethod) [static]`

Get the label as a string (e.g., "PoissonProcess" or "StatisticsOrder").

Definition at line 78 of file [DemandGenerationMethod.cpp](#).

34.48.4.2 `DemandGenerationMethod::EN_DemandGenerationMethod stdair::DemandGenerationMethod::getMethod (const char iMethodChar) [static]`

Get the method value from parsing a single char (e.g., 'P' or 'S').

Definition at line 40 of file [DemandGenerationMethod.cpp](#).

References [describeLabels\(\)](#), [LAST_VALUE](#), [POI_PRO](#), and [STA_ORD](#).

34.48.4.3 `char stdair::DemandGenerationMethod::getMethodLabel (const EN_DemandGenerationMethod & iMethod) [static]`

Get the label as a single char (e.g., 'P' or 'S').

Definition at line 84 of file [DemandGenerationMethod.cpp](#).

34.48.4.4 `std::string stdair::DemandGenerationMethod::getMethodLabelAsString (const EN_DemandGenerationMethod & iMethod) [static]`

Get the label as a string of a single char (e.g., "P" or "S").

Definition at line 90 of file [DemandGenerationMethod.cpp](#).

34.48.4.5 `std::string stdair::DemandGenerationMethod::describeLabels () [static]`

List the labels.

Definition at line 97 of file [DemandGenerationMethod.cpp](#).

References [LAST_VALUE](#).

Referenced by [getMethod\(\)](#).

34.48.4.6 DemandGenerationMethod::EN_DemandGenerationMethod
stdair::DemandGenerationMethod::getMethod () const

Get the enumerated value.

Definition at line 110 of file [DemandGenerationMethod.cpp](#).

Referenced by [DemandGenerationMethod\(\)](#).

34.48.4.7 char stdair::DemandGenerationMethod::getMethodAsChar () const

Get the enumerated value as a short string (e.g., 'P' or 'S').

Definition at line 115 of file [DemandGenerationMethod.cpp](#).

34.48.4.8 std::string stdair::DemandGenerationMethod::getMethodAsString () const

Get the enumerated value as a short string (e.g., "P" or "S").

Definition at line 121 of file [DemandGenerationMethod.cpp](#).

34.48.4.9 const std::string stdair::DemandGenerationMethod::describe () const
[virtual]

Give a description of the structure (e.g., "PoissonProcess" or "StatisticsOrder").

Implements [stdair::StructAbstract](#).

Definition at line 128 of file [DemandGenerationMethod.cpp](#).

34.48.4.10 bool stdair::DemandGenerationMethod::operator== (const
EN_DemandGenerationMethod & iMethod) const

Comparison operator.

Definition at line 136 of file [DemandGenerationMethod.cpp](#).

34.48.4.11 void stdair::StructAbstract::toStream (std::ostream & ioOut) const **[inline,**
inherited]

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Reimplemented in [stdair::YieldRange](#), [stdair::AirlineStruct](#), [stdair::BookingRequestStruct](#), [stdair::CancellationStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::TravelSolutionStruct](#), and [stdair::VirtualClassStruct](#).

Definition at line 29 of file [StructAbstract.hpp](#).

References [stdair::StructAbstract::describe\(\)](#).

Referenced by [operator<<\(\)](#).

34.48.4.12 `virtual void stdair::StructAbstract::fromStream (std::istream & ioIn) [inline, virtual, inherited]`

Read a Business Object from an input stream.

Parameters

<i>istream&</i> the input stream.

Reimplemented in [stdair::ProgressStatusSet](#), [stdair::YieldRange](#), [stdair::AirlineStruct](#), [stdair::BookingRequestStruct](#), [stdair::CancellationStruct](#), [stdair::EventStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::TravelSolutionStruct](#), and [stdair::VirtualClassStruct](#).

Definition at line 38 of file [StructAbstract.hpp](#).

Referenced by [operator>>\(\)](#).

The documentation for this struct was generated from the following files:

- [stdair/basic/DemandGenerationMethod.hpp](#)
- [stdair/basic/DemandGenerationMethod.cpp](#)

34.49 stdair::DictionaryManager Class Reference

Class wrapper of dictionary business methods.

```
#include <stdair/basic/DictionaryManager.hpp>
```

Static Public Member Functions

- static const [stdair::Probability_T](#) [keyToValue](#) (const [DictionaryKey_T](#))
- static const [DictionaryKey_T](#) [valueToKey](#) (const [stdair::Probability_T](#))

34.49.1 Detailed Description

Class wrapper of dictionary business methods.

Definition at line 22 of file [DictionaryManager.hpp](#).

34.49.2 Member Function Documentation

34.49.2.1 `const stdair::Probability_T stdair::DictionaryManager::keyToValue (const DictionaryKey_T iKey) [static]`

Convert from key to value.

Definition at line 12 of file [DictionaryManager.cpp](#).

References [stdair::DEFAULT_NUMBER_OF_SUBDIVISIONS](#).

Referenced by [stdair::ContinuousAttributeLite< T >::displayCumulativeDistribution\(\)](#), [stdair::ContinuousAttributeLite< T >::getDerivativeValue\(\)](#), [stdair::ContinuousAttributeLite< T >::getRemainingProportion\(\)](#), and [stdair::ContinuousAttributeLite< T >::getValue\(\)](#).

34.49.2.2 `const DictionaryKey_T stdair::DictionaryManager::valueToKey (const stdair::Probability_T iValue) [static]`

Convert from value to key.

Definition at line 21 of file [DictionaryManager.cpp](#).

References [stdair::DEFAULT_NUMBER_OF_SUBDIVISIONS](#).

Referenced by [stdair::ContinuousAttributeLite< T >::getValue\(\)](#).

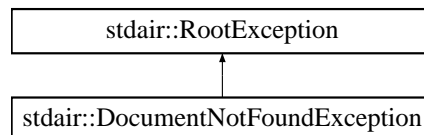
The documentation for this class was generated from the following files:

- [stdair/basic/DictionaryManager.hpp](#)
- [stdair/basic/DictionaryManager.cpp](#)

34.50 stdair::DocumentNotFoundException Class Reference

```
#include <stdair/stdair_exceptions.hpp>
```

Inheritance diagram for `stdair::DocumentNotFoundException`:



Public Member Functions

- [DocumentNotFoundException](#) (const std::string &iWhat)
- const char * [what](#) () const throw ()

Protected Attributes

- std::string [_what](#)

34.50.1 Detailed Description

Document not found.

Definition at line 104 of file [stdair_exceptions.hpp](#).

34.50.2 Constructor & Destructor Documentation

34.50.2.1 `stdair::DocumentNotFoundException::DocumentNotFoundException (const std::string & iWhat)` `[inline]`

Constructor.

Definition at line 107 of file [stdair_exceptions.hpp](#).

34.50.3 Member Function Documentation

34.50.3.1 `const char* stdair::RootException::what () const throw ()` `[inline, inherited]`

Give the details of the exception.

Definition at line 38 of file [stdair_exceptions.hpp](#).

References [stdair::RootException::_what](#).

34.50.4 Member Data Documentation

34.50.4.1 `std::string stdair::RootException::_what` `[protected, inherited]`

Details for the exception.

Definition at line 46 of file [stdair_exceptions.hpp](#).

Referenced by [stdair::RootException::what\(\)](#).

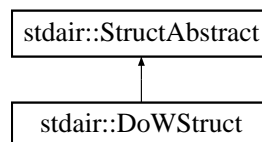
The documentation for this class was generated from the following file:

- [stdair/stdair_exceptions.hpp](#)

34.51 stdair::DoWStruct Struct Reference

```
#include <stdair/bom/DoWStruct.hpp>
```

Inheritance diagram for `stdair::DoWStruct`:



Public Types

- `typedef std::vector< bool >` [BooleanList_T](#)

Public Member Functions

- bool [getDayOfWeek](#) (const unsigned short i) const
- bool [getStandardDayOfWeek](#) (const unsigned short i) const
- void [setDayOfWeek](#) (const unsigned short, const bool)
- const std::string [describe](#) () const
- const std::string [describeShort](#) () const
- [DoWStruct](#) [shift](#) (const long &) const
- [DoWStruct](#) [intersection](#) (const [DoWStruct](#) &) const
- const bool [isValid](#) () const
- [DoWStruct](#) (const std::string &iDowString)
- [DoWStruct](#) ()
- [DoWStruct](#) (const [DoWStruct](#) &)
- [~DoWStruct](#) ()
- void [toStream](#) (std::ostream &ioOut) const
- virtual void [fromStream](#) (std::istream &ioIn)

34.51.1 Detailed Description

Define a Day Of the Week (DoW) sequence.

For instance, 1..11.1 means that the period is active on Mon., Thu., Fri. and Sun.

Definition at line 18 of file [DoWStruct.hpp](#).

34.51.2 Member Typedef Documentation**34.51.2.1 typedef std::vector<bool> stdair::DoWStruct::BooleanList_T**

Define the bit set representing the DoW.

Definition at line 21 of file [DoWStruct.hpp](#).

34.51.3 Constructor & Destructor Documentation**34.51.3.1 stdair::DoWStruct::DoWStruct (const std::string & iDowString)**

Constructor from a given bit set (e.g., "0000011" for the week-ends).

Definition at line 21 of file [DoWStruct.cpp](#).

34.51.3.2 stdair::DoWStruct::DoWStruct ()

Default constructors.

Definition at line 14 of file [DoWStruct.cpp](#).

34.51.3.3 stdair::DoWStruct::DoWStruct (const DoWStruct & iDowStruct)

Definition at line 34 of file [DoWStruct.cpp](#).

34.51.3.4 `stdair::DoWStruct::~~DoWStruct () [inline]`

Default destructor.

Definition at line 63 of file [DoWStruct.hpp](#).

34.51.4 Member Function Documentation

34.51.4.1 `bool stdair::DoWStruct::getDayOfWeek (const unsigned short i) const`

Get the *i*-th day of the week (Monday being the first one).

Definition at line 66 of file [DoWStruct.cpp](#).

Referenced by [intersection\(\)](#), and [isValid\(\)](#).

34.51.4.2 `bool stdair::DoWStruct::getStandardDayOfWeek (const unsigned short i) const`

Get the *i*-th day of the week (Sunday being the first one).

Definition at line 71 of file [DoWStruct.cpp](#).

34.51.4.3 `void stdair::DoWStruct::setDayOfWeek (const unsigned short i, const bool iBool)`

Set the new value for the *i*-th day-of-week.

Definition at line 82 of file [DoWStruct.cpp](#).

Referenced by [intersection\(\)](#), and [shift\(\)](#).

34.51.4.4 `const std::string stdair::DoWStruct::describe () const [virtual]`

Display explicitly (e.g., "Mon.Tue.Wed.Thu.Fri.").

Implements [stdair::StructAbstract](#).

Definition at line 52 of file [DoWStruct.cpp](#).

References [stdair::DOW_STR](#).

Referenced by [stdair::PeriodStruct::describe\(\)](#).

34.51.4.5 `const std::string stdair::DoWStruct::describeShort () const`

Display as a bit set (e.g., "1111100").

Definition at line 40 of file [DoWStruct.cpp](#).

Referenced by [stdair::PeriodStruct::describeShort\(\)](#).

34.51.4.6 `DoWStruct stdair::DoWStruct::shift (const long & iNbOfDays) const`

Build a new DoW struct by shifting the current DoW by a given number.

Definition at line 88 of file [DoWStruct.cpp](#).

References [stdair::DEFAULT_DOW_STRING](#), and [setDayOfWeek\(\)](#).

Referenced by [stdair::PeriodStruct::addDateOffset\(\)](#).

34.51.4.7 DoWStruct [stdair::DoWStruct::intersection](#) ([const DoWStruct & iDoW](#)) const

Build a new DoW struct by intersecting two DoW structs.

Definition at line 104 of file [DoWStruct.cpp](#).

References [stdair::DEFAULT_DOW_STRING](#), [getDayOfWeek\(\)](#), and [setDayOfWeek\(\)](#).

Referenced by [stdair::PeriodStruct::intersection\(\)](#).

34.51.4.8 const bool [stdair::DoWStruct::isValid](#) () const

Return if the DoW struct is valid (i.e., has at least one "true").

Definition at line 117 of file [DoWStruct.cpp](#).

References [getDayOfWeek\(\)](#).

Referenced by [stdair::PeriodStruct::isValid\(\)](#).

34.51.4.9 void [stdair::StructAbstract::toStream](#) ([std::ostream & ioOut](#)) const [inline, inherited]

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Reimplemented in [stdair::YieldRange](#), [stdair::AirlineStruct](#), [stdair::BookingRequestStruct](#), [stdair::CancellationStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::TravelSolutionStruct](#), and [stdair::VirtualClassStruct](#).

Definition at line 29 of file [StructAbstract.hpp](#).

References [stdair::StructAbstract::describe\(\)](#).

Referenced by [operator<<\(\)](#).

34.51.4.10 virtual void [stdair::StructAbstract::fromStream](#) ([std::istream & iIn](#)) [inline, virtual, inherited]

Read a Business Object from an input stream.

Parameters

<i>istream&</i> the input stream.

Reimplemented in [stdair::ProgressStatusSet](#), [stdair::YieldRange](#), [stdair::AirlineStruct](#), [stdair::BookingRequestStruct](#), [stdair::CancellationStruct](#), [stdair::EventStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::TravelSolutionStruct](#), and [stdair::VirtualClassStruct](#).

Definition at line 38 of file [StructAbstract.hpp](#).

Referenced by [operator>>\(\)](#).

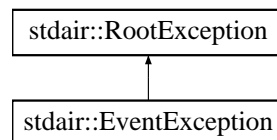
The documentation for this struct was generated from the following files:

- [stdair/bom/DoWStruct.hpp](#)
- [stdair/bom/DoWStruct.cpp](#)

34.52 stdair::EventException Class Reference

```
#include <stdair/stdair_exceptions.hpp>
```

Inheritance diagram for stdair::EventException:



Public Member Functions

- [EventException](#) (const std::string &iWhat)
- const char * [what](#) () const throw ()

Protected Attributes

- std::string [_what](#)

34.52.1 Detailed Description

Event.

Definition at line 196 of file [stdair_exceptions.hpp](#).

34.52.2 Constructor & Destructor Documentation

34.52.2.1 [stdair::EventException::EventException \(const std::string &iWhat \)](#) [inline]

Constructor.

Definition at line 199 of file [stdair_exceptions.hpp](#).

34.52.3 Member Function Documentation

34.52.3.1 `const char* stdair::RootException::what () const throw ()` [inline, inherited]

Give the details of the exception.

Definition at line 38 of file [stdair_exceptions.hpp](#).

References [stdair::RootException::_what](#).

34.52.4 Member Data Documentation

34.52.4.1 `std::string stdair::RootException::_what` [protected, inherited]

Details for the exception.

Definition at line 46 of file [stdair_exceptions.hpp](#).

Referenced by [stdair::RootException::what\(\)](#).

The documentation for this class was generated from the following file:

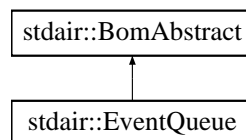
- [stdair/stdair_exceptions.hpp](#)

34.53 stdair::EventQueue Class Reference

Class holding event structures.

```
#include <stdair/bom/EventQueue.hpp>
```

Inheritance diagram for `stdair::EventQueue`:



Public Types

- typedef [EventQueueKey](#) [Key_T](#)
- typedef `std::map< EventType::EN_EventType, ProgressStatus >` [ProgressStatusMap_T](#)

Public Member Functions

- const [Key_T](#) & [getKey](#) () const
- [BomAbstract](#) *const [getParent](#) () const
- const [HolderMap_T](#) & [getHolderMap](#) () const
- const [ProgressStatus](#) & [getStatus](#) () const

- `const Count_T & getCurrentNbOfEvents () const`
- `const Count_T & getExpectedTotalNbOfEvents () const`
- `const Count_T & getActualTotalNbOfEvents () const`
- `ProgressStatus getStatus (const EventType::EN_EventType &) const`
- `const Count_T & getCurrentNbOfEvents (const EventType::EN_EventType &) const`
- `const Count_T & getExpectedTotalNbOfEvents (const EventType::EN_EventType &) const`
- `const Count_T & getActualTotalNbOfEvents (const EventType::EN_EventType &) const`
- `void setStatus (const ProgressStatus &iProgressStatus)`
- `void setStatus (const Count_T &iCurrentNbOfEvents, const Count_T &iExpectedTotalNbOfEvents, const Count_T &iActualTotalNbOfEvents)`
- `void setStatus (const Count_T &iCurrentNbOfEvents, const Count_T &iActualTotalNbOfEvents)`
- `void setCurrentNbOfEvents (const Count_T &iCurrentNbOfEvents)`
- `void setExpectedTotalNbOfEvents (const Count_T &iExpectedTotalNbOfEvents)`
- `void setActualTotalNbOfEvents (const Count_T &iActualTotalNbOfEvents)`
- `void setStatus (const EventType::EN_EventType &iType, const ProgressStatus &iProgressStatus)`
- `void toStream (std::ostream &ioOut) const`
- `void fromStream (std::istream &ioIn)`
- `std::string toString () const`
- `const std::string describeKey () const`
- `std::string display () const`
- `void reset ()`
- `ProgressStatusSet popEvent (EventStruct &)`
- `bool addEvent (EventStruct &)`
- `bool isQueueDone () const`
- `void addStatus (const EventType::EN_EventType &, const NbOfRequests_T &iExpectedTotalNbOfEvents)`
- `void updateStatus (const EventType::EN_EventType &, const ProgressStatus &iProgressStatus)`
- `void updateStatus (const EventType::EN_EventType &, const NbOfEvents_T &iActualTotalNbOfEvents)`
- `ProgressPercentage_T calculateProgress () const`
- `ProgressPercentage_T calculateProgress (const EventType::EN_EventType &) const`
- `Count_T getQueueSize () const`
- `bool isQueueEmpty () const`

Protected Member Functions

- `EventQueue (const Key_T &)`
- `EventQueue (const EventQueue &)`
- `~EventQueue ()`

Protected Attributes

- [Key_T _key](#)
- [BomAbstract * _parent](#)
- [HolderMap_T _holderMap](#)
- [EventList_T _eventList](#)
- [ProgressStatus _progressStatus](#)
- [ProgressStatusMap_T _progressStatusMap](#)

Friends

- class [FacBom](#)
- class [FacBomManager](#)

34.53.1 Detailed Description

Class holding event structures.

Event types may be:

- booking requests,
- optimisation notifications,
- (simulation) break point,
- schedule changes.

The event content would be, respectively:

- a demand stream (generating booking requests),
- a DCP rule (generation optimisation notifications),
- a break point rule (generating simulation break points),
- a schedule update (generating schedule changes).

The [EventQueue](#) object keeps track of the simulation progress, overall and broken down (independently) both by event type and by content key. Following is a full example:

- Break down by event type:
 - Booking request: 9 events out of {expected: 20, actual: 20}
 - Optimisation notification: 7 events out of {expected: 32, actual: 32}
- Break down by content key:
 - "SIN-BKK" demand stream: 5 events out of {expected: 10, actual: 11}
 - "SIN-NRT" demand stream: 4 events out of {expected: 10, actual: 9}

- "SQ 12" DCP rule: 2 events out of {expected: 16, actual: 16}
- "SQ 25" DCP rule: 5 events out of {expected: 16, actual: 16}
- Overall status: 16 events out of {expected: 52, actual: 52}

Definition at line 59 of file [EventQueue.hpp](#).

34.53.2 Member Typedef Documentation

34.53.2.1 `typedef EventQueueKey stdair::EventQueue::Key_T`

Definition allowing to retrieve the associated BOM key type.

Definition at line 68 of file [EventQueue.hpp](#).

34.53.2.2 `typedef std::map<EventType::EN_EventType, ProgressStatus> stdair::EventQueue::ProgressStatusMap_T`

Definition of the (STL) map of [ProgressStatus](#) structures, one for each event type (e.g., booking request, optimisation notification).

Definition at line 76 of file [EventQueue.hpp](#).

34.53.3 Constructor & Destructor Documentation

34.53.3.1 `stdair::EventQueue::EventQueue (const Key_T & iKey) [protected]`

Constructor.

Definition at line 23 of file [EventQueue.cpp](#).

34.53.3.2 `stdair::EventQueue::EventQueue (const EventQueue & iEventQueue) [protected]`

Default copy constructor.

Definition at line 30 of file [EventQueue.cpp](#).

34.53.3.3 `stdair::EventQueue::~~EventQueue () [protected]`

Destructor.

Definition at line 38 of file [EventQueue.cpp](#).

References [_eventList](#).

34.53.4 Member Function Documentation

34.53.4.1 `const Key_T& stdair::EventQueue::getKey () const [inline]`

Get the event queue key.

Definition at line 82 of file [EventQueue.hpp](#).

References [_key](#).

34.53.4.2 BomAbstract* `const stdair::EventQueue::getParent () const` `[inline]`

Get the parent object.

Definition at line 87 of file [EventQueue.hpp](#).

References [_parent](#).

34.53.4.3 const HolderMap_T& `stdair::EventQueue::getHolderMap () const` `[inline]`

Get the map of children holders.

Definition at line 92 of file [EventQueue.hpp](#).

References [_holderMap](#).

34.53.4.4 const ProgressStatus& `stdair::EventQueue::getStatus () const` `[inline]`

Get the overall progress status (for the whole event queue).

Definition at line 97 of file [EventQueue.hpp](#).

References [_progressStatus](#).

Referenced by [popEvent\(\)](#).

34.53.4.5 const Count_T& `stdair::EventQueue::getCurrentNbOfEvents () const`
`[inline]`

Get the current number of events (for the whole event queue).

Definition at line 101 of file [EventQueue.hpp](#).

References [_progressStatus](#), and [stdair::ProgressStatus::getCurrentNb\(\)](#).

34.53.4.6 const Count_T& `stdair::EventQueue::getExpectedTotalNbOfEvents () const`
`[inline]`

Get the expected total number of events (for the whole event queue).

Definition at line 105 of file [EventQueue.hpp](#).

References [_progressStatus](#), and [stdair::ProgressStatus::getExpectedNb\(\)](#).

Referenced by [stdair::STDAIR_Service::getExpectedTotalNumberOfEventsToBeGenerated\(\)](#).

34.53.4.7 const Count_T& `stdair::EventQueue::getActualTotalNbOfEvents () const`
`[inline]`

Get the actual total number of events (for the whole event queue).

Definition at line 109 of file [EventQueue.hpp](#).

References [_progressStatus](#), and [stdair::ProgressStatus::getActualNb\(\)](#).

Referenced by [stdair::STDAIR_Service::getActualTotalNumberOfEventsToBeGenerated\(\)](#).

34.53.4.8 **ProgressStatus** stdair::EventQueue::getStatus (const EventType::EN_EventType & iType) const

Get the progress status for the given event type (e.g., booking request, optimisation notification, schedule change, break point).

Definition at line 257 of file [EventQueue.cpp](#).

References [_progressStatusMap](#).

34.53.4.9 const Count_T & stdair::EventQueue::getCurrentNbOfEvents (const EventType::EN_EventType & iType) const

Get the current number of events for the given event type.

Definition at line 97 of file [EventQueue.cpp](#).

References [_progressStatusMap](#), [display\(\)](#), [stdair::ProgressStatus::getCurrentNb\(\)](#), and [STDAIR_LOG_ERROR](#).

34.53.4.10 const Count_T & stdair::EventQueue::getExpectedTotalNbOfEvents (const EventType::EN_EventType & iType) const

Get the expected total number of events for the given event type.

Definition at line 116 of file [EventQueue.cpp](#).

References [_progressStatusMap](#), [display\(\)](#), [stdair::ProgressStatus::getExpectedNb\(\)](#), and [STDAIR_LOG_ERROR](#).

34.53.4.11 const Count_T & stdair::EventQueue::getActualTotalNbOfEvents (const EventType::EN_EventType & iType) const

Get the actual total number of events for the given event type.

Definition at line 138 of file [EventQueue.cpp](#).

References [_progressStatusMap](#), [display\(\)](#), [stdair::ProgressStatus::getActualNb\(\)](#), and [STDAIR_LOG_ERROR](#).

34.53.4.12 void stdair::EventQueue::setStatus (const ProgressStatus & iProgressStatus) [inline]

Set/update the progress status.

Definition at line 131 of file [EventQueue.hpp](#).

References [_progressStatus](#).

Referenced by [popEvent\(\)](#).

34.53.4.13 void stdair::EventQueue::setStatus (const Count_T & iCurrentNbOfEvents, const Count_T & iExpectedTotalNbOfEvents, const Count_T & iActualTotalNbOfEvents) [inline]

Set/update the progress status.

Definition at line 135 of file [EventQueue.hpp](#).

References [_progressStatus](#), [stdair::ProgressStatus::setActualNb\(\)](#), [stdair::ProgressStatus::setCurrentNb\(\)](#), and [stdair::ProgressStatus::setExpectedNb\(\)](#).

34.53.4.14 `void stdair::EventQueue::setStatus (const Count_T & iCurrentNbOfEvents, const Count_T & iActualTotalNbOfEvents) [inline]`

Set/update the progress status.

Definition at line 143 of file [EventQueue.hpp](#).

References [_progressStatus](#), [stdair::ProgressStatus::setActualNb\(\)](#), and [stdair::ProgressStatus::setCurrentNb\(\)](#).

34.53.4.15 `void stdair::EventQueue::setCurrentNbOfEvents (const Count_T & iCurrentNbOfEvents) [inline]`

Set the current number of events (for the whole event queue).

Definition at line 149 of file [EventQueue.hpp](#).

References [_progressStatus](#), and [stdair::ProgressStatus::setCurrentNb\(\)](#).

34.53.4.16 `void stdair::EventQueue::setExpectedTotalNbOfEvents (const Count_T & iExpectedTotalNbOfEvents) [inline]`

Set the expected total number of events (for the whole event queue).

Definition at line 153 of file [EventQueue.hpp](#).

References [_progressStatus](#), and [stdair::ProgressStatus::setExpectedNb\(\)](#).

34.53.4.17 `void stdair::EventQueue::setActualTotalNbOfEvents (const Count_T & iActualTotalNbOfEvents) [inline]`

Set the actual total number of events (for the whole event queue).

Definition at line 157 of file [EventQueue.hpp](#).

References [_progressStatus](#), and [stdair::ProgressStatus::setActualNb\(\)](#).

34.53.4.18 `void stdair::EventQueue::setStatus (const EventType::EN_EventType & iType, const ProgressStatus & iProgressStatus)`

Set the progress status for the given event type (e.g., booking request, optimisation notification, schedule change, break point).

Definition at line 241 of file [EventQueue.cpp](#).

References [_progressStatusMap](#).

34.53.4.19 `void stdair::EventQueue::toStream (std::ostream & ioOut) const [inline, virtual]`

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Implements [stdair::BomAbstract](#).

Definition at line 176 of file [EventQueue.hpp](#).

References [toString\(\)](#).

34.53.4.20 `void stdair::EventQueue::fromStream (std::istream & ioln) [inline, virtual]`

Read a Business Object from an input stream.

Parameters

<i>istream&</i> the input stream.

Implements [stdair::BomAbstract](#).

Definition at line 185 of file [EventQueue.hpp](#).

34.53.4.21 `std::string stdair::EventQueue::toString () const [virtual]`

Get the serialised version of the Business Object.

Implements [stdair::BomAbstract](#).

Definition at line 43 of file [EventQueue.cpp](#).

References [_eventList](#), [_progressStatus](#), [stdair::ProgressStatus::getActualNb\(\)](#), [stdair::ProgressStatus::getCurrentNb\(\)](#), and [stdair::ProgressStatus::getExpectedNb\(\)](#).

Referenced by [display\(\)](#), [toStream\(\)](#), and [updateStatus\(\)](#).

34.53.4.22 `const std::string stdair::EventQueue::describeKey () const [inline]`

Get a string describing the key.

Definition at line 196 of file [EventQueue.hpp](#).

References [_key](#), and [stdair::EventQueueKey::toString\(\)](#).

34.53.4.23 `std::string stdair::EventQueue::display () const`

Definition at line 53 of file [EventQueue.cpp](#).

References [toString\(\)](#).

Referenced by [calculateProgress\(\)](#), [getActualTotalNbOfEvents\(\)](#), [getCurrentNbOfEvents\(\)](#), and [getExpectedTotalNbOfEvents\(\)](#).

34.53.4.24 `void stdair::EventQueue::reset ()`

Reset the event queue.

The event queue is fully emptied.

Definition at line 78 of file [EventQueue.cpp](#).

References [_eventList](#), [_progressStatus](#), [_progressStatusMap](#), and [stdair::ProgressStatus::reset\(\)](#).

Referenced by [stdair::STDAIR_Service::reset\(\)](#).

34.53.4.25 ProgressStatusSet stdair::EventQueue::popEvent (EventStruct & ioEventStruct)

Pop the next coming (in time) event, and remove it from the event queue.

- The next coming (in time) event corresponds to the event having the earliest date-time stamp. In other words, it is the first/front element of the event queue.
- That (first) event/element is then removed from the event queue
- The progress status is updated for the corresponding event generator.

Definition at line 291 of file [EventQueue.cpp](#).

References [_eventList](#), [_progressStatus](#), [stdair::EventStruct::getEventType\(\)](#), [getStatus\(\)](#), [stdair::ProgressStatusSet::setOverallStatus\(\)](#), [setStatus\(\)](#), and [stdair::ProgressStatusSet::setTypeSpecificStatus\(\)](#).

Referenced by [stdair::STDAIR_Service::popEvent\(\)](#).

34.53.4.26 bool stdair::EventQueue::addEvent (EventStruct & ioEventStruct)

Add event.

If there already is an event with the same date-time, move the given event one nanosecond forward, and retry the insertion until it succeeds.

That method:

- first adds the event structure in the dedicated list,
- then retrieves the corresponding demand stream,
- and update accordingly the corresponding progress statuses.

Parameters

<code>stdair::EventS</code>	The reference on EventStruct is not constant, because the EventStruct object can be altered: its date-time stamp can be changed accordingly to the location where it has been inserted in the event queue.
-----------------------------	--

Definition at line 351 of file [EventQueue.cpp](#).

References [_eventList](#).

34.53.4.27 bool stdair::EventQueue::isQueueDone () const

States whether the event queue has reached the end.

For now, that method states whether the event queue is empty.

Definition at line 72 of file [EventQueue.cpp](#).

References [_eventList](#), and [isQueueEmpty\(\)](#).

Referenced by [stdair::STDAIR_Service::isQueueDone\(\)](#).

34.53.4.28 void stdair::EventQueue::addStatus (const EventType::EN_EventType & iType, const NbOfRequests_T & iExpectedTotalNbOfEvents)

Initialise the progress statuses for the given event type (e.g., request, snapshot).

The progress status is actually a pair of counters:

- The current number of (already generated) events, for the given event type. That number is initialised to 0 (no event has been generated yet).
- The total number of events (to be generated), also for the given event type.

Definition at line 197 of file [EventQueue.cpp](#).

References [_progressStatus](#), [stdair::ProgressStatus::getActualNb\(\)](#), [stdair::ProgressStatus::getExpectedNb\(\)](#), [stdair::ProgressStatus::setActualNb\(\)](#), [stdair::ProgressStatus::setExpectedNb\(\)](#), and [updateStatus\(\)](#).

34.53.4.29 void stdair::EventQueue::updateStatus (const EventType::EN_EventType & iType, const ProgressStatus & iProgressStatus)

Set/update the progress status for the corresponding event type (e.g., booking request, optimisation notification, schedule change, break point).

If there is no [ProgressStatus](#) object for that event type yet, one is inserted. Otherwise, the [ProgressStatus](#) object is updated.

Definition at line 156 of file [EventQueue.cpp](#).

References [_progressStatusMap](#), [stdair::ProgressStatus::getActualNb\(\)](#), [stdair::ProgressStatus::getCurrentNb\(\)](#), [stdair::ProgressStatus::getExpectedNb\(\)](#), [stdair::EventType::getLabel\(\)](#), [stdair::ProgressStatus::setActualNb\(\)](#), [stdair::ProgressStatus::setCurrentNb\(\)](#), [stdair::ProgressStatus::setExpectedNb\(\)](#), [STDAIR_LOG_ERROR](#), and [toString\(\)](#).

Referenced by [addStatus\(\)](#).

34.53.4.30 void stdair::EventQueue::updateStatus (const EventType::EN_EventType & iType, const NbOfEvents_T & iActualTotalNbOfEvents)

Update the progress statuses for the given event type (e.g., booking request, optimisation notification, schedule change, break point).

The progress status is actually a pair of counters:

- The current number of (already generated) events, for the given event type. That number is initialised to 0 (no event has been generated yet).
- The total number of events (to be generated), also for the given event type.

Definition at line 221 of file [EventQueue.cpp](#).

References [_progressStatusMap](#), [stdair::ProgressStatus::getActualNb\(\)](#), and [stdair::ProgressStatus::setActualNb\(\)](#).

34.53.4.31 ProgressPercentage_T [stdair::EventQueue::calculateProgress \(\)](#) const
[inline]

Calculate the progress status.

The progress is status is the ratio of:

- the current number of events, summed over all the demand streams,
- over the total number of events, also summed over all the demand streams.

Definition at line 316 of file [EventQueue.hpp](#).

References [_progressStatus](#), and [stdair::ProgressStatus::progress\(\)](#).

34.53.4.32 ProgressPercentage_T [stdair::EventQueue::calculateProgress \(const EventType::EN_EventType & iType \)](#) const

Calculate the progress status.

The progress is status is the ratio of:

- the current number of events, summed over all the demand streams,
- over the total number of events, also summed over all the demand streams.

Definition at line 273 of file [EventQueue.cpp](#).

References [_progressStatusMap](#), [display\(\)](#), [stdair::ProgressStatus::progress\(\)](#), and [STDAIR_LOG_ERROR](#).

34.53.4.33 Count_T [stdair::EventQueue::getQueueSize \(\)](#) const

Queue size

Definition at line 62 of file [EventQueue.cpp](#).

References [_eventList](#).

34.53.4.34 bool [stdair::EventQueue::isQueueEmpty \(\)](#) const

Is queue empty

Definition at line 67 of file [EventQueue.cpp](#).

References [_eventList](#).

Referenced by [isQueueDone\(\)](#).

34.53.5 Friends And Related Function Documentation

34.53.5.1 friend class FacBom [friend]

Definition at line 60 of file [EventQueue.hpp](#).

34.53.5.2 friend class FacBomManager [friend]

Definition at line 61 of file [EventQueue.hpp](#).

34.53.6 Member Data Documentation

34.53.6.1 Key_T stdair::EventQueue::_key [protected]

Primary key (ID).

Definition at line 360 of file [EventQueue.hpp](#).

Referenced by [describeKey\(\)](#), and [getKey\(\)](#).

34.53.6.2 BomAbstract* stdair::EventQueue::_parent [protected]

Pointer on the parent class ([BomRoot](#)).

Definition at line 365 of file [EventQueue.hpp](#).

Referenced by [getParent\(\)](#).

34.53.6.3 HolderMap_T stdair::EventQueue::_holderMap [protected]

Map holding the children (e.g., DemandStream objects for booking requests, DCPRule objects for optimisation notifications).

Definition at line 372 of file [EventQueue.hpp](#).

Referenced by [getHolderMap\(\)](#).

34.53.6.4 EventList_T stdair::EventQueue::_eventList [protected]

List of events.

Definition at line 377 of file [EventQueue.hpp](#).

Referenced by [addEvent\(\)](#), [getQueueSize\(\)](#), [isQueueDone\(\)](#), [isQueueEmpty\(\)](#), [popEvent\(\)](#), [reset\(\)](#), [toString\(\)](#), and [~EventQueue\(\)](#).

34.53.6.5 ProgressStatus stdair::EventQueue::_progressStatus [protected]

Counters holding the overall progress status.

Definition at line 382 of file [EventQueue.hpp](#).

Referenced by [addStatus\(\)](#), [calculateProgress\(\)](#), [getActualTotalNbOfEvents\(\)](#), [getCurrentNbOfEvents\(\)](#), [getExpectedTotalNbOfEvents\(\)](#), [getStatus\(\)](#), [popEvent\(\)](#), [reset\(\)](#), [setActualTotalNbOfEvents\(\)](#), [setCurrentNbOfEvents\(\)](#), [setExpectedTotalNbOfEvents\(\)](#), [setStatus\(\)](#), and [toString\(\)](#).

34.53.6.6 ProgressStatusMap_T stdair::EventQueue::_progressStatusMap [protected]

Counters holding the overall progress status, for each event type (e.g., booking request, optimisation notification, schedule change, break point).

Definition at line 389 of file [EventQueue.hpp](#).

Referenced by [calculateProgress\(\)](#), [getActualTotalNbOfEvents\(\)](#), [getCurrentNbOfEvents\(\)](#), [getExpectedTotalNbOfEvents\(\)](#), [getStatus\(\)](#), [reset\(\)](#), [setStatus\(\)](#), and [updateStatus\(\)](#).

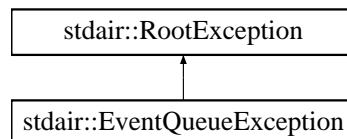
The documentation for this class was generated from the following files:

- [stdair/bom/EventQueue.hpp](#)
- [stdair/bom/EventQueue.cpp](#)

34.54 stdair::EventQueueException Class Reference

```
#include <stdair/stdair_exceptions.hpp>
```

Inheritance diagram for stdair::EventQueueException:



Public Member Functions

- [EventQueueException](#) (const std::string &iWhat)
- const char * [what](#) () const throw ()

Protected Attributes

- std::string [_what](#)

34.54.1 Detailed Description

[EventQueue](#).

Definition at line 203 of file [stdair_exceptions.hpp](#).

34.54.2 Constructor & Destructor Documentation

34.54.2.1 `stdair::EventQueueException::EventQueueException (const std::string & iWhat)`
[inline]

Constructor.

Definition at line 206 of file [stdair_exceptions.hpp](#).

34.54.3 Member Function Documentation

34.54.3.1 `const char* stdair::RootException::what () const throw ()` [`inline`, `inherited`]

Give the details of the exception.

Definition at line 38 of file [stdair_exceptions.hpp](#).

References [stdair::RootException::_what](#).

34.54.4 Member Data Documentation

34.54.4.1 `std::string stdair::RootException::_what` [`protected`, `inherited`]

Details for the exception.

Definition at line 46 of file [stdair_exceptions.hpp](#).

Referenced by [stdair::RootException::what\(\)](#).

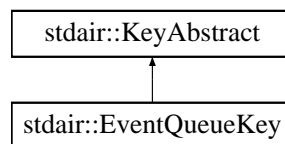
The documentation for this class was generated from the following file:

- [stdair/stdair_exceptions.hpp](#)

34.55 stdair::EventQueueKey Struct Reference

```
#include <stdair/bom/EventQueueKey.hpp>
```

Inheritance diagram for `stdair::EventQueueKey`:



Public Member Functions

- [EventQueueKey](#) (const [EventQueueID_T](#) &)
- [EventQueueKey](#) (const [EventQueueKey](#) &)
- [~EventQueueKey](#) ()
- const [EventQueueID_T](#) & [getEventQueueID](#) () const
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- const std::string [toString](#) () const

34.55.1 Detailed Description

Key of eventqueue.

Definition at line 15 of file [EventQueueKey.hpp](#).

34.55.2 Constructor & Destructor Documentation

34.55.2.1 stdair::EventQueueKey::EventQueueKey (const EventQueueID_T & iEventQueueID)

Constructors.

Definition at line 12 of file [EventQueueKey.cpp](#).

34.55.2.2 stdair::EventQueueKey::EventQueueKey (const EventQueueKey & iKey)

Definition at line 16 of file [EventQueueKey.cpp](#).

34.55.2.3 stdair::EventQueueKey::~EventQueueKey ()

Destructor.

Definition at line 21 of file [EventQueueKey.cpp](#).

34.55.3 Member Function Documentation

34.55.3.1 const EventQueueID_T& stdair::EventQueueKey::getEventQueueID () const [inline]

Get the ID of the [EventQueue](#) object.

Definition at line 31 of file [EventQueueKey.hpp](#).

34.55.3.2 void stdair::EventQueueKey::toStream (std::ostream & ioOut) const [virtual]

Dump a Business Object Key into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 25 of file [EventQueueKey.cpp](#).

References [toString\(\)](#).

34.55.3.3 void stdair::EventQueueKey::fromStream (std::istream & ioin) [virtual]

Read a Business Object Key from an input stream.

Parameters

<i>istream</i> & the input stream.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 30 of file [EventQueueKey.cpp](#).

34.55.3.4 `const std::string stdair::EventQueueKey::toString () const` `[virtual]`

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 34 of file [EventQueueKey.cpp](#).

Referenced by [stdair::EventQueue::describeKey\(\)](#), and [toStream\(\)](#).

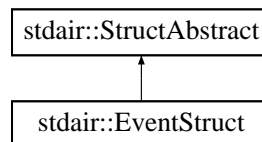
The documentation for this struct was generated from the following files:

- [stdair/bom/EventQueueKey.hpp](#)
- [stdair/bom/EventQueueKey.cpp](#)

34.56 stdair::EventStruct Struct Reference

```
#include <stdair/bom/EventStruct.hpp>
```

Inheritance diagram for `stdair::EventStruct`:

**Public Member Functions**

- `const EventType::EN_EventType & getEventType () const`
- `const BookingRequestStruct & getBookingRequest () const`
- `const CancellationStruct & getCancellation () const`
- `const OptimisationNotificationStruct & getOptimisationNotificationStruct () const`
- `const SnapshotStruct & getSnapshotStruct () const`
- `const RMEventStruct & getRMEvent () const`
- `void fromStream (std::istream &ioln)`
- `const std::string describe () const`
- `EventStruct ()`

- [EventStruct](#) (const [EventType::EN_EventType](#) &, [BookingRequestPtr_T](#))
- [EventStruct](#) (const [EventType::EN_EventType](#) &, [CancellationPtr_T](#))
- [EventStruct](#) (const [EventType::EN_EventType](#) &, const [DateTime_T](#) & iDCPDate, [OptimisationNotificationPtr_T](#))
- [EventStruct](#) (const [EventType::EN_EventType](#) &, [SnapshotPtr_T](#))
- [EventStruct](#) (const [EventType::EN_EventType](#) &, [RMEventPtr_T](#))
- [EventStruct](#) (const [EventStruct](#) &)
- [~EventStruct](#) ()
- void [toStream](#) (std::ostream & ioOut) const

Friends

- struct [EventQueue](#)

34.56.1 Detailed Description

Structure holding the details of an event.

Note

No event should be scheduled before the date-time corresponding to the `DEFAULT_EVENT_OLDEST_DATETIME` constant (as of Feb. 2011, that date is set to Jan. 1, 2010). That constant is specified in the [stdair/basic/BasConst.cpp](#) file. In other words, the simulation should not specified to start before that date-time.

Definition at line 35 of file [EventStruct.hpp](#).

34.56.2 Constructor & Destructor Documentation**34.56.2.1 stdair::EventStruct::EventStruct ()**

Default constructor.

Definition at line 25 of file [EventStruct.cpp](#).

34.56.2.2 stdair::EventStruct::EventStruct (const EventType::EN_EventType & iEventType, BookingRequestPtr_T ioRequestPtr)

Constructor for events corresponding to booking requests.

Definition at line 30 of file [EventStruct.cpp](#).

References [stdair::DEFAULT_EVENT_OLDEST_DATETIME](#).

34.56.2.3 stdair::EventStruct::EventStruct (const EventType::EN_EventType & iEventType, CancellationPtr_T ioCancellationPtr)

Constructor for events corresponding to cancellations.

Definition at line 54 of file [EventStruct.cpp](#).

References [stdair::DEFAULT_EVENT_OLDEST_DATETIME](#).

34.56.2.4 `stdair::EventStruct::EventStruct (const EventType::EN_EventType & iEventType, const DateTime_T & iDCPDate, OptimisationNotificationPtr_T ioOptimisationNotificationPtr)`

Constructor for events corresponding to optimisation requests.

Definition at line 79 of file [EventStruct.cpp](#).

References [stdair::DEFAULT_EVENT_OLDEST_DATETIME](#).

34.56.2.5 `stdair::EventStruct::EventStruct (const EventType::EN_EventType & iEventType, SnapshotPtr_T ioSnapshotPtr)`

Constructor for events corresponding to snapshot requests.

Definition at line 104 of file [EventStruct.cpp](#).

References [stdair::DEFAULT_EVENT_OLDEST_DATETIME](#).

34.56.2.6 `stdair::EventStruct::EventStruct (const EventType::EN_EventType & iEventType, RMEventPtr_T ioRMEventPtr)`

Constructor for events corresponding to RM events.

Definition at line 129 of file [EventStruct.cpp](#).

References [stdair::DEFAULT_EVENT_OLDEST_DATETIME](#).

34.56.2.7 `stdair::EventStruct::EventStruct (const EventStruct & iEventStruct)`

Copy constructor.

Definition at line 154 of file [EventStruct.cpp](#).

34.56.2.8 `stdair::EventStruct::~~EventStruct ()`

Destructor.

Definition at line 208 of file [EventStruct.cpp](#).

34.56.3 Member Function Documentation

34.56.3.1 `const EventType::EN_EventType& stdair::EventStruct::getEventType () const [inline]`

Get the event type

Definition at line 42 of file [EventStruct.hpp](#).

Referenced by [stdair::EventQueue::popEvent\(\)](#).

34.56.3.2 `const BookingRequestStruct& stdair::EventStruct::getBookingRequest () const [inline]`

Get a reference on the booking request referred to by event.

Note

When that event is not of type booking request ([EventType::BKG_REQ](#)), an assertion fails.

Definition at line 52 of file [EventStruct.hpp](#).

```
34.56.3.3 const CancellationStruct& stdair::EventStruct::getCancellation ( ) const  
[inline]
```

Get a reference on the cancellation referred to by event.

Note

When that event is not of type cancellation ([EventType::CX](#)), an assertion fails.

Definition at line 63 of file [EventStruct.hpp](#).

```
34.56.3.4 const OptimisationNotificationStruct&  
stdair::EventStruct::getOptimisationNotificationStruct ( ) const  
[inline]
```

Get a reference on the optimisation notification referred to by event.

Note

When that event is not of type optimisation notification for optimisation notification ([EventType::OPT_NOT_4_FD](#)), an assertion fails.

Definition at line 76 of file [EventStruct.hpp](#).

```
34.56.3.5 const SnapshotStruct& stdair::EventStruct::getSnapshotStruct ( ) const  
[inline]
```

Get a reference on the snapshot referred to by event.

Note

When that event is not of type snapshot for snapshot ([EventType::OPT_NOT_4_FD](#)), an assertion fails.

Definition at line 88 of file [EventStruct.hpp](#).

```
34.56.3.6 const RMEventStruct& stdair::EventStruct::getRMEvent ( ) const [inline]
```

Get a reference on the RM event referred to by the generic event.

Note

When that event is not of type RM event for snapshot ([EventType::OPT_NOT_4_FD](#)), an assertion fails.

Definition at line 100 of file [EventStruct.hpp](#).

34.56.3.7 `void stdair::EventStruct::fromStream (std::istream & ioIn) [virtual]`

Read a Business Object from an input stream.

Parameters

<i>istream</i> & the input stream.

Reimplemented from [stdair::StructAbstract](#).

Definition at line 212 of file [EventStruct.cpp](#).

34.56.3.8 `const std::string stdair::EventStruct::describe () const [virtual]`

Display of the structure.

Implements [stdair::StructAbstract](#).

Definition at line 216 of file [EventStruct.cpp](#).

References [stdair::EventType::BKG_REQ](#), [stdair::EventType::CX](#), [stdair::DEFAULT_EVENT_OLDEST_DATETIME](#), [stdair::EventType::OPT_NOT_4_FD](#), [stdair::EventType::RM](#), and [stdair::EventType::SNAPSHOT](#).

34.56.3.9 `void stdair::StructAbstract::toStream (std::ostream & ioOut) const [inline, inherited]`

Dump a Business Object into an output stream.

Parameters

<i>ostream</i> & the output stream.

Reimplemented in [stdair::YieldRange](#), [stdair::AirlineStruct](#), [stdair::BookingRequestStruct](#), [stdair::CancellationStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::TravelSolutionStruct](#), and [stdair::VirtualClassStruct](#).

Definition at line 29 of file [StructAbstract.hpp](#).

References [stdair::StructAbstract::describe\(\)](#).

Referenced by [operator<<\(\)](#).

34.56.4 Friends And Related Function Documentation

34.56.4.1 `friend struct EventQueue [friend]`

Definition at line 37 of file [EventStruct.hpp](#).

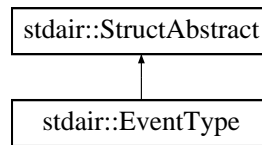
The documentation for this struct was generated from the following files:

- [stdair/bom/EventStruct.hpp](#)
- [stdair/bom/EventStruct.cpp](#)

34.57 stdair::EventType Struct Reference

```
#include <stdair/basic/EventType.hpp>
```

Inheritance diagram for stdair::EventType:



Public Types

- enum [EN_EventType](#) {
 [BKG_REQ](#) = 0, [CX](#), [OPT_NOT_4_FD](#), [OPT_NOT_4_NET](#),
 [SKD_CHG](#), [SNAPSHOT](#), [RM](#), [BRK_PT](#),
 [LAST_VALUE](#) }

Public Member Functions

- [EN_EventType](#) [getType](#) () const
- std::string [getTypeAsString](#) () const
- const std::string [describe](#) () const
- bool [operator==](#) (const [EN_EventType](#) &) const
- [EventType](#) (const [EN_EventType](#) &)
- [EventType](#) (const char iType)
- [EventType](#) (const [EventType](#) &)
- void [toStream](#) (std::ostream &ioOut) const
- virtual void [fromStream](#) (std::istream &ioIn)

Static Public Member Functions

- static const std::string & [getLabel](#) (const [EN_EventType](#) &)
- static char [getTypeLabel](#) (const [EN_EventType](#) &)
- static std::string [getTypeLabelAsString](#) (const [EN_EventType](#) &)
- static std::string [describeLabels](#) ()

34.57.1 Detailed Description

Enumeration of event types.

Definition at line 15 of file [EventType.hpp](#).

34.57.2 Member Enumeration Documentation

34.57.2.1 enum stdair::EventType::EN_EventType

Enumerator:

BKG_REQ
CX
OPT_NOT_4_FD
OPT_NOT_4_NET
SKD_CHG
SNAPSHOT
RM
BRK_PT
LAST_VALUE

Definition at line 17 of file [EventType.hpp](#).

34.57.3 Constructor & Destructor Documentation

34.57.3.1 stdair::EventType::EventType (const EN_EventType & iEventType)

Constructor.

Definition at line 36 of file [EventType.cpp](#).

34.57.3.2 stdair::EventType::EventType (const char iType)

Constructor.

Definition at line 41 of file [EventType.cpp](#).

References [BKG_REQ](#), [BRK_PT](#), [CX](#), [describeLabels\(\)](#), [LAST_VALUE](#), [OPT_NOT_4_FD](#), [OPT_NOT_4_NET](#), [RM](#), [SKD_CHG](#), and [SNAPSHOT](#).

34.57.3.3 stdair::EventType::EventType (const EventType & iEventType)

Default copy constructor.

Definition at line 31 of file [EventType.cpp](#).

34.57.4 Member Function Documentation

34.57.4.1 const std::string & stdair::EventType::getLabel (const EN_EventType & iType)
[static]

Get the label as a string (e.g., "BookingRequest" or "ScheduleChange").

Definition at line 64 of file [EventType.cpp](#).

Referenced by [stdair::EventQueue::updateStatus\(\)](#).

34.57.4.2 `char stdair::EventType::getTypeLabel (const EN_EventType & iType)`
`[static]`

Get the label as a single char (e.g., 'B' or 'S').

Definition at line 69 of file [EventType.cpp](#).

34.57.4.3 `std::string stdair::EventType::getTypeLabelAsString (const EN_EventType & iType)`
`[static]`

Get the label as a string of a single char (e.g., "B" or "S").

Definition at line 74 of file [EventType.cpp](#).

34.57.4.4 `std::string stdair::EventType::describeLabels ()` `[static]`

List the labels.

Definition at line 81 of file [EventType.cpp](#).

References [LAST_VALUE](#).

Referenced by [EventType\(\)](#).

34.57.4.5 `EventType::EN_EventType stdair::EventType::getType () const`

Get the enumerated value.

Definition at line 93 of file [EventType.cpp](#).

34.57.4.6 `std::string stdair::EventType::getTypeAsString () const`

Get the enumerated value as a short string (e.g., "B" or "S").

Definition at line 98 of file [EventType.cpp](#).

34.57.4.7 `const std::string stdair::EventType::describe () const` `[virtual]`

Give a description of the structure (e.g., "BookingRequest" or "ScheduleChange").

Implements [stdair::StructAbstract](#).

Definition at line 105 of file [EventType.cpp](#).

34.57.4.8 `bool stdair::EventType::operator== (const EN_EventType & iType) const`

Comparison operator.

Definition at line 112 of file [EventType.cpp](#).

34.57.4.9 `void stdair::StructAbstract::toStream (std::ostream & ioOut) const` `[inline, inherited]`

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i>	the output stream.
---------------------	--------------------

Reimplemented in [stdair::YieldRange](#), [stdair::AirlineStruct](#), [stdair::BookingRequestStruct](#), [stdair::CancellationStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::TravelSolutionStruct](#), and [stdair::VirtualClassStruct](#).

Definition at line 29 of file [StructAbstract.hpp](#).

References [stdair::StructAbstract::describe\(\)](#).

Referenced by [operator<<\(\)](#).

34.57.4.10 `virtual void stdair::StructAbstract::fromStream (std::istream & ioln) [inline, virtual, inherited]`

Read a Business Object from an input stream.

Parameters

<i>istream&</i> the input stream.

Reimplemented in [stdair::ProgressStatusSet](#), [stdair::YieldRange](#), [stdair::AirlineStruct](#), [stdair::BookingRequestStruct](#), [stdair::CancellationStruct](#), [stdair::EventStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::TravelSolutionStruct](#), and [stdair::VirtualClassStruct](#).

Definition at line 38 of file [StructAbstract.hpp](#).

Referenced by [operator>>\(\)](#).

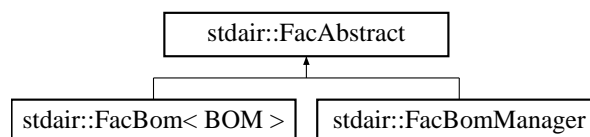
The documentation for this struct was generated from the following files:

- [stdair/basic/EventType.hpp](#)
- [stdair/basic/EventType.cpp](#)

34.58 stdair::FacAbstract Class Reference

```
#include <stdair/factory/FacAbstract.hpp>
```

Inheritance diagram for `stdair::FacAbstract`:



Public Member Functions

- virtual [~FacAbstract\(\)](#)

Protected Member Functions

- [FacAbstract\(\)](#)

34.58.1 Detailed Description

Base class for Factory layer.

Definition at line 10 of file [FacAbstract.hpp](#).

34.58.2 Constructor & Destructor Documentation

34.58.2.1 stdair::FacAbstract::~~FacAbstract () [virtual]

Destructor.

Definition at line 13 of file [FacAbstract.cpp](#).

34.58.2.2 stdair::FacAbstract::FacAbstract () [inline, protected]

Default Constructor.

This constructor is protected to ensure the class is abstract.

Definition at line 18 of file [FacAbstract.hpp](#).

The documentation for this class was generated from the following files:

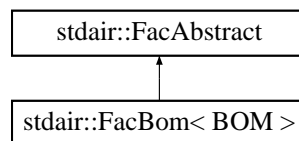
- [stdair/factory/FacAbstract.hpp](#)
- [stdair/factory/FacAbstract.cpp](#)

34.59 stdair::FacBom< BOM > Class Template Reference

Base class for Factory layer.

```
#include <stdair/factory/FacBom.hpp>
```

Inheritance diagram for stdair::FacBom< BOM >:



Public Member Functions

- BOM & [create](#) ()
- BOM & [create](#) (const Key_T &)
- [~FacBom](#) ()
- void [clean](#) ()

Static Public Member Functions

- static [FacBom](#) & [instance](#) ()

Protected Member Functions

- [FacBom](#) ()

34.59.1 Detailed Description

`template<typename BOM>class stdair::FacBom< BOM >`

Base class for Factory layer.

Definition at line 22 of file [FacBom.hpp](#).

34.59.2 Constructor & Destructor Documentation

34.59.2.1 `template<typename BOM> stdair::FacBom< BOM >::FacBom ()`
[inline, protected]

Default Constructor.

Definition at line 49 of file [FacBom.hpp](#).

34.59.2.2 `template<typename BOM> stdair::FacBom< BOM >::~~FacBom ()`
[inline]

Destructor.

Definition at line 55 of file [FacBom.hpp](#).

References [stdair::FacBom< BOM >::clean\(\)](#).

34.59.3 Member Function Documentation

34.59.3.1 `template<typename BOM > FacBom< BOM > & stdair::FacBom< BOM >::instance ()` [static]

Provide the unique instance.

The singleton is instantiated when first used.

Returns

[FacBom](#)&

Definition at line 83 of file [FacBom.hpp](#).

References [stdair::FacSupervisor::instance\(\)](#), and [stdair::FacSupervisor::registerBomFactory\(\)](#).

34.59.3.2 `template<typename BOM > BOM & stdair::FacBom< BOM >::create ()`

Create a BOM object, given a key or not.

Definition at line 111 of file [FacBom.hpp](#).

34.59.3.3 `template<typename BOM > BOM & stdair::FacBom< BOM >::create (const Key_T & iKey)`

Definition at line 117 of file [FacBom.hpp](#).

34.59.3.4 `template<typename BOM > void stdair::FacBom< BOM >::clean ()`

Destroyed all the object instantiated by this factory.

Definition at line 94 of file [FacBom.hpp](#).

Referenced by [stdair::FacBom< BOM >::~~FacBom\(\)](#).

The documentation for this class was generated from the following file:

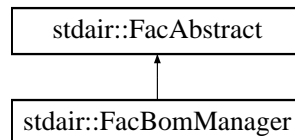
- [stdair/factory/FacBom.hpp](#)

34.60 stdair::FacBomManager Class Reference

Utility class for linking StdAir-based objects.

```
#include <stdair/factory/FacBomManager.hpp>
```

Inheritance diagram for `stdair::FacBomManager`:



Public Member Functions

- [~FacBomManager](#) ()
- `template<>`
void [addToList](#) ([SegmentDate](#) &ioSegmentDate, [SegmentDate](#) &ioMarketingSegmentDate)

Static Public Member Functions

- `template<typename OBJECT2 , typename OBJECT1 >`
static [BomHolder](#)< OBJECT2 > * [getBomHolderPtr](#) (OBJECT1 &)
- `template<typename OBJECT2 , typename OBJECT1 >`
static [BomHolder](#)< OBJECT2 > & [addBomHolder](#) (OBJECT1 &)

- `template<typename OBJECT1 , typename OBJECT2 >`
static void [addToList](#) (OBJECT1 &, OBJECT2 &)
- `template<typename OBJECT1 , typename OBJECT2 >`
static void [addToMap](#) (OBJECT1 &, OBJECT2 &, const [MapKey_T](#) &)
- `template<typename OBJECT1 , typename OBJECT2 >`
static void [addToMap](#) (OBJECT1 &, OBJECT2 &)
- `template<typename OBJECT1 , typename OBJECT2 >`
static void [addToListAndMap](#) (OBJECT1 &, OBJECT2 &)
- `template<typename OBJECT1 , typename OBJECT2 >`
static void [addToListAndMap](#) (OBJECT1 &, OBJECT2 &, const [MapKey_T](#) &)
- `template<typename PARENT , typename CHILD >`
static void [linkWithParent](#) (PARENT &, CHILD &)
- `template<typename OBJECT2 , typename OBJECT1 >`
static void [cloneHolder](#) (OBJECT1 &, const OBJECT1 &)

Protected Member Functions

- [FacBomManager](#) ()

34.60.1 Detailed Description

Utility class for linking StdAir-based objects.

Definition at line 28 of file [FacBomManager.hpp](#).

34.60.2 Constructor & Destructor Documentation

34.60.2.1 stdair::FacBomManager::FacBomManager () [inline, protected]

Default Constructor.

This constructor is protected to comply with the singleton pattern.

Definition at line 200 of file [FacBomManager.hpp](#).

34.60.2.2 stdair::FacBomManager::~~FacBomManager () [inline]

Destructor.

Definition at line 206 of file [FacBomManager.hpp](#).

34.60.3 Member Function Documentation

34.60.3.1 `template<typename OBJECT2 , typename OBJECT1 > BomHolder< OBJECT2 > * stdair::FacBomManager::getBomHolderPtr (OBJECT1 & ioObject1) [static]`

Retrieve a pointer on the holder of children (OBJECT2 type) for the parent (OBJECT1 type). If the holder does not exist, return NULL.

Parameters

<i>typename</i> OBJECT1 & Parent object.
--

Returns

typename BomHolder<OBJECT2>* [BomHolder](#) for the children objects.

Definition at line 239 of file [FacBomManager.hpp](#).

```
34.60.3.2  template<typename OBJECT2 , typename OBJECT1 > BomHolder< OBJECT2 >
          & stdair::FacBomManager::addBomHolder ( OBJECT1 & ioObject1 )  [static]
```

Instantiate a BomHolder<OBJECT2> object, add it to the OBJECT1-typed object, given as parameter, and return a reference on that newly created [BomHolder](#).

Parameters

<i>typename</i> OBJECT1 & Parent object.
--

Returns

typename BomHolder<OBJECT2>& Just created [BomHolder](#) (e.g., for the children objects).

Definition at line 213 of file [FacBomManager.hpp](#).

```
34.60.3.3  template<typename OBJECT1 , typename OBJECT2 > void
          stdair::FacBomManager::addToList ( OBJECT1 & ioObject1, OBJECT2 & ioObject2 )
          [static]
```

Add an OBJECT2-typed object (typically, a child) to the dedicated list held by the OBJECT1-typed object (typically, a parent).

Note

The underlying list is actually stored within an object of type BomHolder<OBJECT2>.

Parameters

<i>typename</i> OBJECT1 & Parent object.
--

<i>typename</i> OBJECT2 & Child object.

Definition at line 299 of file [FacBomManager.hpp](#).

```
34.60.3.4  template<typename OBJECT1 , typename OBJECT2 > void
          stdair::FacBomManager::addToMap ( OBJECT1 & ioObject1, OBJECT2 & ioObject2,
          const MapKey_T & iKey )  [static]
```

Add an OBJECT2-typed object (typically, a child) to the dedicated map held by the OBJECT1-typed object (typically, a parent).

Note

The underlying map is actually stored within an object of type BomHolder<OBJECT2>.

Parameters

<i>typename</i>	OBJECT1& Parent object.
<i>typename</i>	OBJECT2& Child object.
<i>const</i>	MapKey_T&

Definition at line 347 of file [FacBomManager.hpp](#).

Referenced by [addToMap\(\)](#).

```
34.60.3.5  template<typename OBJECT1 , typename OBJECT2 > void
           stdair::FacBomManager::addToMap ( OBJECT1 & ioObject1, OBJECT2 & ioObject2 )
           [static]
```

Add an OBJECT2-typed object (typically, a child) to the dedicated map held by the OBJECT1-typed object (typically, a parent).

Note

The underlying map is actually stored within an object of type BomHolder<OBJECT2>.

Parameters

<i>typename</i>	OBJECT1& Parent object.
<i>typename</i>	OBJECT2& Child object.

Definition at line 366 of file [FacBomManager.hpp](#).

References [addToMap\(\)](#).

```
34.60.3.6  template<typename OBJECT1 , typename OBJECT2 > void
           stdair::FacBomManager::addToListAndMap ( OBJECT1 & ioObject1, OBJECT2 &
           ioObject2 ) [static]
```

Add an OBJECT2-typed object (typically, a child) to the dedicated containers (list and map) held by the OBJECT1-typed object (typically, a parent).

Note

The underlying containers are actually stored within an object of type BomHolder<OBJECT2>.

Parameters

<i>typename</i>	OBJECT1& Parent object.
<i>typename</i>	OBJECT2& Child object.

Definition at line 404 of file [FacBomManager.hpp](#).

```
34.60.3.7 template<typename OBJECT1 , typename OBJECT2 > void
stdair::FacBomManager::addToListAndMap ( OBJECT1 & ioObject1, OBJECT2 &
ioObject2, const MapKey_T & iKey ) [static]
```

Add an OBJECT2-typed object (typically, a child) to the dedicated containers (list and map) held by the OBJECT1-typed object (typically, a parent).

Note

The underlying containers are actually stored within an object of type BomHolder<OBJECT2>.

Parameters

<i>typename</i>	OBJECT1& Parent object.
<i>typename</i>	OBJECT2& Child object.
<i>const</i>	MapKey_T&

Definition at line 384 of file [FacBomManager.hpp](#).

```
34.60.3.8 template<typename PARENT , typename CHILD > void
stdair::FacBomManager::linkWithParent ( PARENT & ioParent, CHILD & ioChild )
[static]
```

Allow the CHILD object to store a pointer on its PARENT object.

Parameters

<i>typename</i>	PARENT& Parent object.
<i>typename</i>	CHILD& Child object.

Definition at line 422 of file [FacBomManager.hpp](#).

Referenced by [stdair::serialiseHelper\(\)](#).

```
34.60.3.9 template<typename OBJECT2 , typename OBJECT1 > void
stdair::FacBomManager::cloneHolder ( OBJECT1 & ioDest, const OBJECT1 & iOri )
[static]
```

Clone the underlying containers (held by the BomHolder<OBJECT2>-typed holder) of the OBJECT1-typed object.

Note

The underlying containers are actually stored within an object of type BomHolder<OBJECT2>.

Parameters

<i>typename</i>	OBJECT1& Parent object.
<i>typename</i>	OBJECT2& Child object.

Definition at line 430 of file [FacBomManager.hpp](#).

References [stdair::BomHolder< BOM >::_bomList](#), and [stdair::BomHolder< BOM >::_bomMap](#).

34.60.3.10 `template<> void stdair::FacBomManager::addToList (SegmentDate & ioSegmentDate, SegmentDate & ioMarketingSegmentDate) [inline]`

Definition at line 450 of file [FacBomManager.hpp](#).

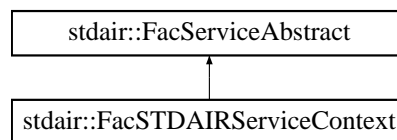
The documentation for this class was generated from the following file:

- [stdair/factory/FacBomManager.hpp](#)

34.61 stdair::FacServiceAbstract Class Reference

```
#include <stdair/service/FacServiceAbstract.hpp>
```

Inheritance diagram for `stdair::FacServiceAbstract`:



Public Types

- `typedef std::vector< ServiceAbstract * > ServicePool_T`

Public Member Functions

- virtual `~FacServiceAbstract ()`
- void `clean ()`

Protected Member Functions

- `FacServiceAbstract ()`

Protected Attributes

- `ServicePool_T _pool`

34.61.1 Detailed Description

Base class for the (Service) Factory layer.

Definition at line 16 of file [FacServiceAbstract.hpp](#).

34.61.2 Member Typedef Documentation

34.61.2.1 `typedef std::vector<ServiceAbstract*> stdair::FacServiceAbstract::ServicePool_T`

Define the list (pool) of Service objects.

Definition at line 20 of file [FacServiceAbstract.hpp](#).

34.61.3 Constructor & Destructor Documentation

34.61.3.1 `stdair::FacServiceAbstract::~~FacServiceAbstract () [virtual]`

Destructor.

Definition at line 13 of file [FacServiceAbstract.cpp](#).

References [clean\(\)](#).

34.61.3.2 `stdair::FacServiceAbstract::FacServiceAbstract () [inline, protected]`

Default Constructor.

This constructor is protected to ensure the class is abstract.

Definition at line 31 of file [FacServiceAbstract.hpp](#).

34.61.4 Member Function Documentation

34.61.4.1 `void stdair::FacServiceAbstract::clean ()`

Destroyed all the object instantiated by this factory.

Definition at line 18 of file [FacServiceAbstract.cpp](#).

References [_pool](#).

Referenced by [~FacServiceAbstract\(\)](#).

34.61.5 Member Data Documentation

34.61.5.1 `ServicePool_T stdair::FacServiceAbstract::_pool [protected]`

List of instantiated Business Objects

Definition at line 34 of file [FacServiceAbstract.hpp](#).

Referenced by [clean\(\)](#), and [stdair::FacSTDAIRServiceContext::create\(\)](#).

The documentation for this class was generated from the following files:

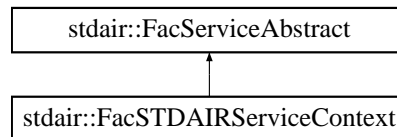
- [stdair/service/FacServiceAbstract.hpp](#)
- [stdair/service/FacServiceAbstract.cpp](#)

34.62 stdair::FacSTDAIRServiceContext Class Reference

Factory for [Bucket](#).

```
#include <stdair/service/FacSTDAIRServiceContext.hpp>
```

Inheritance diagram for stdair::FacSTDAIRServiceContext:



Public Types

- typedef std::vector< [ServiceAbstract](#) * > [ServicePool_T](#)

Public Member Functions

- [~FacSTDAIRServiceContext](#) ()
- [STDAIR_ServiceContext](#) & [create](#) ()
- void [clean](#) ()

Static Public Member Functions

- static [FacSTDAIRServiceContext](#) & [instance](#) ()

Protected Member Functions

- [FacSTDAIRServiceContext](#) ()

Protected Attributes

- [ServicePool_T](#) _pool

34.62.1 Detailed Description

Factory for [Bucket](#).

Definition at line 18 of file [FacSTDAIRServiceContext.hpp](#).

34.62.2 Member Typedef Documentation

34.62.2.1 `typedef std::vector<ServiceAbstract*>
stdair::FacServiceAbstract::ServicePool_T
[inherited]`

Define the list (pool) of Service objects.

Definition at line 20 of file [FacServiceAbstract.hpp](#).

34.62.3 Constructor & Destructor Documentation

34.62.3.1 `stdair::FacSTDAIRServiceContext::~~FacSTDAIRServiceContext ()`

Destructor.

The Destruction put the `_instance` to NULL in order to be clean for the next [FacSTDAIRServiceContext::instance\(\)](#).

Definition at line 16 of file [FacSTDAIRServiceContext.cpp](#).

34.62.3.2 `stdair::FacSTDAIRServiceContext::FacSTDAIRServiceContext () [inline,
protected]`

Default Constructor.

This constructor is protected in order to ensure the singleton pattern.

Definition at line 54 of file [FacSTDAIRServiceContext.hpp](#).

Referenced by [instance\(\)](#).

34.62.4 Member Function Documentation

34.62.4.1 `FacSTDAIRServiceContext & stdair::FacSTDAIRServiceContext::instance ()
[static]`

Provide the unique instance.

The singleton is instantiated when first used.

Returns

[FacSTDAIRServiceContext&](#)

Definition at line 21 of file [FacSTDAIRServiceContext.cpp](#).

References [FacSTDAIRServiceContext\(\)](#).

34.62.4.2 `STDAIR_ServiceContext & stdair::FacSTDAIRServiceContext::create ()`

Create a new [STDAIR_ServiceContext](#) object.

This new object is added to the list of instantiated objects.

Returns

[STDAIR_ServiceContext](#)& The newly created object.

Definition at line 33 of file [FacSTDAIRServiceContext.cpp](#).

References [stdair::FacServiceAbstract::_pool](#).

34.62.4.3 void [stdair::FacServiceAbstract::clean](#) () [inherited]

Destroyed all the object instantiated by this factory.

Definition at line 18 of file [FacServiceAbstract.cpp](#).

References [stdair::FacServiceAbstract::_pool](#).

Referenced by [stdair::FacServiceAbstract::~~FacServiceAbstract\(\)](#).

34.62.5 Member Data Documentation

34.62.5.1 [ServicePool_T](#) [stdair::FacServiceAbstract::_pool](#) [protected, inherited]

List of instantiated Business Objects

Definition at line 34 of file [FacServiceAbstract.hpp](#).

Referenced by [stdair::FacServiceAbstract::clean\(\)](#), and [create\(\)](#).

The documentation for this class was generated from the following files:

- [stdair/service/FacSTDAIRServiceContext.hpp](#)
- [stdair/service/FacSTDAIRServiceContext.cpp](#)

34.63 stdair::FacSupervisor Class Reference

```
#include <stdair/service/FacSupervisor.hpp>
```

Public Types

- typedef std::list< [FacAbstract](#) * > [BomFactoryPool_T](#)
- typedef std::list< [FacServiceAbstract](#) * > [ServiceFactoryPool_T](#)

Public Member Functions

- void [registerBomFactory](#) ([FacAbstract](#) *)
- void [registerServiceFactory](#) ([FacServiceAbstract](#) *)
- void [cleanBomLayer](#) ()
- void [cleanServiceLayer](#) ()
- [~FacSupervisor](#) ()

Static Public Member Functions

- static [FacSupervisor](#) & [instance](#) ()
- static void [cleanLoggerService](#) ()
- static void [cleanDBSessionManager](#) ()
- static void [cleanAll](#) ()

Protected Member Functions

- [FacSupervisor](#) ()
- [FacSupervisor](#) (const [FacSupervisor](#) &)

34.63.1 Detailed Description

Singleton class to register and clean all Factories.

Definition at line 20 of file [FacSupervisor.hpp](#).

34.63.2 Member Typedef Documentation

34.63.2.1 `typedef std::list<FacAbstract*>
stdair::FacSupervisor::BomFactoryPool_T`

Define the pool (list) of factories.

Definition at line 25 of file [FacSupervisor.hpp](#).

34.63.2.2 `typedef std::list<FacServiceAbstract*>
stdair::FacSupervisor::ServiceFactoryPool_T`

Definition at line 26 of file [FacSupervisor.hpp](#).

34.63.3 Constructor & Destructor Documentation

34.63.3.1 `stdair::FacSupervisor::~~FacSupervisor ()`

Destructor.

That destructors is applied on the static instance. It then deletes in turn all the other registered objects.

Definition at line 27 of file [FacSupervisor.cpp](#).

References [cleanBomLayer\(\)](#), and [cleanServiceLayer\(\)](#).

34.63.3.2 `stdair::FacSupervisor::FacSupervisor ()` [`inline`, `protected`]

Default Constructor.

This constructor is protected to ensure the singleton pattern.

Definition at line 102 of file [FacSupervisor.hpp](#).

Referenced by [instance\(\)](#).

34.63.3.3 `stdair::FacSupervisor::FacSupervisor (const FacSupervisor &) [inline, protected]`

Definition at line 103 of file [FacSupervisor.hpp](#).

34.63.4 Member Function Documentation

34.63.4.1 `FacSupervisor & stdair::FacSupervisor::instance () [static]`

Provide the unique (static) instance of the [FacSupervisor](#) object.

The singleton is instantiated when first used.

Returns

[FacSupervisor&](#)

Definition at line 18 of file [FacSupervisor.cpp](#).

References [FacSupervisor\(\)](#).

Referenced by [stdair::FacBom< BOM >::instance\(\)](#).

34.63.4.2 `void stdair::FacSupervisor::registerBomFactory (FacAbstract * ioFac_ptr)`

Register a newly instantiated concrete factory for the Bom layer.

When a concrete Factory is firstly instantiated this factory have to register itself to the [FacSupervisor](#)

Parameters

<i>FacBom*</i>	The concrete Factory to register.
----------------	-----------------------------------

Definition at line 33 of file [FacSupervisor.cpp](#).

Referenced by [stdair::FacBom< BOM >::instance\(\)](#).

34.63.4.3 `void stdair::FacSupervisor::registerServiceFactory (FacServiceAbstract * ioFac_ptr)`

Register a newly instantiated concrete factory for the Service layer.

When a concrete Factory is firstly instantiated this factory have to register itself to the [FacSupervisor](#).

Parameters

<i>FacService-Abstract*</i>	the concrete Factory to register.
-----------------------------	-----------------------------------

Definition at line 38 of file [FacSupervisor.cpp](#).

34.63.4.4 void stdair::FacSupervisor::cleanBomLayer ()

Clean all registered object.

Call the clean method of all the instantiated factories for the BomStructure layer.

Definition at line 43 of file [FacSupervisor.cpp](#).

Referenced by [~FacSupervisor\(\)](#).

34.63.4.5 void stdair::FacSupervisor::cleanServiceLayer ()

Clean all Service created object.

Call the clean method of all the instantiated factories for the Service layer.

Definition at line 57 of file [FacSupervisor.cpp](#).

Referenced by [~FacSupervisor\(\)](#).

34.63.4.6 void stdair::FacSupervisor::cleanLoggerService () [static]

Delete the static instance of the [Logger](#) object.

Definition at line 71 of file [FacSupervisor.cpp](#).

Referenced by [cleanAll\(\)](#).

34.63.4.7 void stdair::FacSupervisor::cleanDBSessionManager () [static]

Delete the static instance of the [DBSessionManager](#) object.

Definition at line 77 of file [FacSupervisor.cpp](#).

Referenced by [cleanAll\(\)](#).

34.63.4.8 void stdair::FacSupervisor::cleanAll () [static]

Clean the static instance. As the static instance (singleton) is deleted, all the other registered objects will be deleted in turn.

Definition at line 83 of file [FacSupervisor.cpp](#).

References [cleanDBSessionManager\(\)](#), and [cleanLoggerService\(\)](#).

The documentation for this class was generated from the following files:

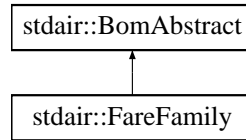
- [stdair/service/FacSupervisor.hpp](#)
- [stdair/service/FacSupervisor.cpp](#)

34.64 stdair::FareFamily Class Reference

Class representing the actual attributes for a family fare.

```
#include <stdair/bom/FareFamily.hpp>
```

Inheritance diagram for stdair::FareFamily:



Public Types

- typedef [FareFamilyKey](#) [Key_T](#)

Public Member Functions

- const [Key_T](#) & [getKey](#) () const
- [BomAbstract](#) *const [getParent](#) () const
- const [FamilyCode_T](#) & [getFamilyCode](#) () const
- const [HolderMap_T](#) & [getHolderMap](#) () const
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- std::string [toString](#) () const
- const std::string [describeKey](#) () const
- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int iFileVersion)

Public Attributes

- [Key_T _key](#)
- [BomAbstract](#) * [_parent](#)
- [HolderMap_T](#) [_holderMap](#)

Protected Member Functions

- [FareFamily](#) (const [Key_T](#) &)
- virtual [~FareFamily](#) ()

Friends

- class [FacBom](#)
- class [FacBomManager](#)
- class [boost::serialization::access](#)

34.64.1 Detailed Description

Class representing the actual attributes for a family fare.

Definition at line 27 of file [FareFamily.hpp](#).

34.64.2 Member Typedef Documentation

34.64.2.1 typedef FareFamilyKey stdair::FareFamily::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 37 of file [FareFamily.hpp](#).

34.64.3 Constructor & Destructor Documentation

34.64.3.1 stdair::FareFamily::FareFamily (const Key_T & iKey) [protected]

Constructor.

Definition at line 29 of file [FareFamily.cpp](#).

34.64.3.2 stdair::FareFamily::~~FareFamily () [protected, virtual]

Destructor.

Definition at line 33 of file [FareFamily.cpp](#).

34.64.4 Member Function Documentation

34.64.4.1 const Key_T& stdair::FareFamily::getKey () const [inline]

Get the family fare key.

Definition at line 43 of file [FareFamily.hpp](#).

References [_key](#).

34.64.4.2 BomAbstract* const stdair::FareFamily::getParent () const [inline]

Get the parent object.

Definition at line 48 of file [FareFamily.hpp](#).

References [_parent](#).

34.64.4.3 const FamilyCode_T& stdair::FareFamily::getFamilyCode () const [inline]

Get the family fare code (part of the primary key).

Definition at line 53 of file [FareFamily.hpp](#).

References [_key](#), and [stdair::FareFamilyKey::getFamilyCode\(\)](#).

Referenced by [stdair::jsonBookingClassExport\(\)](#), and [stdair::jsonFareFamilyExport\(\)](#).

34.64.4.4 `const HolderMap_T& stdair::FareFamily::getHolderMap () const` `[inline]`

Get the map of children holders.

Definition at line 58 of file [FareFamily.hpp](#).

References [_holderMap](#).

34.64.4.5 `void stdair::FareFamily::toStream (std::ostream & ioOut) const` `[inline, virtual]`

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Implements [stdair::BomAbstract](#).

Definition at line 70 of file [FareFamily.hpp](#).

References [toString\(\)](#).

34.64.4.6 `void stdair::FareFamily::fromStream (std::istream & ioin)` `[inline, virtual]`

Read a Business Object from an input stream.

Parameters

<i>istream&</i> the input stream.

Implements [stdair::BomAbstract](#).

Definition at line 79 of file [FareFamily.hpp](#).

34.64.4.7 `std::string stdair::FareFamily::toString () const` `[virtual]`

Get the serialised version of the Business Object.

Implements [stdair::BomAbstract](#).

Definition at line 37 of file [FareFamily.cpp](#).

References [describeKey\(\)](#).

Referenced by [toStream\(\)](#).

34.64.4.8 `const std::string stdair::FareFamily::describeKey () const` `[inline]`

Get a string describing the key.

Definition at line 90 of file [FareFamily.hpp](#).

References [_key](#), and [stdair::FareFamilyKey::toString\(\)](#).

Referenced by [toString\(\)](#).

34.64.4.9 `template<class Archive > void stdair::FareFamily::serialize (Archive & ar, const unsigned int iFileVersion)`

Serialisation.

Definition at line 59 of file [FareFamily.cpp](#).

References [_key](#).

34.64.5 Friends And Related Function Documentation

34.64.5.1 `friend class FacBom [friend]`

Definition at line 28 of file [FareFamily.hpp](#).

34.64.5.2 `friend class FacBomManager [friend]`

Definition at line 29 of file [FareFamily.hpp](#).

34.64.5.3 `friend class boost::serialization::access [friend]`

Definition at line 30 of file [FareFamily.hpp](#).

34.64.6 Member Data Documentation

34.64.6.1 `Key_T stdair::FareFamily::_key`

Primary key (fare family code).

Definition at line 141 of file [FareFamily.hpp](#).

Referenced by [describeKey\(\)](#), [getFamilyCode\(\)](#), [getKey\(\)](#), and [serialize\(\)](#).

34.64.6.2 `BomAbstract* stdair::FareFamily::_parent`

Pointer on the parent class ([SegmentCabin](#)).

Definition at line 146 of file [FareFamily.hpp](#).

Referenced by [getParent\(\)](#).

34.64.6.3 `HolderMap_T stdair::FareFamily::_holderMap`

Map holding the children ([BookingClass](#) objects).

Definition at line 151 of file [FareFamily.hpp](#).

Referenced by [getHolderMap\(\)](#).

The documentation for this class was generated from the following files:

- `stdair/bom/FareFamily.hpp`

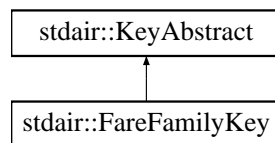
- [stdair/bom/FareFamily.cpp](#)

34.65 stdair::FareFamilyKey Struct Reference

Key of a given fare family, made of a fare family code.

```
#include <stdair/bom/FareFamilyKey.hpp>
```

Inheritance diagram for stdair::FareFamilyKey:



Public Member Functions

- [FareFamilyKey](#) (const [FamilyCode_T](#) &iFamilyCode)
- [FareFamilyKey](#) (const [FareFamilyKey](#) &)
- [~FareFamilyKey](#) ()
- const [FamilyCode_T](#) & [getFamilyCode](#) () const
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- const std::string [toString](#) () const
- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int iFileVersion)

Friends

- class [boost::serialization::access](#)

34.65.1 Detailed Description

Key of a given fare family, made of a fare family code.

Definition at line 26 of file [FareFamilyKey.hpp](#).

34.65.2 Constructor & Destructor Documentation

34.65.2.1 stdair::FareFamilyKey::FareFamilyKey (const FamilyCode_T & iFamilyCode)

Constructor.

Definition at line 28 of file [FareFamilyKey.cpp](#).

34.65.2.2 stdair::FareFamilyKey::FareFamilyKey (const FareFamilyKey & iFareFamilyKey)

Copy constructor.

Definition at line 23 of file [FareFamilyKey.cpp](#).

34.65.2.3 stdair::FareFamilyKey::~~FareFamilyKey ()

Destructor.

Definition at line 33 of file [FareFamilyKey.cpp](#).

34.65.3 Member Function Documentation

34.65.3.1 const FamilyCode_T& stdair::FareFamilyKey::getFamilyCode () const [inline]

Get the family code.

Definition at line 56 of file [FareFamilyKey.hpp](#).

Referenced by [stdair::FareFamily::getFamilyCode\(\)](#).

34.65.3.2 void stdair::FareFamilyKey::toStream (std::ostream & ioOut) const [virtual]

Dump a Business Object Key into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 37 of file [FareFamilyKey.cpp](#).

References [toString\(\)](#).

34.65.3.3 void stdair::FareFamilyKey::fromStream (std::istream & ioIn) [virtual]

Read a Business Object Key from an input stream.

Parameters

<i>istream&</i> the input stream.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 42 of file [FareFamilyKey.cpp](#).

34.65.3.4 const std::string stdair::FareFamilyKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 46 of file [FareFamilyKey.cpp](#).

Referenced by [stdair::FareFamily::describeKey\(\)](#), and [toStream\(\)](#).

34.65.3.5 `template<class Archive > void stdair::FareFamilyKey::serialize (Archive & ar, const unsigned int iFileVersion)`

Serialisation.

Definition at line 68 of file [FareFamilyKey.cpp](#).

34.65.4 Friends And Related Function Documentation

34.65.4.1 `friend class boost::serialization::access [friend]`

Definition at line 27 of file [FareFamilyKey.hpp](#).

The documentation for this struct was generated from the following files:

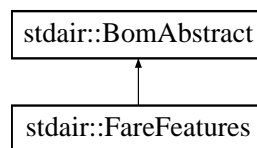
- [stdair/bom/FareFamilyKey.hpp](#)
- [stdair/bom/FareFamilyKey.cpp](#)

34.66 stdair::FareFeatures Class Reference

Class representing the actual attributes for a fare date-period.

```
#include <stdair/bom/FareFeatures.hpp>
```

Inheritance diagram for `stdair::FareFeatures`:



Public Types

- typedef [FareFeaturesKey](#) `Key_T`

Public Member Functions

- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- std::string [toString](#) () const

- const std::string [describeKey](#) () const
- const [Key_T](#) & [getKey](#) () const
- [BomAbstract](#) *const [getParent](#) () const
- const [HolderMap_T](#) & [getHolderMap](#) () const
- const [TripType_T](#) & [getTripType](#) () const
- const [DayDuration_T](#) & [getAdvancePurchase](#) () const
- const [SaturdayStay_T](#) & [getSaturdayStay](#) () const
- const [ChangeFees_T](#) & [getChangeFees](#) () const
- const [NonRefundable_T](#) & [getRefundableOption](#) () const
- const [DayDuration_T](#) & [getMinimumStay](#) () const
- bool [isTripTypeValid](#) (const [TripType_T](#) &) const
- bool [isStayDurationValid](#) (const [DayDuration_T](#) &) const
- bool [isAdvancePurchaseValid](#) (const [DateTime_T](#) & iBookingRequestDateTime, const [DateTime_T](#) & iFlightDateTime) const

Protected Member Functions

- [FareFeatures](#) (const [Key_T](#) &)
- virtual [~FareFeatures](#) ()

Protected Attributes

- [Key_T _key](#)
- [BomAbstract](#) * [_parent](#)
- [HolderMap_T _holderMap](#)

Friends

- class [FacBom](#)
- class [FacBomManager](#)

34.66.1 Detailed Description

Class representing the actual attributes for a fare date-period.

Definition at line 18 of file [FareFeatures.hpp](#).

34.66.2 Member Typedef Documentation

34.66.2.1 typedef FareFeaturesKey stdair::FareFeatures::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 27 of file [FareFeatures.hpp](#).

34.66.3 Constructor & Destructor Documentation

34.66.3.1 stdair::FareFeatures::FareFeatures (const Key_T & iKey) [protected]

Main constructor.

Definition at line 34 of file [FareFeatures.cpp](#).

34.66.3.2 stdair::FareFeatures::~~FareFeatures () [protected, virtual]

Destructor.

Definition at line 39 of file [FareFeatures.cpp](#).

34.66.4 Member Function Documentation

34.66.4.1 void stdair::FareFeatures::toStream (std::ostream & ioOut) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Implements [stdair::BomAbstract](#).

Definition at line 36 of file [FareFeatures.hpp](#).

References [toString\(\)](#).

34.66.4.2 void stdair::FareFeatures::fromStream (std::istream & iIn) [inline, virtual]

Read a Business Object from an input stream.

Parameters

<i>istream&</i> the input stream.

Implements [stdair::BomAbstract](#).

Definition at line 45 of file [FareFeatures.hpp](#).

34.66.4.3 std::string stdair::FareFeatures::toString () const [virtual]

Get the serialised version of the Business Object.

Implements [stdair::BomAbstract](#).

Definition at line 43 of file [FareFeatures.cpp](#).

References [describeKey\(\)](#).

Referenced by [toStream\(\)](#).

34.66.4.4 `const std::string stdair::FareFeatures::describeKey () const` `[inline]`

Get a string describing the key.

Definition at line 56 of file [FareFeatures.hpp](#).

References [_key](#), and [stdair::FareFeaturesKey::toString\(\)](#).

Referenced by [toString\(\)](#).

34.66.4.5 `const Key_T& stdair::FareFeatures::getKey () const` `[inline]`

Get the primary key (trip type, advance purchase,... ,cabin code).

Definition at line 66 of file [FareFeatures.hpp](#).

References [_key](#).

34.66.4.6 `BomAbstract* const stdair::FareFeatures::getParent () const` `[inline]`

Get a reference on the parent object instance.

Definition at line 73 of file [FareFeatures.hpp](#).

References [_parent](#).

34.66.4.7 `const HolderMap_T& stdair::FareFeatures::getHolderMap () const` `[inline]`

Get a reference on the children holder.

Definition at line 80 of file [FareFeatures.hpp](#).

References [_holderMap](#).

34.66.4.8 `const TripType_T& stdair::FareFeatures::getTripType () const` `[inline]`

Get the trip type.

Definition at line 87 of file [FareFeatures.hpp](#).

References [_key](#), and [stdair::FareFeaturesKey::getTripType\(\)](#).

Referenced by [isTripTypeValid\(\)](#).

34.66.4.9 `const DayDuration_T& stdair::FareFeatures::getAdvancePurchase () const`
`[inline]`

Get the fare day duration.

Definition at line 94 of file [FareFeatures.hpp](#).

References [_key](#), and [stdair::FareFeaturesKey::getAdvancePurchase\(\)](#).

Referenced by [isAdvancePurchaseValid\(\)](#).

34.66.4.10 `const SaturdayStay_T& stdair::FareFeatures::getSaturdayStay () const`
`[inline]`

Get the fare saturday stay option.

Definition at line 101 of file [FareFeatures.hpp](#).

References [_key](#), and [stdair::FareFeaturesKey::getSaturdayStay\(\)](#).

34.66.4.11 `const ChangeFees_T& stdair::FareFeatures::getChangeFees () const`
`[inline]`

Get the change fees criterion.

Definition at line 108 of file [FareFeatures.hpp](#).

References [_key](#), and [stdair::FareFeaturesKey::getChangeFees\(\)](#).

34.66.4.12 `const NonRefundable_T& stdair::FareFeatures::getRefundableOption () const`
`[inline]`

Get the refundable option.

Definition at line 115 of file [FareFeatures.hpp](#).

References [_key](#), and [stdair::FareFeaturesKey::getRefundableOption\(\)](#).

34.66.4.13 `const DayDuration_T& stdair::FareFeatures::getMinimumStay () const`
`[inline]`

Get the minimum stay.

Definition at line 122 of file [FareFeatures.hpp](#).

References [_key](#), and [stdair::FareFeaturesKey::getMinimumStay\(\)](#).

Referenced by [isStayDurationValid\(\)](#).

34.66.4.14 `bool stdair::FareFeatures::isTripTypeValid (const TripType_T &`
`iBookingRequestTripType) const`

Check whether the fare rule trip type corresponds to the booking request trip type.

Definition at line 51 of file [FareFeatures.cpp](#).

References [getTripType\(\)](#), [stdair::TRIP_TYPE_INBOUND](#), [stdair::TRIP_TYPE_OUTBOUND](#), and [stdair::TRIP_TYPE_ROUND_TRIP](#).

34.66.4.15 `bool stdair::FareFeatures::isStayDurationValid (const DayDuration_T &`
`iStayDuration) const`

Check whether a given stay duration is greater or equal to the minimum stay of the fare rule.

Definition at line 76 of file [FareFeatures.cpp](#).

References [getMinimumStay\(\)](#).

34.66.4.16 `bool stdair::FareFeatures::isAdvancePurchaseValid (const DateTime_T & iBookingRequestDateTime, const DateTime_T & iFlightDateTime) const`

Check whether a booking request date is valid compared the required advance purchase number of days of the fare rule.

Definition at line 89 of file [FareFeatures.cpp](#).

References [getAdvancePurchase\(\)](#).

34.66.5 Friends And Related Function Documentation

34.66.5.1 `friend class FacBom` [[friend](#)]

Definition at line 19 of file [FareFeatures.hpp](#).

34.66.5.2 `friend class FacBomManager` [[friend](#)]

Definition at line 20 of file [FareFeatures.hpp](#).

34.66.6 Member Data Documentation

34.66.6.1 `Key_T stdair::FareFeatures::_key` [[protected](#)]

Primary key (flight number and departure date).

Definition at line 175 of file [FareFeatures.hpp](#).

Referenced by [describeKey\(\)](#), [getAdvancePurchase\(\)](#), [getChangeFees\(\)](#), [getKey\(\)](#), [getMinimumStay\(\)](#), [getRefundableOption\(\)](#), [getSaturdayStay\(\)](#), and [getTripType\(\)](#).

34.66.6.2 `BomAbstract* stdair::FareFeatures::_parent` [[protected](#)]

Pointer on the parent class.

Definition at line 180 of file [FareFeatures.hpp](#).

Referenced by [getParent\(\)](#).

34.66.6.3 `HolderMap_T stdair::FareFeatures::_holderMap` [[protected](#)]

Map holding the children.

Definition at line 185 of file [FareFeatures.hpp](#).

Referenced by [getHolderMap\(\)](#).

The documentation for this class was generated from the following files:

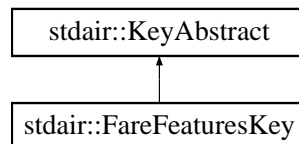
- [stdair/bom/FareFeatures.hpp](#)
- [stdair/bom/FareFeatures.cpp](#)

34.67 stdair::FareFeaturesKey Struct Reference

Key of date-period.

```
#include <stdair/bom/FareFeaturesKey.hpp>
```

Inheritance diagram for stdair::FareFeaturesKey:



Public Member Functions

- [FareFeaturesKey](#) (const [TripType_T](#) &, const [DayDuration_T](#) &, const [SaturdayStay_T](#) &, const [ChangeFees_T](#) &, const [NonRefundable_T](#) &, const [DayDuration_T](#) &)
- [FareFeaturesKey](#) (const [FareFeaturesKey](#) &)
- [~FareFeaturesKey](#) ()
- const [TripType_T](#) & [getTripType](#) () const
- const [DayDuration_T](#) & [getAdvancePurchase](#) () const
- const [SaturdayStay_T](#) & [getSaturdayStay](#) () const
- const [ChangeFees_T](#) & [getChangeFees](#) () const
- const [NonRefundable_T](#) & [getRefundableOption](#) () const
- const [DayDuration_T](#) & [getMinimumStay](#) () const
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- const std::string [toString](#) () const

34.67.1 Detailed Description

Key of date-period.

Definition at line 18 of file [FareFeaturesKey.hpp](#).

34.67.2 Constructor & Destructor Documentation

34.67.2.1 [stdair::FareFeaturesKey::FareFeaturesKey](#) (const [TripType_T](#) & *iTripType*, const [DayDuration_T](#) & *iAdvancePurchase*, const [SaturdayStay_T](#) & *iSaturdayStay*, const [ChangeFees_T](#) & *iChangeFees*, const [NonRefundable_T](#) & *iNonRefundable*, const [DayDuration_T](#) & *iMinimumStay*)

Main constructor.

Definition at line 26 of file [FareFeaturesKey.cpp](#).

34.67.2.2 `stdair::FareFeaturesKey::FareFeaturesKey (const FareFeaturesKey & iKey)`

Copy constructor.

Definition at line 38 of file [FareFeaturesKey.cpp](#).

34.67.2.3 `stdair::FareFeaturesKey::~FareFeaturesKey ()`

Destructor.

Definition at line 48 of file [FareFeaturesKey.cpp](#).

34.67.3 Member Function Documentation

34.67.3.1 `const TripType_T& stdair::FareFeaturesKey::getTripType () const` `[inline]`

Get the fare trip type.

Definition at line 39 of file [FareFeaturesKey.hpp](#).

Referenced by [stdair::FareFeatures::getTripType\(\)](#).

34.67.3.2 `const DayDuration_T& stdair::FareFeaturesKey::getAdvancePurchase () const` `[inline]`

Get the fare day duration.

Definition at line 46 of file [FareFeaturesKey.hpp](#).

Referenced by [stdair::FareFeatures::getAdvancePurchase\(\)](#).

34.67.3.3 `const SaturdayStay_T& stdair::FareFeaturesKey::getSaturdayStay () const` `[inline]`

Get the fare saturday stay option.

Definition at line 53 of file [FareFeaturesKey.hpp](#).

Referenced by [stdair::FareFeatures::getSaturdayStay\(\)](#).

34.67.3.4 `const ChangeFees_T& stdair::FareFeaturesKey::getChangeFees () const` `[inline]`

Get the change fees criterion.

Definition at line 60 of file [FareFeaturesKey.hpp](#).

Referenced by [stdair::FareFeatures::getChangeFees\(\)](#).

34.67.3.5 `const NonRefundable_T& stdair::FareFeaturesKey::getRefundableOption ()` `const` `[inline]`

Get the refundable option.

Definition at line 67 of file [FareFeaturesKey.hpp](#).

Referenced by [stdair::FareFeatures::getRefundableOption\(\)](#).

34.67.3.6 `const DayDuration_T& stdair::FareFeaturesKey::getMinimumStay () const`
[inline]

Get the minimum stay.

Definition at line 74 of file [FareFeaturesKey.hpp](#).

Referenced by [stdair::FareFeatures::getMinimumStay\(\)](#).

34.67.3.7 `void stdair::FareFeaturesKey::toStream (std::ostream & ioOut) const`
[virtual]

Dump a Business Object Key into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 52 of file [FareFeaturesKey.cpp](#).

References [toString\(\)](#).

34.67.3.8 `void stdair::FareFeaturesKey::fromStream (std::istream & ioIn)` [virtual]

Read a Business Object Key from an input stream.

Parameters

<i>istream&</i> the input stream.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 57 of file [FareFeaturesKey.cpp](#).

34.67.3.9 `const std::string stdair::FareFeaturesKey::toString () const` [virtual]

Get the serialised version of the Business Object Key. That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 61 of file [FareFeaturesKey.cpp](#).

Referenced by [stdair::FareFeatures::describeKey\(\)](#), and [toStream\(\)](#).

The documentation for this struct was generated from the following files:

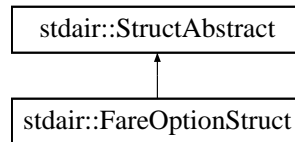
- [stdair/bom/FareFeaturesKey.hpp](#)
- [stdair/bom/FareFeaturesKey.cpp](#)

34.68 stdair::FareOptionStruct Struct Reference

Structure holding the elements of a fare option.

```
#include <stdair/bom/FareOptionStruct.hpp>
```

Inheritance diagram for stdair::FareOptionStruct:



Public Member Functions

- const [ClassList_StringList_T](#) & [getClassPath](#) () const
- const [Fare_T](#) & [getFare](#) () const
- const [Availability_T](#) & [getAvailability](#) () const
- const [ChangeFees_T](#) [getChangeFees](#) () const
- const [NonRefundable_T](#) [getNonRefundable](#) () const
- const [SaturdayStay_T](#) [getSaturdayStay](#) () const
- void [addClassList](#) (const std::string)
- void [emptyClassList](#) ()
- void [setFare](#) (const [Fare_T](#) &iFare)
- void [setAvailability](#) (const [Availability_T](#) &iAvl)
- void [setChangeFees](#) (const [ChangeFees_T](#) iRes)
- void [setNonRefundable](#) (const [NonRefundable_T](#) iRes)
- void [setSaturdayStay](#) (const [SaturdayStay_T](#) iRes)
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- const std::string [describe](#) () const
- const std::string [display](#) () const
- [FareOptionStruct](#) ()
- [FareOptionStruct](#) (const std::string &iClassPath, const [Fare_T](#) &, const [ChangeFees_T](#) &, const [NonRefundable_T](#) &, const [SaturdayStay_T](#) &)
- [FareOptionStruct](#) (const [FareOptionStruct](#) &)
- [~FareOptionStruct](#) ()

34.68.1 Detailed Description

Structure holding the elements of a fare option.

Definition at line 20 of file [FareOptionStruct.hpp](#).

34.68.2 Constructor & Destructor Documentation

34.68.2.1 stdair::FareOptionStruct::FareOptionStruct ()

Default constructor.

Definition at line 14 of file [FareOptionStruct.cpp](#).

34.68.2.2 `stdair::FareOptionStruct::FareOptionStruct (const std::string & iClassPath, const Fare_T & iFare, const ChangeFees_T & iChangeFee, const NonRefundable_T & iNonRefundable, const SaturdayStay_T & iSaturdayNightStay)`

Main constructor.

Definition at line 26 of file [FareOptionStruct.cpp](#).

34.68.2.3 `stdair::FareOptionStruct::FareOptionStruct (const FareOptionStruct & iFO)`

Copy constructor.

Definition at line 19 of file [FareOptionStruct.cpp](#).

34.68.2.4 `stdair::FareOptionStruct::~~FareOptionStruct ()`

Destructor.

Definition at line 38 of file [FareOptionStruct.cpp](#).

34.68.3 Member Function Documentation

34.68.3.1 `const ClassList_StringList_T& stdair::FareOptionStruct::getClassPath () const`
[inline]

Get the class-path.

Definition at line 24 of file [FareOptionStruct.hpp](#).

34.68.3.2 `const Fare_T& stdair::FareOptionStruct::getFare () const` [inline]

Get the fare value.

Definition at line 29 of file [FareOptionStruct.hpp](#).

34.68.3.3 `const Availability_T& stdair::FareOptionStruct::getAvailability () const`
[inline]

Get the availability.

Definition at line 34 of file [FareOptionStruct.hpp](#).

34.68.3.4 `const ChangeFees_T stdair::FareOptionStruct::getChangeFees () const`
[inline]

Get the change fees.

Definition at line 39 of file [FareOptionStruct.hpp](#).

34.68.3.5 `const NonRefundable_T stdair::FareOptionStruct::getNonRefundable () const`
[inline]

State whether the ticket is refundable.

Definition at line 44 of file [FareOptionStruct.hpp](#).

34.68.3.6 `const SaturdayStay_T stdair::FareOptionStruct::getSaturdayStay () const`
[inline]

State whether there is a condition on the saturday night stay.

Definition at line 49 of file [FareOptionStruct.hpp](#).

34.68.3.7 `void stdair::FareOptionStruct::addClassList (const std::string iClassCodeList)`

Set the class-path.

Definition at line 93 of file [FareOptionStruct.cpp](#).

34.68.3.8 `void stdair::FareOptionStruct::emptyClassList ()`

Empty the class-path.

Definition at line 98 of file [FareOptionStruct.cpp](#).

34.68.3.9 `void stdair::FareOptionStruct::setFare (const Fare_T & iFare)` [inline]

Set the fare value.

Definition at line 63 of file [FareOptionStruct.hpp](#).

34.68.3.10 `void stdair::FareOptionStruct::setAvailability (const Availability_T & iAvl)`
[inline]

Set the availability.

Definition at line 68 of file [FareOptionStruct.hpp](#).

34.68.3.11 `void stdair::FareOptionStruct::setChangeFees (const ChangeFees_T iRes)`
[inline]

Set the change fees.

Definition at line 73 of file [FareOptionStruct.hpp](#).

34.68.3.12 `void stdair::FareOptionStruct::setNonRefundable (const NonRefundable_T iRes)`
[inline]

Set the flag for the ticket refundability.

Definition at line 78 of file [FareOptionStruct.hpp](#).

34.68.3.13 `void stdair::FareOptionStruct::setSaturdayStay (const SaturdayStay_T iRes)`
[inline]

Set the flag for the saturday night stay condition.

Definition at line 83 of file [FareOptionStruct.hpp](#).

34.68.3.14 `void stdair::FareOptionStruct::toStream (std::ostream & ioOut) const`

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Reimplemented from [stdair::StructAbstract](#).

Definition at line 42 of file [FareOptionStruct.cpp](#).

References [describe\(\)](#).

34.68.3.15 void stdair::FareOptionStruct::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object from an input stream.

Parameters

<i>istream&</i> the input stream.

Reimplemented from [stdair::StructAbstract](#).

Definition at line 47 of file [FareOptionStruct.cpp](#).

34.68.3.16 const std::string stdair::FareOptionStruct::describe () const [virtual]

Display of the structure.

Implements [stdair::StructAbstract](#).

Definition at line 51 of file [FareOptionStruct.cpp](#).

Referenced by [stdair::TravelSolutionStruct::describe\(\)](#), and [toStream\(\)](#).

34.68.3.17 const std::string stdair::FareOptionStruct::display () const

Display of the structure.

Definition at line 73 of file [FareOptionStruct.cpp](#).

Referenced by [stdair::TravelSolutionStruct::display\(\)](#).

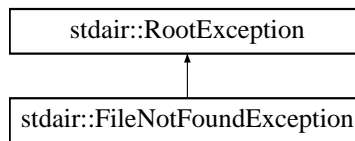
The documentation for this struct was generated from the following files:

- [stdair/bom/FareOptionStruct.hpp](#)
- [stdair/bom/FareOptionStruct.cpp](#)

34.69 stdair::FileNotFoundException Class Reference

```
#include <stdair/stdair_exceptions.hpp>
```

Inheritance diagram for stdair::FileNotFoundException:



Public Member Functions

- [FileNotFoundException](#) (const std::string &iWhat)
- const char * [what](#) () const throw ()

Protected Attributes

- std::string [_what](#)

34.69.1 Detailed Description

File not found.

Definition at line 50 of file [stdair_exceptions.hpp](#).

34.69.2 Constructor & Destructor Documentation

34.69.2.1 `stdair::FileNotFoundException::FileNotFoundException (const std::string & iWhat)`
[inline]

Constructor.

Definition at line 53 of file [stdair_exceptions.hpp](#).

34.69.3 Member Function Documentation

34.69.3.1 `const char* stdair::RootException::what () const throw ()` [inline, inherited]

Give the details of the exception.

Definition at line 38 of file [stdair_exceptions.hpp](#).

References [stdair::RootException::_what](#).

34.69.4 Member Data Documentation

34.69.4.1 `std::string stdair::RootException::_what` [protected, inherited]

Details for the exception.

Definition at line 46 of file [stdair_exceptions.hpp](#).

Referenced by [stdair::RootException::what\(\)](#).

The documentation for this class was generated from the following file:

- [stdair/stdair_exceptions.hpp](#)

34.70 stdair::FlagSaver Struct Reference

Public Member Functions

- [FlagSaver](#) (std::ostream &oStream)
- [~FlagSaver](#) ()

34.70.1 Detailed Description

Helper singleton structure to store the current formatting flags of any given output stream. The flags are re-set at the structure destruction.

Definition at line 46 of file [BomJSONExport.cpp](#).

34.70.2 Constructor & Destructor Documentation

34.70.2.1 stdair::FlagSaver::FlagSaver (std::ostream & oStream) [inline]

Constructor.

Definition at line 49 of file [BomJSONExport.cpp](#).

34.70.2.2 stdair::FlagSaver::~~FlagSaver () [inline]

Destructor.

Definition at line 54 of file [BomJSONExport.cpp](#).

The documentation for this struct was generated from the following file:

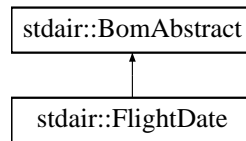
- [stdair/bom/BomJSONExport.cpp](#)

34.71 stdair::FlightDate Class Reference

Class representing the actual attributes for an airline flight-date.

```
#include <stdair/bom/FlightDate.hpp>
```

Inheritance diagram for stdair::FlightDate:



Public Types

- typedef [FlightDateKey](#) [Key_T](#)

Public Member Functions

- const [Key_T](#) & [getKey](#) () const
- [BomAbstract](#) *const [getParent](#) () const
- const [FlightNumber_T](#) & [getFlightNumber](#) () const
- const [Date_T](#) & [getDepartureDate](#) () const
- const [AirlineCode_T](#) & [getAirlineCode](#) () const
- const [HolderMap_T](#) & [getHolderMap](#) () const
- [LegDate](#) * [getLegDate](#) (const std::string &iLegDateKeyStr) const
- [LegDate](#) * [getLegDate](#) (const [LegDateKey](#) &) const
- [SegmentDate](#) * [getSegmentDate](#) (const std::string &iSegmentDateKeyStr) const
- [SegmentDate](#) * [getSegmentDate](#) (const [SegmentDateKey](#) &) const
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- std::string [toString](#) () const
- const std::string [describeKey](#) () const
- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int iFileVersion)

Protected Member Functions

- [FlightDate](#) (const [Key_T](#) &)
- virtual [~FlightDate](#) ()

Protected Attributes

- [Key_T _key](#)
- [BomAbstract](#) * [_parent](#)
- [HolderMap_T _holderMap](#)

Friends

- class [FacBom](#)
- class [FacBomManager](#)
- class [boost::serialization::access](#)

34.71.1 Detailed Description

Class representing the actual attributes for an airline flight-date.

Definition at line 35 of file [FlightDate.hpp](#).

34.71.2 Member Typedef Documentation

34.71.2.1 typedef FlightDateKey stdair::FlightDate::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 45 of file [FlightDate.hpp](#).

34.71.3 Constructor & Destructor Documentation

34.71.3.1 stdair::FlightDate::FlightDate (const Key_T & iKey) [protected]

Main constructor.

Definition at line 30 of file [FlightDate.cpp](#).

34.71.3.2 stdair::FlightDate::~~FlightDate () [protected, virtual]

Destructor.

Definition at line 34 of file [FlightDate.cpp](#).

34.71.4 Member Function Documentation

34.71.4.1 const Key_T& stdair::FlightDate::getKey () const [inline]

Get the flight-date key.

Definition at line 51 of file [FlightDate.hpp](#).

References [_key](#).

34.71.4.2 BomAbstract* const stdair::FlightDate::getParent () const [inline]

Get the parent object.

Definition at line 56 of file [FlightDate.hpp](#).

References [_parent](#).

Referenced by [getAirlineCode\(\)](#).

34.71.4.3 const FlightNumber_T& stdair::FlightDate::getFlightNumber () const [inline]

Get the flight number (part of the primary key).

Definition at line 61 of file [FlightDate.hpp](#).

References [_key](#), and [stdair::FlightDateKey::getFlightNumber\(\)](#).

Referenced by [stdair::BomJSONExport::jsonExport\(\)](#).

34.71.4.4 `const Date_T& stdair::FlightDate::getDepartureDate () const` `[inline]`

Get the flight date (part of the primary key).

Definition at line 66 of file [FlightDate.hpp](#).

References [_key](#), and [stdair::FlightDateKey::getDepartureDate\(\)](#).

Referenced by [stdair::BomJSONExport::jsonExport\(\)](#).

34.71.4.5 `const AirlineCode_T & stdair::FlightDate::getAirlineCode () const`

Get the airline code (key of the parent object).

Note

That method assumes that the parent object derives from the [Inventory](#) class, as it needs to have access to the [getAirlineCode\(\)](#) method.

Definition at line 38 of file [FlightDate.cpp](#).

References [stdair::Inventory::getAirlineCode\(\)](#), and [getParent\(\)](#).

Referenced by [stdair::LegDate::getAirlineCode\(\)](#), and [stdair::BomJSONExport::jsonExport\(\)](#).

34.71.4.6 `const HolderMap_T& stdair::FlightDate::getHolderMap () const` `[inline]`

Get the map of children holders.

Definition at line 82 of file [FlightDate.hpp](#).

References [_holderMap](#).

34.71.4.7 `LegDate * stdair::FlightDate::getLegDate (const std::string & iLegDateKeyStr) const`

Get a pointer on the [LegDate](#) object corresponding to the given key.

Note

The [LegDate](#) object can be inherited from, if needed. In that case, a `dynamic_cast<>` may be needed.

Parameters

<code>const std::string&</code> The leg-date key.

Returns

`LegDate*` Found [LegDate](#) object. NULL if not found.

Definition at line 53 of file [FlightDate.cpp](#).

Referenced by [getLegDate\(\)](#), and [stdair::BomRetriever::retrieveDummyLegCabin\(\)](#).

34.71.4.8 `LegDate * stdair::FlightDate::getLegDate (const LegDateKey & iLegDateKey) const`

Get a pointer on the [LegDate](#) object corresponding to the given key.

Note

The [LegDate](#) object can be inherited from, if needed. In that case, a `dynamic_cast<>` may be needed.

Parameters

<code>const</code> LegDateKey & The leg-date key
--

Returns

`LegDate*` Found [LegDate](#) object. NULL if not found.

Definition at line 60 of file [FlightDate.cpp](#).

References [getLegDate\(\)](#), and [stdair::LegDateKey::toString\(\)](#).

34.71.4.9 `SegmentDate * stdair::FlightDate::getSegmentDate (const std::string & iSegmentDateKeyStr) const`

Get a pointer on the [SegmentDate](#) object corresponding to the given key.

Note

The [SegmentDate](#) object can be inherited from, if needed. In that case, a `dynamic_cast<>` may be needed.

Parameters

<code>const</code> <code>std::string</code> & The segment-date key.

Returns

`SegmentDate*` Found [SegmentDate](#) object. NULL if not found.

Definition at line 66 of file [FlightDate.cpp](#).

Referenced by [getSegmentDate\(\)](#), [stdair::BomRetriever::retrieveDummySegmentCabin\(\)](#), [stdair::BomRetriever::retrieveSegmentDateFromKey\(\)](#), and [stdair::BomRetriever::retrieveSegmentDateFromLongKey\(\)](#).

34.71.4.10 `SegmentDate * stdair::FlightDate::getSegmentDate (const SegmentDateKey & iSegmentDateKey) const`

Get a pointer on the [SegmentDate](#) object corresponding to the given key.

Note

The [SegmentDate](#) object can be inherited from, if needed. In that case, a `dynamic_cast<>`

cast<> may be needed.

Parameters

<i>const</i> SegmentDateKey & The segment-date key
--

Returns

SegmentDate* Found [SegmentDate](#) object. NULL if not found.

Definition at line 74 of file [FlightDate.cpp](#).

References [getSegmentDate\(\)](#), and [stdair::SegmentDateKey::toString\(\)](#).

34.71.4.11 void stdair::FlightDate::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Implements [stdair::BomAbstract](#).

Definition at line 141 of file [FlightDate.hpp](#).

References [toString\(\)](#).

34.71.4.12 void stdair::FlightDate::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters

<i>istream&</i> the input stream.

Implements [stdair::BomAbstract](#).

Definition at line 150 of file [FlightDate.hpp](#).

34.71.4.13 std::string stdair::FlightDate::toString () const [virtual]

Get the serialised version of the Business Object.

Implements [stdair::BomAbstract](#).

Definition at line 46 of file [FlightDate.cpp](#).

References [describeKey\(\)](#).

Referenced by [toStream\(\)](#).

34.71.4.14 `const std::string stdair::FlightDate::describeKey () const` `[inline]`

Get a string describing the key.

Definition at line 161 of file [FlightDate.hpp](#).

References [_key](#), and [stdair::FlightDateKey::toString\(\)](#).

Referenced by [toString\(\)](#).

34.71.4.15 `template<class Archive > void stdair::FlightDate::serialize (Archive & ar, const unsigned int iFileVersion)`

Serialisation.

Definition at line 187 of file [CmdBomSerialiser.cpp](#).

References [_key](#).

34.71.5 Friends And Related Function Documentation

34.71.5.1 `friend class FacBom` `[friend]`

Definition at line 36 of file [FlightDate.hpp](#).

34.71.5.2 `friend class FacBomManager` `[friend]`

Definition at line 37 of file [FlightDate.hpp](#).

34.71.5.3 `friend class boost::serialization::access` `[friend]`

Definition at line 38 of file [FlightDate.hpp](#).

34.71.6 Member Data Documentation

34.71.6.1 `Key_T stdair::FlightDate::_key` `[protected]`

Primary key (flight number and departure date).

Definition at line 215 of file [FlightDate.hpp](#).

Referenced by [describeKey\(\)](#), [getDepartureDate\(\)](#), [getFlightNumber\(\)](#), [getKey\(\)](#), and [serialize\(\)](#).

34.71.6.2 `BomAbstract* stdair::FlightDate::_parent` `[protected]`

Pointer on the parent class ([Inventory](#)).

Definition at line 220 of file [FlightDate.hpp](#).

Referenced by [getParent\(\)](#).

34.71.6.3 `HolderMap_T stdair::FlightDate::_holderMap` `[protected]`

Map holding the children ([SegmentDate](#) and [LegDate](#) objects).

Definition at line 225 of file [FlightDate.hpp](#).

Referenced by [getHolderMap\(\)](#).

The documentation for this class was generated from the following files:

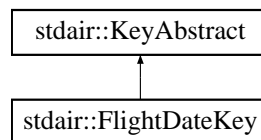
- [stdair/bom/FlightDate.hpp](#)
- [stdair/bom/FlightDate.cpp](#)
- [stdair/command/CmdBomSerialiser.cpp](#)

34.72 stdair::FlightDateKey Struct Reference

Key of a given flight-date, made of a flight number and a departure date.

```
#include <stdair/bom/FlightDateKey.hpp>
```

Inheritance diagram for stdair::FlightDateKey:



Public Member Functions

- [FlightDateKey](#) (const [FlightNumber_T](#) &, const [Date_T](#) &)
- [FlightDateKey](#) (const [FlightDateKey](#) &)
- [~FlightDateKey](#) ()
- const [FlightNumber_T](#) & [getFlightNumber](#) () const
- const [Date_T](#) & [getDepartureDate](#) () const
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- const std::string [toString](#) () const
- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int iFileVersion)

Friends

- class [boost::serialization::access](#)

34.72.1 Detailed Description

Key of a given flight-date, made of a flight number and a departure date.

Definition at line 28 of file [FlightDateKey.hpp](#).

34.72.2 Constructor & Destructor Documentation

34.72.2.1 `stdair::FlightDateKey::FlightDateKey (const FlightNumber_T & iFlightNumber, const Date_T & iFlightDate)`

Constructor.

Definition at line 28 of file [FlightDateKey.cpp](#).

34.72.2.2 `stdair::FlightDateKey::FlightDateKey (const FlightDateKey & iKey)`

Copy constructor.

Definition at line 34 of file [FlightDateKey.cpp](#).

34.72.2.3 `stdair::FlightDateKey::~~FlightDateKey ()`

Destructor.

Definition at line 39 of file [FlightDateKey.cpp](#).

34.72.3 Member Function Documentation

34.72.3.1 `const FlightNumber_T& stdair::FlightDateKey::getFlightNumber () const [inline]`

Get the flight number.

Definition at line 58 of file [FlightDateKey.hpp](#).

Referenced by [stdair::FlightDate::getFlightNumber\(\)](#).

34.72.3.2 `const Date_T& stdair::FlightDateKey::getDepartureDate () const [inline]`

Get the departure date of the (first leg of the) flight.

Definition at line 63 of file [FlightDateKey.hpp](#).

Referenced by [stdair::OnDDateKey::getDate\(\)](#), and [stdair::FlightDate::getDepartureDate\(\)](#).

34.72.3.3 `void stdair::FlightDateKey::toStream (std::ostream & ioOut) const [virtual]`

Dump a Business Object Key into an output stream.

Parameters

<code>ostream&</code> the output stream.
--

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 43 of file [FlightDateKey.cpp](#).

References [toString\(\)](#).

34.72.3.4 void stdair::FlightDateKey::fromStream (std::istream & *ioln*) [virtual]

Read a Business Object Key from an input stream.

Parameters

<i>istream&</i> the input stream.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 48 of file [FlightDateKey.cpp](#).

34.72.3.5 const std::string stdair::FlightDateKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 52 of file [FlightDateKey.cpp](#).

References [stdair::DEFAULT_KEY_SUB_FLD_DELIMITER](#).

Referenced by [stdair::FlightDate::describeKey\(\)](#), [stdair::Inventory::getFlightDate\(\)](#), [stdair::BomRetriever::retrieveSegment\(\)](#) and [toString\(\)](#).

34.72.3.6 template<class Archive > void stdair::FlightDateKey::serialize (Archive & *ar*, const unsigned int *iFileVersion*)

Serialisation.

Definition at line 77 of file [FlightDateKey.cpp](#).

34.72.4 Friends And Related Function Documentation

34.72.4.1 friend class boost::serialization::access [friend]

Definition at line 29 of file [FlightDateKey.hpp](#).

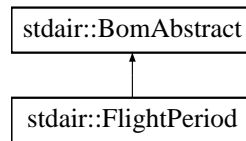
The documentation for this struct was generated from the following files:

- [stdair/bom/FlightDateKey.hpp](#)
- [stdair/bom/FlightDateKey.cpp](#)

34.73 stdair::FlightPeriod Class Reference

```
#include <stdair/bom/FlightPeriod.hpp>
```

Inheritance diagram for stdair::FlightPeriod:



Public Types

- typedef [FlightPeriodKey](#) [Key_T](#)

Public Member Functions

- const [Key_T](#) & [getKey](#) () const
- [BomAbstract](#) *const [getParent](#) () const
- const [FlightNumber_T](#) & [getFlightNumber](#) () const
- const [PeriodStruct](#) & [getPeriod](#) () const
- const [HolderMap_T](#) & [getHolderMap](#) () const
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- std::string [toString](#) () const
- const std::string [describeKey](#) () const

Protected Member Functions

- [FlightPeriod](#) (const [Key_T](#) &)
- [FlightPeriod](#) (const [FlightPeriod](#) &)
- [~FlightPeriod](#) ()

Protected Attributes

- [Key_T_key](#)
- [BomAbstract](#) * [_parent](#)
- [HolderMap_T_holderMap](#)

Friends

- class [FacBom](#)
- class [FacBomManager](#)

34.73.1 Detailed Description

Class representing the actual attributes for an airline flight-period.

Definition at line 15 of file [FlightPeriod.hpp](#).

34.73.2 Member Typedef Documentation

34.73.2.1 typedef FlightPeriodKey stdair::FlightPeriod::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 22 of file [FlightPeriod.hpp](#).

34.73.3 Constructor & Destructor Documentation

34.73.3.1 stdair::FlightPeriod::FlightPeriod (const Key_T & iKey) [protected]

Default constructors.

Definition at line 12 of file [FlightPeriod.cpp](#).

34.73.3.2 stdair::FlightPeriod::FlightPeriod (const FlightPeriod &) [protected]

34.73.3.3 stdair::FlightPeriod::~~FlightPeriod () [protected]

Destructor.

Definition at line 17 of file [FlightPeriod.cpp](#).

34.73.4 Member Function Documentation

34.73.4.1 const Key_T& stdair::FlightPeriod::getKey () const [inline]

Get the flight-period key.

Definition at line 27 of file [FlightPeriod.hpp](#).

References [_key](#).

34.73.4.2 BomAbstract* const stdair::FlightPeriod::getParent () const [inline]

Get the parent object.

Definition at line 30 of file [FlightPeriod.hpp](#).

References [_parent](#).

34.73.4.3 const FlightNumber_T& stdair::FlightPeriod::getFlightNumber () const [inline]

Get the flight number (part of the primary key).

Definition at line 33 of file [FlightPeriod.hpp](#).

References [_key](#), and [stdair::FlightPeriodKey::getFlightNumber\(\)](#).

34.73.4.4 const PeriodStruct& stdair::FlightPeriod::getPeriod () const [inline]

Get the departure period (part of the key).

Definition at line 38 of file [FlightPeriod.hpp](#).

References [_key](#), and [stdair::FlightPeriodKey::getPeriod\(\)](#).

34.73.4.5 `const HolderMap_T& stdair::FlightPeriod::getHolderMap () const` `[inline]`

Get the map of children holders.

Definition at line 41 of file [FlightPeriod.hpp](#).

References [_holderMap](#).

34.73.4.6 `void stdair::FlightPeriod::toStream (std::ostream & ioOut) const` `[inline, virtual]`

Dump a Business Object into an output stream.

Parameters

<code>ostream&</code> the output stream.
--

Implements [stdair::BomAbstract](#).

Definition at line 48 of file [FlightPeriod.hpp](#).

References [toString\(\)](#).

34.73.4.7 `void stdair::FlightPeriod::fromStream (std::istream & ioln)` `[inline, virtual]`

Read a Business Object from an input stream.

Parameters

<code>istream&</code> the input stream.

Implements [stdair::BomAbstract](#).

Definition at line 52 of file [FlightPeriod.hpp](#).

34.73.4.8 `std::string stdair::FlightPeriod::toString () const` `[virtual]`

Get the serialised version of the Business Object.

Implements [stdair::BomAbstract](#).

Definition at line 21 of file [FlightPeriod.cpp](#).

References [describeKey\(\)](#).

Referenced by [toStream\(\)](#).

34.73.4.9 `const std::string stdair::FlightPeriod::describeKey () const` `[inline]`

Get a string describing the key.

Definition at line 58 of file [FlightPeriod.hpp](#).

References [_key](#), and [stdair::FlightPeriodKey::toString\(\)](#).

Referenced by [toString\(\)](#).

34.73.5 Friends And Related Function Documentation

34.73.5.1 friend class FacBom [friend]

Definition at line 16 of file [FlightPeriod.hpp](#).

34.73.5.2 friend class FacBomManager [friend]

Definition at line 17 of file [FlightPeriod.hpp](#).

34.73.6 Member Data Documentation

34.73.6.1 Key_T stdair::FlightPeriod::_key [protected]

Definition at line 69 of file [FlightPeriod.hpp](#).

Referenced by [describeKey\(\)](#), [getFlightNumber\(\)](#), [getKey\(\)](#), and [getPeriod\(\)](#).

34.73.6.2 BomAbstract* stdair::FlightPeriod::_parent [protected]

Definition at line 70 of file [FlightPeriod.hpp](#).

Referenced by [getParent\(\)](#).

34.73.6.3 HolderMap_T stdair::FlightPeriod::_holderMap [protected]

Definition at line 71 of file [FlightPeriod.hpp](#).

Referenced by [getHolderMap\(\)](#).

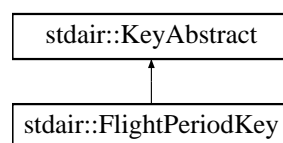
The documentation for this class was generated from the following files:

- [stdair/bom/FlightPeriod.hpp](#)
- [stdair/bom/FlightPeriod.cpp](#)

34.74 stdair::FlightPeriodKey Struct Reference

```
#include <stdair/bom/FlightPeriodKey.hpp>
```

Inheritance diagram for stdair::FlightPeriodKey:



Public Member Functions

- [FlightPeriodKey](#) (const [FlightNumber_T](#) &, const [PeriodStruct](#) &)
- [FlightPeriodKey](#) (const [FlightPeriodKey](#) &)
- [~FlightPeriodKey](#) ()
- const [FlightNumber_T](#) & [getFlightNumber](#) () const
- const [PeriodStruct](#) & [getPeriod](#) () const
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- const std::string [toString](#) () const

34.74.1 Detailed Description

Key of flight-period.

Definition at line 13 of file [FlightPeriodKey.hpp](#).

34.74.2 Constructor & Destructor Documentation

34.74.2.1 `stdair::FlightPeriodKey::FlightPeriodKey (const FlightNumber_T & iFlightNumber, const PeriodStruct & iPeriod)`

Constructors.

Definition at line 10 of file [FlightPeriodKey.cpp](#).

34.74.2.2 `stdair::FlightPeriodKey::FlightPeriodKey (const FlightPeriodKey & iKey)`

Definition at line 16 of file [FlightPeriodKey.cpp](#).

34.74.2.3 `stdair::FlightPeriodKey::~~FlightPeriodKey ()`

Destructor.

Definition at line 21 of file [FlightPeriodKey.cpp](#).

34.74.3 Member Function Documentation

34.74.3.1 `const FlightNumber_T & stdair::FlightPeriodKey::getFlightNumber () const`
`[inline]`

Get the flight number.

Definition at line 28 of file [FlightPeriodKey.hpp](#).

Referenced by [stdair::FlightPeriod::getFlightNumber\(\)](#).

34.74.3.2 `const PeriodStruct& stdair::FlightPeriodKey::getPeriod () const` `[inline]`

Get the active days-of-week.

Definition at line 33 of file [FlightPeriodKey.hpp](#).

Referenced by [stdair::FlightPeriod::getPeriod\(\)](#).

```
34.74.3.3 void stdair::FlightPeriodKey::toStream ( std::ostream & ioOut ) const
          [virtual]
```

Dump a Business Object Key into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 25 of file [FlightPeriodKey.cpp](#).

References [toString\(\)](#).

```
34.74.3.4 void stdair::FlightPeriodKey::fromStream ( std::istream & ioln ) [virtual]
```

Read a Business Object Key from an input stream.

Parameters

<i>istream&</i> the input stream.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 30 of file [FlightPeriodKey.cpp](#).

```
34.74.3.5 const std::string stdair::FlightPeriodKey::toString ( ) const [virtual]
```

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-period.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 34 of file [FlightPeriodKey.cpp](#).

References [stdair::PeriodStruct::describeShort\(\)](#).

Referenced by [stdair::FlightPeriod::describeKey\(\)](#), and [toStream\(\)](#).

The documentation for this struct was generated from the following files:

- [stdair/bom/FlightPeriodKey.hpp](#)
- [stdair/bom/FlightPeriodKey.cpp](#)

34.75 FloatingPoint< RawType > Class Template Reference

```
#include <stdair/basic/float_utils_google.hpp>
```

Public Types

- typedef [TypeWithSize](#)< sizeof(RawType)>::UInt [Bits](#)

Public Member Functions

- [FloatingPoint](#) (const RawType &x)
- const [Bits](#) & [bits](#) () const
- [Bits](#) [exponent_bits](#) () const
- [Bits](#) [fraction_bits](#) () const
- [Bits](#) [sign_bit](#) () const
- bool [is_nan](#) () const
- bool [AlmostEquals](#) (const [FloatingPoint](#) &rhs) const

Static Public Member Functions

- static RawType [ReinterpretBits](#) (const [Bits](#) bits)
- static RawType [Infinity](#) ()

Static Public Attributes

- static const size_t [kBitCount](#) = 8*sizeof(RawType)
- static const size_t [kFractionBitCount](#)
- static const size_t [kExponentBitCount](#) = [kBitCount](#) - 1 - [kFractionBitCount](#)
- static const [Bits](#) [kSignBitMask](#) = static_cast<[Bits](#)>(1) << ([kBitCount](#) - 1)
- static const [Bits](#) [kFractionBitMask](#)
- static const [Bits](#) [kExponentBitMask](#) = ~([kSignBitMask](#) | [kFractionBitMask](#))
- static const size_t [kMaxUlp](#) = 4

34.75.1 Detailed Description

template<typename RawType>class FloatingPoint< RawType >

Definition at line 117 of file [float_utils_google.hpp](#).

34.75.2 Member Typedef Documentation

34.75.2.1 template<typename RawType> typedef [TypeWithSize](#)<sizeof(RawType)>::UInt
[FloatingPoint](#)< RawType >::Bits

Definition at line 121 of file [float_utils_google.hpp](#).

34.75.3 Constructor & Destructor Documentation

34.75.3.1 `template<typename RawType> FloatingPoint< RawType >::FloatingPoint (const RawType & x) [inline, explicit]`

Definition at line 165 of file [float_utils_google.hpp](#).

34.75.4 Member Function Documentation

34.75.4.1 `template<typename RawType> static RawType FloatingPoint< RawType >::ReinterpretBits (const Bits bits) [inline, static]`

Definition at line 172 of file [float_utils_google.hpp](#).

References [FloatingPoint< RawType >::bits\(\)](#).

Referenced by [FloatingPoint< RawType >::Infinity\(\)](#).

34.75.4.2 `template<typename RawType> static RawType FloatingPoint< RawType >::Infinity () [inline, static]`

Definition at line 179 of file [float_utils_google.hpp](#).

References [FloatingPoint< RawType >::kExponentBitMask](#), and [FloatingPoint< RawType >::ReinterpretBits\(\)](#).

34.75.4.3 `template<typename RawType> const Bits& FloatingPoint< RawType >::bits () const [inline]`

Definition at line 186 of file [float_utils_google.hpp](#).

Referenced by [FloatingPoint< RawType >::ReinterpretBits\(\)](#).

34.75.4.4 `template<typename RawType> Bits FloatingPoint< RawType >::exponent.bits () const [inline]`

Definition at line 189 of file [float_utils_google.hpp](#).

References [FloatingPoint< RawType >::kExponentBitMask](#).

Referenced by [FloatingPoint< RawType >::is_nan\(\)](#).

34.75.4.5 `template<typename RawType> Bits FloatingPoint< RawType >::fraction.bits () const [inline]`

Definition at line 192 of file [float_utils_google.hpp](#).

References [FloatingPoint< RawType >::kFractionBitMask](#).

Referenced by [FloatingPoint< RawType >::is_nan\(\)](#).

34.75.4.6 `template<typename RawType> Bits FloatingPoint< RawType >::sign_bit () const [inline]`

Definition at line 195 of file [float_utils_google.hpp](#).

References [FloatingPoint< RawType >::kSignBitMask](#).

34.75.4.7 `template<typename RawType> bool FloatingPoint< RawType >::is_nan ()
const [inline]`

Definition at line 198 of file [float_utils_google.hpp](#).

References [FloatingPoint< RawType >::exponent_bits\(\)](#), [FloatingPoint< RawType >::fraction_bits\(\)](#), and [FloatingPoint< RawType >::kExponentBitMask](#).

Referenced by [FloatingPoint< RawType >::AlmostEquals\(\)](#).

34.75.4.8 `template<typename RawType> bool FloatingPoint< RawType >::AlmostEquals (
const FloatingPoint< RawType > & rhs) const [inline]`

Definition at line 210 of file [float_utils_google.hpp](#).

References [FloatingPoint< RawType >::is_nan\(\)](#), and [FloatingPoint< RawType >::kMaxUlp](#).

34.75.5 Member Data Documentation

34.75.5.1 `template<typename RawType> const size_t FloatingPoint< RawType
>::kBitCount = 8*sizeof(RawType) [static]`

Definition at line 126 of file [float_utils_google.hpp](#).

34.75.5.2 `template<typename RawType> const size_t FloatingPoint< RawType
>::kFractionBitCount [static]`

Initial value:

```
std::numeric_limits<RawType>::digits - 1
```

Definition at line 129 of file [float_utils_google.hpp](#).

34.75.5.3 `template<typename RawType> const size_t FloatingPoint< RawType
>::kExponentBitCount = kBitCount - 1 - kFractionBitCount [static]`

Definition at line 133 of file [float_utils_google.hpp](#).

34.75.5.4 `template<typename RawType> const Bits FloatingPoint< RawType
>::kSignBitMask = static_cast<Bits>(1) << (kBitCount - 1) [static]`

Definition at line 136 of file [float_utils_google.hpp](#).

Referenced by [FloatingPoint< RawType >::sign_bit\(\)](#).

34.75.5.5 `template<typename RawType> const Bits FloatingPoint< RawType
>::kFractionBitMask [static]`

Initial value:

```
~static_cast<Bits>(0) >> (kExponentBitCount + 1)
```

Definition at line 139 of file [float_utils_google.hpp](#).

Referenced by [FloatingPoint< RawType >::fraction_bits\(\)](#).

```
34.75.5.6 template<typename RawType> const Bits FloatingPoint< RawType
>::kExponentBitMask = ~(kSignBitMask | kFractionBitMask)
[static]
```

Definition at line 143 of file [float_utils_google.hpp](#).

Referenced by [FloatingPoint< RawType >::exponent_bits\(\)](#), [FloatingPoint< RawType >::Infinity\(\)](#), and [FloatingPoint< RawType >::is_nan\(\)](#).

```
34.75.5.7 template<typename RawType> const size_t FloatingPoint< RawType
>::kMaxUlp = 4 [static]
```

Definition at line 157 of file [float_utils_google.hpp](#).

Referenced by [FloatingPoint< RawType >::AlmostEquals\(\)](#).

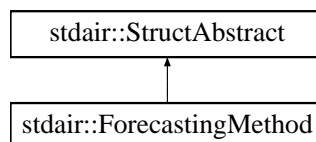
The documentation for this class was generated from the following file:

- [stdair/basic/float_utils_google.hpp](#)

34.76 stdair::ForecastingMethod Struct Reference

```
#include <stdair/basic/ForecastingMethod.hpp>
```

Inheritance diagram for stdair::ForecastingMethod:



Public Types

- enum [EN_ForecastingMethod](#) { [ADD_PK](#) = 0, [MUL_PK](#), [LAST_VALUE](#) }

Public Member Functions

- [EN_ForecastingMethod](#) [getMethod](#) () const
- std::string [getMethodAsString](#) () const
- const std::string [describe](#) () const
- bool [operator==](#) (const [EN_ForecastingMethod](#) &) const

- [ForecastingMethod](#) (const [EN_ForecastingMethod](#) &)
- [ForecastingMethod](#) (const char *iMethod*)
- [ForecastingMethod](#) (const [ForecastingMethod](#) &)
- void [toStream](#) (std::ostream &ioOut) const
- virtual void [fromStream](#) (std::istream &ioIn)

Static Public Member Functions

- static const std::string & [getLabel](#) (const [EN_ForecastingMethod](#) &)
- static char [getMethodLabel](#) (const [EN_ForecastingMethod](#) &)
- static std::string [getMethodLabelAsString](#) (const [EN_ForecastingMethod](#) &)
- static std::string [describeLabels](#) ()

34.76.1 Detailed Description

Enumeration of forecasting methods.

Definition at line 15 of file [ForecastingMethod.hpp](#).

34.76.2 Member Enumeration Documentation

34.76.2.1 enum stdair::ForecastingMethod::EN_ForecastingMethod

Enumerator:

ADD_PK

MUL_PK

LAST_VALUE

Definition at line 17 of file [ForecastingMethod.hpp](#).

34.76.3 Constructor & Destructor Documentation

34.76.3.1 stdair::ForecastingMethod::ForecastingMethod (const [EN_ForecastingMethod](#) & *iForecastingMethod*)

Constructor.

Definition at line 36 of file [ForecastingMethod.cpp](#).

34.76.3.2 stdair::ForecastingMethod::ForecastingMethod (const char *iMethod*)

Constructor.

Definition at line 41 of file [ForecastingMethod.cpp](#).

References [ADD_PK](#), [describeLabels\(\)](#), [LAST_VALUE](#), and [MUL_PK](#).

34.76.3.3 stdair::ForecastingMethod::ForecastingMethod (const ForecastingMethod & iForecastingMethod)

Default copy constructor.

Definition at line 30 of file [ForecastingMethod.cpp](#).

34.76.4 Member Function Documentation

34.76.4.1 const std::string & stdair::ForecastingMethod::getLabel (const EN_ForecastingMethod & iMethod) [static]

Get the label as a string (e.g., "BookingRequest" or "ScheduleChange").

Definition at line 59 of file [ForecastingMethod.cpp](#).

34.76.4.2 char stdair::ForecastingMethod::getMethodLabel (const EN_ForecastingMethod & iMethod) [static]

Get the label as a single char (e.g., 'B' or 'S').

Definition at line 64 of file [ForecastingMethod.cpp](#).

34.76.4.3 std::string stdair::ForecastingMethod::getMethodLabelAsString (const EN_ForecastingMethod & iMethod) [static]

Get the label as a string of a single char (e.g., "B" or "S").

Definition at line 70 of file [ForecastingMethod.cpp](#).

34.76.4.4 std::string stdair::ForecastingMethod::describeLabels () [static]

List the labels.

Definition at line 77 of file [ForecastingMethod.cpp](#).

References [LAST_VALUE](#).

Referenced by [ForecastingMethod\(\)](#).

34.76.4.5 ForecastingMethod::EN_ForecastingMethod stdair::ForecastingMethod::getMethod () const

Get the enumerated value.

Definition at line 89 of file [ForecastingMethod.cpp](#).

34.76.4.6 std::string stdair::ForecastingMethod::getMethodAsString () const

Get the enumerated value as a short string (e.g., "B" or "S").

Definition at line 94 of file [ForecastingMethod.cpp](#).

34.76.4.7 `const std::string stdair::ForecastingMethod::describe () const [virtual]`

Give a description of the structure (e.g., "BookingRequest" or "ScheduleChange").

Implements [stdair::StructAbstract](#).

Definition at line 101 of file [ForecastingMethod.cpp](#).

34.76.4.8 `bool stdair::ForecastingMethod::operator== (const EN_ForecastingMethod & iMethod) const`

Comparison operator.

Definition at line 109 of file [ForecastingMethod.cpp](#).

34.76.4.9 `void stdair::StructAbstract::toStream (std::ostream & ioOut) const [inline, inherited]`

Dump a Business Object into an output stream.

Parameters

<code>ostream&</code> the output stream.
--

Reimplemented in [stdair::YieldRange](#), [stdair::AirlineStruct](#), [stdair::BookingRequestStruct](#), [stdair::CancellationStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::TravelSolutionStruct](#), and [stdair::VirtualClassStruct](#).

Definition at line 29 of file [StructAbstract.hpp](#).

References [stdair::StructAbstract::describe\(\)](#).

Referenced by [operator<<\(\)](#).

34.76.4.10 `virtual void stdair::StructAbstract::fromStream (std::istream & ioIn) [inline, virtual, inherited]`

Read a Business Object from an input stream.

Parameters

<code>istream&</code> the input stream.

Reimplemented in [stdair::ProgressStatusSet](#), [stdair::YieldRange](#), [stdair::AirlineStruct](#), [stdair::BookingRequestStruct](#), [stdair::CancellationStruct](#), [stdair::EventStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::TravelSolutionStruct](#), and [stdair::VirtualClassStruct](#).

Definition at line 38 of file [StructAbstract.hpp](#).

Referenced by [operator>>\(\)](#).

The documentation for this struct was generated from the following files:

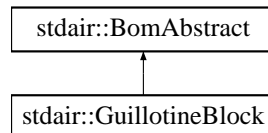
- [stdair/basic/ForecastingMethod.hpp](#)
- [stdair/basic/ForecastingMethod.cpp](#)

34.77 stdair::GuillotineBlock Class Reference

Class representing the actual attributes for an airline guillotine-block.

```
#include <stdair/bom/GuillotineBlock.hpp>
```

Inheritance diagram for stdair::GuillotineBlock:



Public Types

- typedef [GuillotineBlockKey](#) [Key_T](#)

Public Member Functions

- const [Key_T](#) & [getKey](#) () const
- [BomAbstract](#) *const [getParent](#) () const
- const [GuillotineNumber_T](#) & [getGuillotineNumber](#) () const
- const [HolderMap_T](#) & [getHolderMap](#) () const
- const [SegmentCabinIndexMap_T](#) & [getSegmentCabinIndexMap](#) () const
- const [ValueTypeIndexMap_T](#) & [getValueTypeIndexMap](#) () const
- const [BlockIndex_T](#) & [getBlockIndex](#) (const [MapKey_T](#) &) const
- const [BlockNumber_T](#) & [getBlockNumber](#) (const [SegmentCabin](#) &) const
- [ConstSegmentCabinDTDSnapshotView_T](#) [getConstSegmentCabinDTDBookingSnapshotView](#) (const [BlockNumber_T](#), const [BlockNumber_T](#), const [DTD_T](#)) const
- [ConstSegmentCabinDTDRangeSnapshotView_T](#) [getConstSegmentCabinDTDRangeBookingSnapshotView](#) (const [BlockNumber_T](#), const [BlockNumber_T](#), const [DTD_T](#), const [DTD_T](#)) const
- [SegmentCabinDTDSnapshotView_T](#) [getSegmentCabinDTDBookingSnapshotView](#) (const [BlockNumber_T](#), const [BlockNumber_T](#), const [DTD_T](#))
- [SegmentCabinDTDRangeSnapshotView_T](#) [getSegmentCabinDTDRangeBookingSnapshotView](#) (const [BlockNumber_T](#), const [BlockNumber_T](#), const [DTD_T](#), const [DTD_T](#))
- [ConstSegmentCabinDTDSnapshotView_T](#) [getConstSegmentCabinDTDCancellationSnapshotView](#) (const [BlockNumber_T](#), const [BlockNumber_T](#), const [DTD_T](#)) const
- [ConstSegmentCabinDTDRangeSnapshotView_T](#) [getConstSegmentCabinDTDRangeCancellationSnapshotView](#) (const [BlockNumber_T](#), const [BlockNumber_T](#), const [DTD_T](#), const [DTD_T](#)) const
- [SegmentCabinDTDSnapshotView_T](#) [getSegmentCabinDTDCancellationSnapshotView](#) (const [BlockNumber_T](#), const [BlockNumber_T](#), const [DTD_T](#))

- [SegmentCabinDTDRangeSnapshotView_T](#) [getSegmentCabinDTDRangeCancellationSnapshotView](#) (const [BlockNumber_T](#), const [BlockNumber_T](#), const [DTD_T](#), const [DTD_T](#))
- [ConstSegmentCabinDTDSnapshotView_T](#) [getConstSegmentCabinDTDProductAndPriceOrientedBookingSnapshotView](#) (const [BlockNumber_T](#), const [BlockNumber_T](#), const [DTD_T](#)) const
- [ConstSegmentCabinDTDRangeSnapshotView_T](#) [getConstSegmentCabinDTDRangeProductAndPriceOrientedBookingSnapshotView](#) (const [BlockNumber_T](#), const [BlockNumber_T](#), const [DTD_T](#), const [DTD_T](#)) const
- [SegmentCabinDTDSnapshotView_T](#) [getSegmentCabinDTDProductAndPriceOrientedBookingSnapshotView](#) (const [BlockNumber_T](#), const [BlockNumber_T](#), const [DTD_T](#))
- [SegmentCabinDTDRangeSnapshotView_T](#) [getSegmentCabinDTDRangeProductAndPriceOrientedBookingSnapshotView](#) (const [BlockNumber_T](#), const [BlockNumber_T](#), const [DTD_T](#), const [DTD_T](#))
- [ConstSegmentCabinDTDSnapshotView_T](#) [getConstSegmentCabinDTDAvailabilitySnapshotView](#) (const [BlockNumber_T](#), const [BlockNumber_T](#), const [DTD_T](#)) const
- [ConstSegmentCabinDTDRangeSnapshotView_T](#) [getConstSegmentCabinDTDRangeAvailabilitySnapshotView](#) (const [BlockNumber_T](#), const [BlockNumber_T](#), const [DTD_T](#), const [DTD_T](#)) const
- [SegmentCabinDTDSnapshotView_T](#) [getSegmentCabinDTDAvailabilitySnapshotView](#) (const [BlockNumber_T](#), const [BlockNumber_T](#), const [DTD_T](#))
- [SegmentCabinDTDRangeSnapshotView_T](#) [getSegmentCabinDTDRangeAvailabilitySnapshotView](#) (const [BlockNumber_T](#), const [BlockNumber_T](#), const [DTD_T](#), const [DTD_T](#))
- void [initSnapshotBlocks](#) (const [SegmentCabinIndexMap_T](#) &, const [ValueTypeIndexMap_T](#) &)
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- std::string [toString](#) () const
- const std::string [describeKey](#) () const
- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int iFileVersion)

Protected Member Functions

- [GuillotineBlock](#) (const [Key_T](#) &)
- virtual [~GuillotineBlock](#) ()

Protected Attributes

- [Key_T_key](#)
- [BomAbstract](#) * [_parent](#)
- [HolderMap_T_holderMap](#)
- [SegmentCabinIndexMap_T_segmentCabinIndexMap](#)
- [ValueTypeIndexMap_T_valueTypesIndexMap](#)

- [SnapshotBlock_T_bookingSnapshotBlock](#)
- [SnapshotBlock_T_cancellationSnapshotBlock](#)
- [SnapshotBlock_T_productAndPriceOrientedBookingSnapshotBlock](#)
- [SnapshotBlock_T_availabilitySnapshotBlock](#)

Friends

- class [FacBom](#)
- class [FacBomManager](#)
- class [boost::serialization::access](#)

34.77.1 Detailed Description

Class representing the actual attributes for an airline guillotine-block.

Definition at line 31 of file [GuillotineBlock.hpp](#).

34.77.2 Member Typedef Documentation

34.77.2.1 typedef [GuillotineBlockKey](#) [stdair::GuillotineBlock::Key_T](#)

Definition allowing to retrieve the associated BOM key type.

Definition at line 41 of file [GuillotineBlock.hpp](#).

34.77.3 Constructor & Destructor Documentation

34.77.3.1 [stdair::GuillotineBlock::GuillotineBlock](#) ([const Key_T & iKey](#)) [\[protected\]](#)

Main constructor.

Definition at line 34 of file [GuillotineBlock.cpp](#).

34.77.3.2 [stdair::GuillotineBlock::~~GuillotineBlock](#) () [\[protected, virtual\]](#)

Destructor.

Definition at line 38 of file [GuillotineBlock.cpp](#).

34.77.4 Member Function Documentation

34.77.4.1 [const Key_T& stdair::GuillotineBlock::getKey](#) () [const](#) [\[inline\]](#)

Get the guillotine-block key.

Definition at line 47 of file [GuillotineBlock.hpp](#).

References [_key](#).

34.77.4.2 BomAbstract* `const stdair::GuillotineBlock::getParent () const [inline]`

Get the parent object.

Definition at line 52 of file [GuillotineBlock.hpp](#).

References [_parent](#).

34.77.4.3 const GuillotineNumber_T `stdair::GuillotineBlock::getGuillotineNumber () const [inline]`

Get the guillotine number (part of the primary key).

Definition at line 57 of file [GuillotineBlock.hpp](#).

References [_key](#), and [stdair::GuillotineBlockKey::getGuillotineNumber\(\)](#).

34.77.4.4 const HolderMap_T `stdair::GuillotineBlock::getHolderMap () const [inline]`

Get the map of children holders.

Definition at line 64 of file [GuillotineBlock.hpp](#).

References [_holderMap](#).

34.77.4.5 const SegmentCabinIndexMap_T `stdair::GuillotineBlock::getSegmentCabinIndexMap () const [inline]`

Get the segment-cabin index map.

Definition at line 69 of file [GuillotineBlock.hpp](#).

References [_segmentCabinIndexMap](#).

34.77.4.6 const ValueTypeIndexMap_T `stdair::GuillotineBlock::getValueTypeIndexMap () const [inline]`

Get the value type index map.

Definition at line 74 of file [GuillotineBlock.hpp](#).

References [_valueTypesIndexMap](#).

34.77.4.7 const BlockIndex_T `stdair::GuillotineBlock::getBlockIndex (const MapKey_T & iKey) const`

Get the block index corresponding to the given value type.

Definition at line 79 of file [GuillotineBlock.cpp](#).

References [_valueTypesIndexMap](#).

34.77.4.8 const BlockNumber_T `stdair::GuillotineBlock::getBlockNumber (const SegmentCabin & iSegmentCabin) const`

Get the block number corresponding to the givent segment-cabin.

Definition at line 88 of file [GuillotineBlock.cpp](#).

References [_segmentCabinIndexMap](#).

34.77.4.9 ConstSegmentCabinDTDSnapshotView_T
 stdair::GuillotineBlock::getConstSegmentCabinDTDBookingSnapshotView (const
 BlockNumber_T iSCldxBegin, const BlockNumber_T iSCldxEnd, const
 DTD_T iDTD) const

Get the const view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 97 of file [GuillotineBlock.cpp](#).

References [_bookingSnapshotBlock](#), and [_valueTypesIndexMap](#).

34.77.4.10 ConstSegmentCabinDTDRangeSnapshotView_T
 stdair::GuillotineBlock::getConstSegmentCabinDTDRangeBookingSnapshotView (const
 BlockNumber_T iSCldxBegin, const BlockNumber_T iSCldxEnd, const
 DTD_T iDTDBegin, const DTD_T iDTDEnd) const

Get the const view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 110 of file [GuillotineBlock.cpp](#).

34.77.4.11 SegmentCabinDTDSnapshotView_T
 stdair::GuillotineBlock::getSegmentCabinDTDBookingSnapshotView (const
 BlockNumber_T iSCldxBegin, const BlockNumber_T iSCldxEnd, const
 DTD_T iDTD)

Get the view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 121 of file [GuillotineBlock.cpp](#).

References [_bookingSnapshotBlock](#), and [_valueTypesIndexMap](#).

34.77.4.12 SegmentCabinDTDRangeSnapshotView_T
 stdair::GuillotineBlock::getSegmentCabinDTDRangeBookingSnapshotView (const
 BlockNumber_T iSCldxBegin, const BlockNumber_T iSCldxEnd, const
 DTD_T iDTDBegin, const DTD_T iDTDEnd)

Get the view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 133 of file [GuillotineBlock.cpp](#).

References [_bookingSnapshotBlock](#), and [_valueTypesIndexMap](#).

34.77.4.13 ConstSegmentCabinDTDSnapshotView_T
 stdair::GuillotineBlock::getConstSegmentCabinDTDCancellationSnapshotView (const
 BlockNumber_T iSCldxBegin, const BlockNumber_T iSCldxEnd, const
 DTD_T iDTD) const

Get the const view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 146 of file [GuillotineBlock.cpp](#).

References [_cancellationSnapshotBlock](#), and [_valueTypesIndexMap](#).

34.77.4.14 ConstSegmentCabinDTDRangeSnapshotView_T
 stdair::GuillotineBlock::getConstSegmentCabinDTDRangeCancellationSnapshotView
 (const BlockNumber_T iSCldxBegin, const BlockNumber_T iSCldxEnd,
 const DTD_T iDTDBegin, const DTD_T iDTDEnd) const

Get the const view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 159 of file [GuillotineBlock.cpp](#).

34.77.4.15 SegmentCabinDTDSnapshotView_T
 stdair::GuillotineBlock::getSegmentCabinDTDCancellationSnapshotView (const
 BlockNumber_T iSCldxBegin, const BlockNumber_T iSCldxEnd, const
 DTD_T iDTD)

Get the view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 170 of file [GuillotineBlock.cpp](#).

References [_cancellationSnapshotBlock](#), and [_valueTypesIndexMap](#).

34.77.4.16 SegmentCabinDTDRangeSnapshotView_T
 stdair::GuillotineBlock::getSegmentCabinDTDRangeCancellationSnapshotView (const
 BlockNumber_T iSCldxBegin, const BlockNumber_T iSCldxEnd, const
 DTD_T iDTDBegin, const DTD_T iDTDEnd)

Get the view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 182 of file [GuillotineBlock.cpp](#).

References [_cancellationSnapshotBlock](#), and [_valueTypesIndexMap](#).

34.77.4.17 ConstSegmentCabinDTDSnapshotView_T
 stdair::GuillotineBlock::getConstSegmentCabinDTDRangeProductAndPriceOrientedBookingSnapshotView
 (const BlockNumber_T iSCldxBegin, const BlockNumber_T iSCldxEnd,
 const DTD_T iDTD) const

Get the const view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 195 of file [GuillotineBlock.cpp](#).

References [_productAndPriceOrientedBookingSnapshotBlock](#), and [_valueTypesIndexMap](#).

34.77.4.18 ConstSegmentCabinDTDRangeSnapshotView_T
 stdair::GuillotineBlock::getConstSegmentCabinDTDRangeProductAndPriceOrientedBookingSnapshotView
 (const BlockNumber_T iSCldxBegin, const BlockNumber_T iSCldxEnd,
 const DTD_T iDTDBegin, const DTD_T iDTDEnd) const

Get the const view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 208 of file [GuillotineBlock.cpp](#).

34.77.4.19 SegmentCabinDTDSnapshotView_T
stdair::GuillotineBlock::getSegmentCabinDTDProductAndPriceOrientedBookingSnapshotView
(const BlockNumber_T iSCldxBegin, const BlockNumber_T iSCldxEnd,
const DTD_T iDTD)

Get the view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 219 of file [GuillotineBlock.cpp](#).

References [_productAndPriceOrientedBookingSnapshotBlock](#), and [_valueTypesIndexMap](#).

34.77.4.20 SegmentCabinDTDRangeSnapshotView_T
stdair::GuillotineBlock::getSegmentCabinDTDRangeProductAndPriceOrientedBookingSnapshotView
(const BlockNumber_T iSCldxBegin, const BlockNumber_T iSCldxEnd,
const DTD_T iDTDBegin, const DTD_T iDTDEnd)

Get the view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 231 of file [GuillotineBlock.cpp](#).

References [_productAndPriceOrientedBookingSnapshotBlock](#), and [_valueTypesIndexMap](#).

34.77.4.21 ConstSegmentCabinDTDSnapshotView_T
stdair::GuillotineBlock::getConstSegmentCabinDTDAvailabilitySnapshotView (const
BlockNumber_T iSCldxBegin, const BlockNumber_T iSCldxEnd, const
DTD_T iDTD) const

Get the const view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 244 of file [GuillotineBlock.cpp](#).

References [_availabilitySnapshotBlock](#), and [_valueTypesIndexMap](#).

34.77.4.22 ConstSegmentCabinDTDRangeSnapshotView_T
stdair::GuillotineBlock::getConstSegmentCabinDTDRangeAvailabilitySnapshotView (const
BlockNumber_T iSCldxBegin, const BlockNumber_T iSCldxEnd, const
DTD_T iDTDBegin, const DTD_T iDTDEnd) const

Get the const view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 257 of file [GuillotineBlock.cpp](#).

34.77.4.23 SegmentCabinDTDSnapshotView_T
stdair::GuillotineBlock::getSegmentCabinDTDAvailabilitySnapshotView (const
BlockNumber_T iSCldxBegin, const BlockNumber_T iSCldxEnd, const
DTD_T iDTD)

Get the view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 268 of file [GuillotineBlock.cpp](#).

References [_availabilitySnapshotBlock](#), and [_valueTypesIndexMap](#).

34.77.4.24 **SegmentCabinDTDRangeSnapshotView_T**
 stdair::GuillotineBlock::getSegmentCabinDTDRangeAvailabilitySnapshotView (
 const BlockNumber_T *iSCIdxBegin*, const BlockNumber_T *iSCIdxEnd*, const
 DTD_T *iDTDBegin*, const DTD_T *iDTDEnd*)

Get the view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 280 of file [GuillotineBlock.cpp](#).

References [_availabilitySnapshotBlock](#), and [_valueTypesIndexMap](#).

34.77.4.25 void stdair::GuillotineBlock::initSnapshotBlocks (const
 SegmentCabinIndexMap_T & *iSegmentCabinIndexMap*, const
 ValueTypeIndexMap_T & *iValueTypeIndexMap*)

Set the segment-cabin and value type index maps and initialise the snapshot blocks.

Definition at line 50 of file [GuillotineBlock.cpp](#).

References [_availabilitySnapshotBlock](#), [_bookingSnapshotBlock](#), [_cancellationSnapshotBlock](#), [_productAndPriceOrientedBookingSnapshotBlock](#), [_segmentCabinIndexMap](#), [_valueTypesIndexMap](#), and [stdair::DEFAULT_MAX_DTD](#).

34.77.4.26 void stdair::GuillotineBlock::toStream (std::ostream & *ioOut*) const [inline,
 virtual]

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Implements [stdair::BomAbstract](#).

Definition at line 209 of file [GuillotineBlock.hpp](#).

References [toString\(\)](#).

34.77.4.27 void stdair::GuillotineBlock::fromStream (std::istream & *ioIn*) [inline,
 virtual]

Read a Business Object from an input stream.

Parameters

<i>istream&</i> the input stream.

Implements [stdair::BomAbstract](#).

Definition at line 218 of file [GuillotineBlock.hpp](#).

34.77.4.28 std::string stdair::GuillotineBlock::toString () const [virtual]

Get the serialised version of the Business Object.

Implements [stdair::BomAbstract](#).

Definition at line 42 of file [GuillotineBlock.cpp](#).

References [describeKey\(\)](#).

Referenced by [toStream\(\)](#).

34.77.4.29 `const std::string stdair::GuillotineBlock::describeKey () const` `[inline]`

Get a string describing the key.

Definition at line 229 of file [GuillotineBlock.hpp](#).

References [_key](#), and [stdair::GuillotineBlockKey::toString\(\)](#).

Referenced by [toString\(\)](#).

34.77.4.30 `template<class Archive > void stdair::GuillotineBlock::serialize (Archive & ar,
const unsigned int iFileVersion)`

Serialisation.

Definition at line 307 of file [GuillotineBlock.cpp](#).

References [_key](#).

34.77.5 Friends And Related Function Documentation

34.77.5.1 `friend class FacBom` `[friend]`

Definition at line 32 of file [GuillotineBlock.hpp](#).

34.77.5.2 `friend class FacBomManager` `[friend]`

Definition at line 33 of file [GuillotineBlock.hpp](#).

34.77.5.3 `friend class boost::serialization::access` `[friend]`

Definition at line 34 of file [GuillotineBlock.hpp](#).

34.77.6 Member Data Documentation

34.77.6.1 `Key_T stdair::GuillotineBlock::_key` `[protected]`

Primary key (guillotine number and departure block).

Definition at line 278 of file [GuillotineBlock.hpp](#).

Referenced by [describeKey\(\)](#), [getGuillotineNumber\(\)](#), [getKey\(\)](#), and [serialize\(\)](#).

34.77.6.2 `BomAbstract* stdair::GuillotineBlock::_parent` `[protected]`

Pointer on the parent class ([Inventory](#)).

Definition at line 281 of file [GuillotineBlock.hpp](#).

Referenced by [getParent\(\)](#).

34.77.6.3 HolderMap_T stdair::GuillotineBlock::_holderMap [protected]

Map holding the children.

Definition at line 284 of file [GuillotineBlock.hpp](#).

Referenced by [getHolderMap\(\)](#).

34.77.6.4 SegmentCabinIndexMap_T stdair::GuillotineBlock::_segmentCabinIndexMap [protected]

Map holding the segment-cabin position within the snapshot blocks.

Definition at line 287 of file [GuillotineBlock.hpp](#).

Referenced by [getBlockNumber\(\)](#), [getSegmentCabinIndexMap\(\)](#), and [initSnapshotBlocks\(\)](#).

34.77.6.5 ValueTypeIndexMap_T stdair::GuillotineBlock::_valueTypesIndexMap [protected]

Map holding the value type (class, Q-equivalent, etc) within a a segment-cabin inside the snapshot blocks.

Definition at line 291 of file [GuillotineBlock.hpp](#).

Referenced by [getBlockIndex\(\)](#), [getConstSegmentCabinDTDAvailabilitySnapshotView\(\)](#), [getConstSegmentCabinDTDBookingSnapshotView\(\)](#), [getConstSegmentCabinDTDCancellationSnapshotView\(\)](#), [getConstSegmentCabinDTDProductAndPriceOrientedBookingSnapshotView\(\)](#), [getSegmentCabinDTDAvailabilitySnapshotView\(\)](#), [getSegmentCabinDTDBookingSnapshotView\(\)](#), [getSegmentCabinDTDCancellationSnapshotView\(\)](#), [getSegmentCabinDTDProductAndPriceOrientedBookingSnapshotView\(\)](#), [getSegmentCabinDTDRangeAvailabilitySnapshotView\(\)](#), [getSegmentCabinDTDRangeBookingSnapshotView\(\)](#), [getSegmentCabinDTDRangeCancellationSnapshotView\(\)](#), [getSegmentCabinDTDRangeProductAndPriceOrientedBookingSnapshotView\(\)](#), [getValueTypeIndexMap\(\)](#), and [initSnapshotBlocks\(\)](#).

34.77.6.6 SnapshotBlock_T stdair::GuillotineBlock::_bookingSnapshotBlock [protected]

Booking snapshot block.

Definition at line 294 of file [GuillotineBlock.hpp](#).

Referenced by [getConstSegmentCabinDTDBookingSnapshotView\(\)](#), [getSegmentCabinDTDBookingSnapshotView\(\)](#), [getSegmentCabinDTDRangeBookingSnapshotView\(\)](#), and [initSnapshotBlocks\(\)](#).

34.77.6.7 SnapshotBlock_T stdair::GuillotineBlock::_cancellationSnapshotBlock [protected]

Cancellation snapshot block.

Definition at line 297 of file [GuillotineBlock.hpp](#).

Referenced by [getConstSegmentCabinDTDCancellationSnapshotView\(\)](#), [getSegmentCabinDTDCancellationSnapshotView\(\)](#), [getSegmentCabinDTDRangeCancellationSnapshotView\(\)](#), and [initSnapshotBlocks\(\)](#).

34.77.6.8 SnapshotBlock_T stdair::GuillotineBlock::_productAndPriceOrientedBookingSnapshotBlock [protected]

Price & product oriented booking block.

Definition at line 300 of file [GuillotineBlock.hpp](#).

Referenced by [getConstSegmentCabinDTDProductAndPriceOrientedBookingSnapshotView\(\)](#), [getSegmentCabinDTDProductAndPriceOrientedBookingSnapshotView\(\)](#), [getSegmentCabinDTDRangeProductAndPriceOrientedBookingSnapshotView\(\)](#), and [initSnapshotBlocks\(\)](#).

34.77.6.9 SnapshotBlock_T stdair::GuillotineBlock::_availabilitySnapshotBlock [protected]

Availability block.

Definition at line 303 of file [GuillotineBlock.hpp](#).

Referenced by [getConstSegmentCabinDTDAvailabilitySnapshotView\(\)](#), [getSegmentCabinDTDAvailabilitySnapshotView\(\)](#), [getSegmentCabinDTDRangeAvailabilitySnapshotView\(\)](#), and [initSnapshotBlocks\(\)](#).

The documentation for this class was generated from the following files:

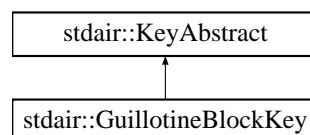
- [stdair/bom/GuillotineBlock.hpp](#)
- [stdair/bom/GuillotineBlock.cpp](#)

34.78 stdair::GuillotineBlockKey Struct Reference

Key of a given guillotine block, made of a guillotine number.

```
#include <stdair/bom/GuillotineBlockKey.hpp>
```

Inheritance diagram for stdair::GuillotineBlockKey:



Public Member Functions

- [GuillotineBlockKey](#) (const [GuillotineNumber_T](#) &)
- [GuillotineBlockKey](#) (const [GuillotineBlockKey](#) &)

- `~GuillotineBlockKey ()`
- `const GuillotineNumber_T & getGuillotineNumber () const`
- `void toStream (std::ostream &ioOut) const`
- `void fromStream (std::istream &ioIn)`
- `const std::string toString () const`
- `template<class Archive >`
`void serialize (Archive &ar, const unsigned int iFileVersion)`

Friends

- class `boost::serialization::access`

34.78.1 Detailed Description

Key of a given guillotine block, made of a guillotine number.

Definition at line 26 of file [GuillotineBlockKey.hpp](#).

34.78.2 Constructor & Destructor Documentation

34.78.2.1 `stdair::GuillotineBlockKey::GuillotineBlockKey (const GuillotineNumber_T & iGuillotineNumber)`

Constructor.

Definition at line 26 of file [GuillotineBlockKey.cpp](#).

34.78.2.2 `stdair::GuillotineBlockKey::GuillotineBlockKey (const GuillotineBlockKey & iKey)`

Copy constructor.

Definition at line 31 of file [GuillotineBlockKey.cpp](#).

34.78.2.3 `stdair::GuillotineBlockKey::~~GuillotineBlockKey ()`

Destructor.

Definition at line 36 of file [GuillotineBlockKey.cpp](#).

34.78.3 Member Function Documentation

34.78.3.1 `const GuillotineNumber_T & stdair::GuillotineBlockKey::getGuillotineNumber ()`
`const [inline]`

Get the guillotine number.

Definition at line 56 of file [GuillotineBlockKey.hpp](#).

Referenced by [stdair::GuillotineBlock::getGuillotineNumber\(\)](#).

34.78.3.2 `void stdair::GuillotineBlockKey::toStream (std::ostream & ioOut) const`
`[virtual]`

Dump a Business Object Key into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 40 of file [GuillotineBlockKey.cpp](#).

References [toString\(\)](#).

34.78.3.3 `void stdair::GuillotineBlockKey::fromStream (std::istream & ioIn)` `[virtual]`

Read a Business Object Key from an input stream.

Parameters

<i>istream&</i> the input stream.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 45 of file [GuillotineBlockKey.cpp](#).

34.78.3.4 `const std::string stdair::GuillotineBlockKey::toString () const` `[virtual]`

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-block.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 49 of file [GuillotineBlockKey.cpp](#).

Referenced by [stdair::GuillotineBlock::describeKey\(\)](#), and [toStream\(\)](#).

34.78.3.5 `template<class Archive > void stdair::GuillotineBlockKey::serialize (Archive & ar, const unsigned int iFileVersion)`

Serialisation.

Definition at line 71 of file [GuillotineBlockKey.cpp](#).

34.78.4 Friends And Related Function Documentation

34.78.4.1 `friend class boost::serialization::access` `[friend]`

Definition at line 27 of file [GuillotineBlockKey.hpp](#).

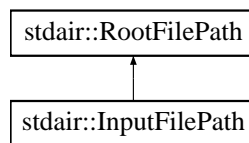
The documentation for this struct was generated from the following files:

- [stdair/bom/GuillotineBlockKey.hpp](#)
- [stdair/bom/GuillotineBlockKey.cpp](#)

34.79 stdair::InputFilePath Class Reference

```
#include <stdair/stdair_file.hpp>
```

Inheritance diagram for stdair::InputFilePath:



Public Member Functions

- [InputFilePath](#) (const [Filename_T](#) &iFilename)
- const char * [name](#) () const

Protected Attributes

- const [Filename_T](#) _filename

34.79.1 Detailed Description

Input File.

Definition at line [54](#) of file [stdair_file.hpp](#).

34.79.2 Constructor & Destructor Documentation

34.79.2.1 stdair::InputFilePath::InputFilePath (const [Filename_T](#) &iFilename)
[inline]

Constructor.

Definition at line [57](#) of file [stdair_file.hpp](#).

34.79.3 Member Function Documentation

34.79.3.1 const char* stdair::RootFilePath::name () const [inline, inherited]

Give the details of the exception.

Definition at line 42 of file [stdair_file.hpp](#).

References [stdair::RootFilePath::_filename](#).

34.79.4 Member Data Documentation

34.79.4.1 `const Filename_T stdair::RootFilePath::_filename` [protected, inherited]

Name of the file.

Definition at line 50 of file [stdair_file.hpp](#).

Referenced by [stdair::RootFilePath::name\(\)](#).

The documentation for this class was generated from the following file:

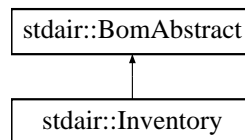
- [stdair/stdair_file.hpp](#)

34.80 stdair::Inventory Class Reference

Class representing the actual attributes for an airline inventory.

```
#include <stdair/bom/Inventory.hpp>
```

Inheritance diagram for `stdair::Inventory`:



Public Types

- typedef [InventoryKey](#) `Key_T`

Public Member Functions

- const [Key_T](#) & [getKey](#) () const
- const [AirlineCode_T](#) & [getAirlineCode](#) () const
- [BomAbstract](#) *const [getParent](#) () const
- const [HolderMap_T](#) & [getHolderMap](#) () const
- [FlightDate](#) * [getFlightDate](#) (const std::string &iFlightDateKeyStr) const
- [FlightDate](#) * [getFlightDate](#) (const [FlightDateKey](#) &) const
- void [setAirlineFeature](#) (const [AirlineFeature](#) *ioAirlineFeaturePtr)
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)

- std::string [toString](#) () const
- const std::string [describeKey](#) () const
- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int iFileVersion)

Protected Member Functions

- [Inventory](#) (const [Key_T](#) &)
- [~Inventory](#) ()

Protected Attributes

- [Key_T _key](#)
- [BomAbstract](#) * [_parent](#)
- const [AirlineFeature](#) * [_airlineFeature](#)
- [HolderMap_T _holderMap](#)

Friends

- class [FacBom](#)
- class [FacBomManager](#)
- class [boost::serialization::access](#)

34.80.1 Detailed Description

Class representing the actual attributes for an airline inventory.

Definition at line 33 of file [Inventory.hpp](#).

34.80.2 Member Typedef Documentation

34.80.2.1 typedef InventoryKey stdair::Inventory::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 43 of file [Inventory.hpp](#).

34.80.3 Constructor & Destructor Documentation

34.80.3.1 stdair::Inventory::Inventory (const Key_T & iKey) [protected]

Constructor.

Definition at line 27 of file [Inventory.cpp](#).

34.80.3.2 stdair::Inventory::~~Inventory () [protected]

Destructor.

Definition at line 31 of file [Inventory.cpp](#).

34.80.4 Member Function Documentation

34.80.4.1 const Key_T& stdair::Inventory::getKey () const [inline]

Get the inventory key (airline code).

Definition at line 49 of file [Inventory.hpp](#).

References [_key](#).

34.80.4.2 const AirlineCode_T& stdair::Inventory::getAirlineCode () const [inline]

Get the airline code (inventory/primary key).

Definition at line 54 of file [Inventory.hpp](#).

References [_key](#), and [stdair::InventoryKey::getAirlineCode\(\)](#).

Referenced by [stdair::OnDDate::getAirlineCode\(\)](#), [stdair::FlightDate::getAirlineCode\(\)](#), and [stdair::BomRetriever::retrieveSegmentDateFromLongKey\(\)](#).

34.80.4.3 BomAbstract* const stdair::Inventory::getParent () const [inline]

Get the parent object.

Definition at line 59 of file [Inventory.hpp](#).

References [_parent](#).

34.80.4.4 const HolderMap_T& stdair::Inventory::getHolderMap () const [inline]

Get the map of children.

Definition at line 64 of file [Inventory.hpp](#).

References [_holderMap](#).

34.80.4.5 FlightDate * stdair::Inventory::getFlightDate (const std::string & iFlightDateKeyStr) const

Get a pointer on the [FlightDate](#) object corresponding to the given key.

Note

The [FlightDate](#) object can be inherited from, if needed. In that case, a `dynamic_cast<>` may be needed.

Parameters

<i>const</i> std::string& The flight-date key.
--

Returns

FlightDate* Found [FlightDate](#) object. NULL if not found.

Definition at line [43](#) of file [Inventory.cpp](#).

Referenced by [getFlightDate\(\)](#), [stdair::BomRetriever::retrieveFlightDateFromKey\(\)](#), and [stdair::BomRetriever::retrieveFlightDateFromLongKey\(\)](#).

34.80.4.6 `FlightDate * stdair::Inventory::getFlightDate (const FlightDateKey & iFlightDateKey) const`

Get a pointer on the [FlightDate](#) object corresponding to the given key.

Note

The [FlightDate](#) object can be inherited from, if needed. In that case, a `dynamic_cast<>` may be needed.

Parameters

<code>const</code> FlightDateKey & The flight-date key
--

Returns

FlightDate* Found [FlightDate](#) object. NULL if not found.

Definition at line [51](#) of file [Inventory.cpp](#).

References [getFlightDate\(\)](#), and [stdair::FlightDateKey::toString\(\)](#).

34.80.4.7 `void stdair::Inventory::setAirlineFeature (const AirlineFeature * ioAirlineFeaturePtr) [inline]`

Set the airline feature.

Definition at line [96](#) of file [Inventory.hpp](#).

References [_airlineFeature](#).

34.80.4.8 `void stdair::Inventory::toStream (std::ostream & ioOut) const [inline, virtual]`

Dump a Business Object into an output stream.

Parameters

<code>ostream&</code> the output stream.
--

Implements [stdair::BomAbstract](#).

Definition at line [108](#) of file [Inventory.hpp](#).

References [toString\(\)](#).

34.80.4.9 `void stdair::Inventory::fromStream (std::istream & ioln)` `[inline, virtual]`

Read a Business Object from an input stream.

Parameters

<i>istream</i> & the input stream.

Implements [stdair::BomAbstract](#).

Definition at line 117 of file [Inventory.hpp](#).

34.80.4.10 `std::string stdair::Inventory::toString () const` `[virtual]`

Get the serialised version of the Business Object.

Implements [stdair::BomAbstract](#).

Definition at line 35 of file [Inventory.cpp](#).

References [describeKey\(\)](#).

Referenced by [toStream\(\)](#).

34.80.4.11 `const std::string stdair::Inventory::describeKey () const` `[inline]`

Get a string describing the key.

Definition at line 128 of file [Inventory.hpp](#).

References [_key](#), and [stdair::InventoryKey::toString\(\)](#).

Referenced by [toString\(\)](#).

34.80.4.12 `template<class Archive > void stdair::Inventory::serialize (Archive & ar, const unsigned int iFileVersion)`

Serialisation.

Definition at line 160 of file [CmdBomSerialiser.cpp](#).

References [_key](#).

34.80.5 Friends And Related Function Documentation

34.80.5.1 `friend class FacBom` `[friend]`

Definition at line 34 of file [Inventory.hpp](#).

34.80.5.2 `friend class FacBomManager` `[friend]`

Definition at line 35 of file [Inventory.hpp](#).

34.80.5.3 friend class boost::serialization::access [friend]

Definition at line 36 of file [Inventory.hpp](#).

34.80.6 Member Data Documentation

34.80.6.1 Key_T stdair::Inventory::_key [protected]

Primary key (airline code).

Definition at line 180 of file [Inventory.hpp](#).

Referenced by [describeKey\(\)](#), [getAirlineCode\(\)](#), [getKey\(\)](#), and [serialize\(\)](#).

34.80.6.2 BomAbstract* stdair::Inventory::_parent [protected]

Pointer on the parent class ([BomRoot](#)).

Definition at line 185 of file [Inventory.hpp](#).

Referenced by [getParent\(\)](#).

34.80.6.3 const AirlineFeature* stdair::Inventory::_airlineFeature [protected]

Features specific to the airline.

Definition at line 190 of file [Inventory.hpp](#).

Referenced by [setAirlineFeature\(\)](#).

34.80.6.4 HolderMap_T stdair::Inventory::_holderMap [protected]

Map holding the children ([FlightDate](#) objects).

Definition at line 195 of file [Inventory.hpp](#).

Referenced by [getHolderMap\(\)](#).

The documentation for this class was generated from the following files:

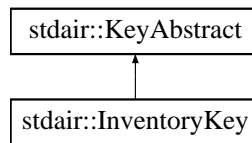
- [stdair/bom/Inventory.hpp](#)
- [stdair/bom/Inventory.cpp](#)
- [stdair/command/CmdBomSerialiser.cpp](#)

34.81 stdair::InventoryKey Struct Reference

Key of a given inventory, made of the airline code.

```
#include <stdair/bom/InventoryKey.hpp>
```

Inheritance diagram for stdair::InventoryKey:



Public Member Functions

- [InventoryKey](#) (const [AirlineCode_T](#) &iAirlineCode)
- [InventoryKey](#) (const [InventoryKey](#) &)
- [~InventoryKey](#) ()
- const [AirlineCode_T](#) & [getAirlineCode](#) () const
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- const std::string [toString](#) () const
- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int iFileVersion)

Friends

- class [boost::serialization::access](#)

34.81.1 Detailed Description

Key of a given inventory, made of the airline code.

Definition at line 26 of file [InventoryKey.hpp](#).

34.81.2 Constructor & Destructor Documentation

34.81.2.1 stdair::InventoryKey::InventoryKey (const [AirlineCode_T](#) & *iAirlineCode*)

Constructor.

Definition at line 23 of file [InventoryKey.cpp](#).

34.81.2.2 stdair::InventoryKey::InventoryKey (const [InventoryKey](#) & *iKey*)

Copy constructor.

Definition at line 28 of file [InventoryKey.cpp](#).

34.81.2.3 stdair::InventoryKey::~~InventoryKey ()

Destructor.

Definition at line 33 of file [InventoryKey.cpp](#).

34.81.3 Member Function Documentation

34.81.3.1 `const AirlineCode_T& stdair::InventoryKey::getAirlineCode () const`
[inline]

Get the airline code.

Definition at line 58 of file [InventoryKey.hpp](#).

Referenced by [stdair::Inventory::getAirlineCode\(\)](#).

34.81.3.2 `void stdair::InventoryKey::toStream (std::ostream & ioOut) const` [virtual]

Dump a Business Object Key into an output stream.

Parameters

<code>ostream&</code> the output stream.
--

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 37 of file [InventoryKey.cpp](#).

References [toString\(\)](#).

34.81.3.3 `void stdair::InventoryKey::fromStream (std::istream & iIn)` [virtual]

Read a Business Object Key from an input stream.

Parameters

<code>istream&</code> the input stream.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 42 of file [InventoryKey.cpp](#).

34.81.3.4 `const std::string stdair::InventoryKey::toString () const` [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 46 of file [InventoryKey.cpp](#).

Referenced by [stdair::Inventory::describeKey\(\)](#), [stdair::BomRoot::getInventory\(\)](#), and [toStream\(\)](#).

34.81.3.5 `template<class Archive > void stdair::InventoryKey::serialize (Archive & ar, const unsigned int iFileVersion)`

Serialisation.

Definition at line 68 of file [InventoryKey.cpp](#).

34.81.4 Friends And Related Function Documentation

34.81.4.1 friend class boost::serialization::access [friend]

Definition at line 27 of file [InventoryKey.hpp](#).

The documentation for this struct was generated from the following files:

- [stdair/bom/InventoryKey.hpp](#)
- [stdair/bom/InventoryKey.cpp](#)

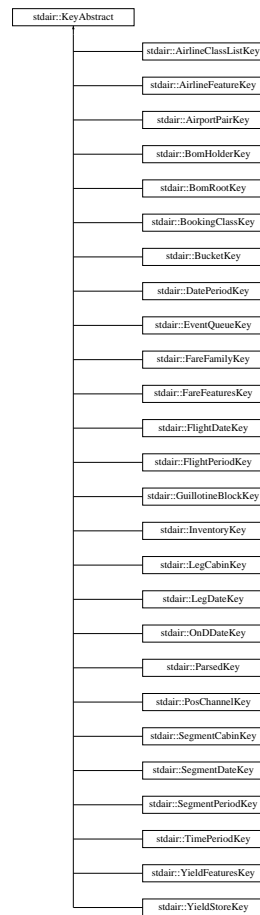
34.82 stdair::KeyAbstract Struct Reference

Base class for the keys of Business Object Model (BOM) layer.

Note that that key allows to differentiate two objects at the same level only. For instance, the segment-date key allows to differentiate two segment-dates under a given flight-date, but does not allow to differentiate two segment-dates in general.

```
#include <stdair/bom/KeyAbstract.hpp>
```

Inheritance diagram for stdair::KeyAbstract:



Public Member Functions

- virtual void [toStream](#) (std::ostream &ioOut) const
Dump a Business Object Key into an output stream.
- virtual void [fromStream](#) (std::istream &ioIn)
Read a Business Object Key from an input stream.
- virtual const std::string [toString](#) () const
*Get the serialised version of the Business Object Key.
That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.
For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.*
- virtual [~KeyAbstract](#) ()
Default destructor.

34.82.1 Detailed Description

Base class for the keys of Business Object Model (BOM) layer.

Note that that key allows to differentiate two objects at the same level only. For instance, the segment-date key allows to differentiate two segment-dates under a given flight-date, but does not allow to differentiate two segment-dates in general.

Definition at line 26 of file [KeyAbstract.hpp](#).

34.82.2 Constructor & Destructor Documentation

34.82.2.1 `virtual stdair::KeyAbstract::~KeyAbstract () [inline, virtual]`

Default destructor.

Definition at line 56 of file [KeyAbstract.hpp](#).

34.82.3 Member Function Documentation

34.82.3.1 `virtual void stdair::KeyAbstract::toStream (std::ostream & ioOut) const [inline, virtual]`

Dump a Business Object Key into an output stream.

Parameters

<i>ioOut</i> ostream& the output stream.
--

Reimplemented in [stdair::AirlineClassListKey](#), [stdair::AirlineFeatureKey](#), [stdair::AirportPairKey](#), [stdair::BomHolderKey](#), [stdair::BomRootKey](#), [stdair::BookingClassKey](#), [stdair::BucketKey](#), [stdair::DatePeriodKey](#), [stdair::EventQueueKey](#), [stdair::FareFamilyKey](#), [stdair::FareFeaturesKey](#), [stdair::FlightDateKey](#), [stdair::FlightPeriodKey](#), [stdair::GuillotineBlockKey](#), [stdair::InventoryKey](#), [stdair::LegCabinKey](#), [stdair::LegDateKey](#), [stdair::OnDDateKey](#), [stdair::ParsedKey](#), [stdair::PosChannelKey](#), [stdair::SegmentCabinKey](#), [stdair::SegmentDateKey](#), [stdair::SegmentPeriodKey](#), [stdair::TimePeriodKey](#), [stdair::YieldFeaturesKey](#), and [stdair::YieldStoreKey](#).

Definition at line 34 of file [KeyAbstract.hpp](#).

Referenced by [operator<<\(\)](#).

34.82.3.2 `virtual void stdair::KeyAbstract::fromStream (std::istream & ioIn) [inline, virtual]`

Read a Business Object Key from an input stream.

Parameters

<i>ioIn</i> istream& the input stream.
--

Reimplemented in [stdair::AirlineClassListKey](#), [stdair::AirlineFeatureKey](#), [stdair::AirportPairKey](#), [stdair::BomHolderKey](#), [stdair::BomRootKey](#), [stdair::BookingClassKey](#), [stdair::BucketKey](#),

[stdair::DatePeriodKey](#), [stdair::EventQueueKey](#), [stdair::FareFamilyKey](#), [stdair::FareFeaturesKey](#), [stdair::FlightDateKey](#), [stdair::FlightPeriodKey](#), [stdair::GuillotineBlockKey](#), [stdair::InventoryKey](#), [stdair::LegCabinKey](#), [stdair::LegDateKey](#), [stdair::OnDDateKey](#), [stdair::ParsedKey](#), [stdair::PosChannelKey](#), [stdair::SegmentCabinKey](#), [stdair::SegmentDateKey](#), [stdair::SegmentPeriodKey](#), [stdair::TimePeriodKey](#), [stdair::YieldFeaturesKey](#), and [stdair::YieldStoreKey](#).

Definition at line 40 of file [KeyAbstract.hpp](#).

Referenced by [operator>>\(\)](#).

```
34.82.3.3 virtual const std::string stdair::KeyAbstract::toString ( ) const [inline,
virtual]
```

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Parameters

out	const	std::string The serialised version of the Business Object Key.
-----	-------	--

Reimplemented in [stdair::AirlineClassListKey](#), [stdair::AirlineFeatureKey](#), [stdair::AirportPairKey](#), [stdair::BomHolderKey](#), [stdair::BomRootKey](#), [stdair::BookingClassKey](#), [stdair::BucketKey](#), [stdair::DatePeriodKey](#), [stdair::EventQueueKey](#), [stdair::FareFamilyKey](#), [stdair::FareFeaturesKey](#), [stdair::FlightDateKey](#), [stdair::FlightPeriodKey](#), [stdair::GuillotineBlockKey](#), [stdair::InventoryKey](#), [stdair::LegCabinKey](#), [stdair::LegDateKey](#), [stdair::OnDDateKey](#), [stdair::ParsedKey](#), [stdair::PosChannelKey](#), [stdair::SegmentCabinKey](#), [stdair::SegmentDateKey](#), [stdair::SegmentPeriodKey](#), [stdair::TimePeriodKey](#), [stdair::YieldFeaturesKey](#), and [stdair::YieldStoreKey](#).

Definition at line 51 of file [KeyAbstract.hpp](#).

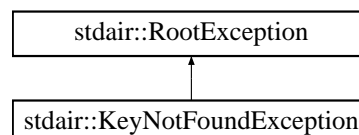
The documentation for this struct was generated from the following file:

- [stdair/bom/KeyAbstract.hpp](#)

34.83 stdair::KeyNotFoundException Class Reference

```
#include <stdair/stdair_exceptions.hpp>
```

Inheritance diagram for `stdair::KeyNotFoundException`:



Public Member Functions

- [KeyNotFoundException](#) (const std::string &iWhat)
- const char * [what](#) () const throw ()

Protected Attributes

- std::string [_what](#)

34.83.1 Detailed Description

Not found key.

Definition at line 126 of file [stdair_exceptions.hpp](#).

34.83.2 Constructor & Destructor Documentation

34.83.2.1 `stdair::KeyNotFoundException::KeyNotFoundException (const std::string & iWhat)`
[inline]

Constructor.

Definition at line 129 of file [stdair_exceptions.hpp](#).

34.83.3 Member Function Documentation

34.83.3.1 `const char* stdair::RootException::what () const throw ()` [inline,
inherited]

Give the details of the exception.

Definition at line 38 of file [stdair_exceptions.hpp](#).

References [stdair::RootException::_what](#).

34.83.4 Member Data Documentation

34.83.4.1 `std::string stdair::RootException::_what` [protected, inherited]

Details for the exception.

Definition at line 46 of file [stdair_exceptions.hpp](#).

Referenced by [stdair::RootException::what\(\)](#).

The documentation for this class was generated from the following file:

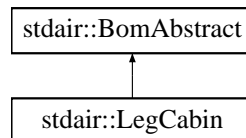
- [stdair/stdair_exceptions.hpp](#)

34.84 stdair::LegCabin Class Reference

Class representing the actual attributes for an airline leg-cabin.

```
#include <stdair/bom/LegCabin.hpp>
```

Inheritance diagram for stdair::LegCabin:



Public Types

- typedef [LegCabinKey](#) [Key_T](#)

Public Member Functions

- const [Key_T](#) & [getKey](#) () const
- [BomAbstract](#) *const [getParent](#) () const
- const [CabinCode_T](#) & [getCabinCode](#) () const
- const [MapKey_T](#) [getFullerKey](#) () const
- const [HolderMap_T](#) & [getHolderMap](#) () const
- const [CabinCapacity_T](#) & [getOfferedCapacity](#) () const
- const [CabinCapacity_T](#) & [getPhysicalCapacity](#) () const
- const [NbOfSeats_T](#) & [getSoldSeat](#) () const
- const [CommittedSpace_T](#) & [getCommittedSpace](#) () const
- const [Availability_T](#) & [getAvailabilityPool](#) () const
- const [Availability_T](#) & [getAvailability](#) () const
- const [BidPrice_T](#) & [getCurrentBidPrice](#) () const
- const [BidPrice_T](#) & [getPreviousBidPrice](#) () const
- const [BidPriceVector_T](#) & [getBidPriceVector](#) () const
- const [CapacityAdjustment_T](#) & [getRegradeAdjustment](#) () const
- const [AuthorizationLevel_T](#) & [getAuthorizationLevel](#) () const
- const [UPR_T](#) & [getUPR](#) () const
- const [Availability_T](#) & [getNetAvailability](#) () const
- const [Availability_T](#) & [getGrossAvailability](#) () const
- const [OverbookingRate_T](#) & [getAvgCancellationPercentage](#) () const
- const [NbOfSeats_T](#) & [getETB](#) () const
- const [NbOfSeats_T](#) & [getStaffNbOfSeats](#) () const
- const [NbOfSeats_T](#) & [getWLNbOfSeats](#) () const
- const [NbOfSeats_T](#) & [getGroupNbOfSeats](#) () const
- [VirtualClassList_T](#) & [getVirtualClassList](#) ()
- [BidPriceVector_T](#) & [getBidPriceVector](#) ()
- const [YieldLevelDemandMap_T](#) [getYieldLevelDemandMap](#) ()

- void [setCapacities](#) (const [CabinCapacity_T](#) &iCapacity)
- void [setSoldSeat](#) (const [NbOfSeats_T](#) &iSoldSeat)
- void [setCommittedSpace](#) (const [CommittedSpace_T](#) &iCommittedSpace)
- void [setAvailabilityPool](#) (const [Availability_T](#) &iAvailabilityPool)
- void [setAvailability](#) (const [Availability_T](#) &iAvailability)
- void [setCurrentBidPrice](#) (const [BidPrice_T](#) &iBidPrice)
- void [setPreviousBidPrice](#) (const [BidPrice_T](#) &iBidPrice)
- void [updatePreviousBidPrice](#) ()
- void [setRegradeAdjustment](#) (const [CapacityAdjustment_T](#) &iRegradeAdjustment)
- void [setAuthorizationLevel](#) (const [AuthorizationLevel_T](#) &iAU)
- void [setUPR](#) (const [UPR_T](#) &iUPR)
- void [setNetAvailability](#) (const [Availability_T](#) &iNAV)
- void [setGrossAvailability](#) (const [Availability_T](#) &iGAV)
- void [setAvgCancellationPercentage](#) (const [OverbookingRate_T](#) &iACP)
- void [setETB](#) (const [NbOfSeats_T](#) &iETB)
- void [setStaffNbOfSeats](#) (const [NbOfSeats_T](#) &iStaffSeats)
- void [setWLNbOfSeats](#) (const [NbOfSeats_T](#) &iWLSeats)
- void [setGroupNbOfSeats](#) (const [NbOfSeats_T](#) &iGroupSeats)
- void [updateCurrentBidPrice](#) ()
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- std::string [toString](#) () const
- const std::string [describeKey](#) () const
- const std::string [displayVirtualClassList](#) () const
- void [updateFromReservation](#) (const [NbOfBookings_T](#) &)
- void [addVirtualClass](#) (const [VirtualClassStruct](#) &iVC)
- void [emptyVirtualClassList](#) ()
- void [emptyBidPriceVector](#) ()
- void [addDemandInformation](#) (const [YieldValue_T](#) &, const [MeanValue_T](#) &, const [StdDevValue_T](#) &)
- void [emptyYieldLevelDemandMap](#) ()

Public Attributes

- [CapacityAdjustment_T_dcsRegrade](#)
- [AuthorizationLevel_T_au](#)
- [UPR_T_upr](#)
- [Availability_T_nav](#)
- [Availability_T_gav](#)
- [OverbookingRate_T_acp](#)
- [NbOfSeats_T_etb](#)
- [NbOfSeats_T_staffNbOfBookings](#)
- [NbOfSeats_T_wlNbOfBookings](#)
- [NbOfSeats_T_groupNbOfBookings](#)

Protected Member Functions

- [LegCabin](#) (const [Key_T](#) &)
- [~LegCabin](#) ()

Protected Attributes

- [Key_T _key](#)
- [BomAbstract](#) * [_parent](#)
- [HolderMap_T _holderMap](#)
- [CabinCapacity_T _offeredCapacity](#)
- [CabinCapacity_T _physicalCapacity](#)
- [NbOfSeats_T _soldSeat](#)
- [CommittedSpace_T _committedSpace](#)
- [Availability_T _availabilityPool](#)
- [Availability_T _availability](#)
- [BidPrice_T _currentBidPrice](#)
- [BidPrice_T _previousBidPrice](#)
- [BidPriceVector_T _bidPriceVector](#)
- [VirtualClassList_T _virtualClassList](#)
- [YieldLevelDemandMap_T _yieldLevelDemandMap](#)

Friends

- class [FacBom](#)
- class [FacBomManager](#)

34.84.1 Detailed Description

Class representing the actual attributes for an airline leg-cabin.

Definition at line 24 of file [LegCabin.hpp](#).

34.84.2 Member Typedef Documentation

34.84.2.1 typedef [LegCabinKey](#) [stdair::LegCabin::Key_T](#)

Definition allowing to retrieve the associated BOM key type.

Definition at line 33 of file [LegCabin.hpp](#).

34.84.3 Constructor & Destructor Documentation

34.84.3.1 [stdair::LegCabin::LegCabin](#) (const [Key_T](#) & *iKey*) [protected]

Constructor.

Definition at line 30 of file [LegCabin.cpp](#).

34.84.3.2 stdair::LegCabin::~~LegCabin () [protected]

Destructor.

Definition at line 43 of file [LegCabin.cpp](#).

34.84.4 Member Function Documentation

34.84.4.1 const Key_T& stdair::LegCabin::getKey () const [inline]

Get the leg-cabin key (cabin code).

Definition at line 40 of file [LegCabin.hpp](#).

References [_key](#).

34.84.4.2 BomAbstract* const stdair::LegCabin::getParent () const [inline]

Get the parent object.

Definition at line 47 of file [LegCabin.hpp](#).

References [_parent](#).

34.84.4.3 const CabinCode_T& stdair::LegCabin::getCabinCode () const [inline]

Get the cabin code (from key).

Definition at line 54 of file [LegCabin.hpp](#).

References [_key](#), and [stdair::LegCabinKey::getCabinCode\(\)](#).

Referenced by [getFullerKey\(\)](#), and [stdair::jsonLegCabinExport\(\)](#).

34.84.4.4 const MapKey_T stdair::LegCabin::getFullerKey () const

Get the (leg-date, leg-cabin) key (board point and cabin code).

Note

That method assumes that the parent object derives from the [SegmentDate](#) class, as it needs to have access to the [describeKey\(\)](#) method.

Definition at line 54 of file [LegCabin.cpp](#).

References [stdair::DEFAULT_KEY_FLD_DELIMITER](#), [stdair::LegDate::describeKey\(\)](#), and [getCabinCode\(\)](#).

34.84.4.5 const HolderMap_T& stdair::LegCabin::getHolderMap () const [inline]

Get the map of children holders.

Definition at line 70 of file [LegCabin.hpp](#).

References [_holderMap](#).

34.84.4.6 **const CabinCapacity_T& stdair::LegCabin::getOfferedCapacity () const**
[inline]

Get the cabin offered capacity.

Definition at line 75 of file [LegCabin.hpp](#).

References [_offeredCapacity](#).

34.84.4.7 **const CabinCapacity_T& stdair::LegCabin::getPhysicalCapacity () const**
[inline]

Get the cabin physical capacity.

Definition at line 80 of file [LegCabin.hpp](#).

References [_physicalCapacity](#).

34.84.4.8 **const NbOfSeats_T& stdair::LegCabin::getSoldSeat () const** [inline]

Get the number of sold seat.

Definition at line 85 of file [LegCabin.hpp](#).

References [_soldSeat](#).

34.84.4.9 **const CommittedSpace_T& stdair::LegCabin::getCommittedSpace () const**
[inline]

Get the value of committed space.

Definition at line 90 of file [LegCabin.hpp](#).

References [_committedSpace](#).

34.84.4.10 **const Availability_T& stdair::LegCabin::getAvailabilityPool () const**
[inline]

Get the value of the availability pool.

Definition at line 95 of file [LegCabin.hpp](#).

References [_availabilityPool](#).

34.84.4.11 **const Availability_T& stdair::LegCabin::getAvailability () const** [inline]

Get the value of the availability.

Definition at line 100 of file [LegCabin.hpp](#).

References [_availability](#).

34.84.4.12 **const BidPrice_T& stdair::LegCabin::getCurrentBidPrice () const** [inline]

Get the current Bid-Price.

Definition at line 105 of file [LegCabin.hpp](#).

References [_currentBidPrice](#).

34.84.4.13 **const BidPrice_T& stdair::LegCabin::getPreviousBidPrice () const**
[inline]

Get the previous Bid-Price.

Definition at line 110 of file [LegCabin.hpp](#).

References [_previousBidPrice](#).

34.84.4.14 **const BidPriceVector_T& stdair::LegCabin::getBidPriceVector () const**
[inline]

Get the Bid-Price Vector.

Definition at line 115 of file [LegCabin.hpp](#).

References [_bidPriceVector](#).

34.84.4.15 **const CapacityAdjustment_T& stdair::LegCabin::getRegradeAdjustment () const**
const [inline]

Get the capacity adjustment due to check-in (DCS) regrade.

Definition at line 120 of file [LegCabin.hpp](#).

References [_dcsRegrade](#).

34.84.4.16 **const AuthorizationLevel_T& stdair::LegCabin::getAuthorizationLevel () const**
[inline]

Authorisation Level (AU).

Definition at line 125 of file [LegCabin.hpp](#).

References [_au](#).

34.84.4.17 **const UPR_T& stdair::LegCabin::getUPR () const** [inline]

Unsold Protection (UPR).

Definition at line 130 of file [LegCabin.hpp](#).

References [_upr](#).

34.84.4.18 **const Availability_T& stdair::LegCabin::getNetAvailability () const**
[inline]

Net Availability (NAV).

Definition at line 135 of file [LegCabin.hpp](#).

References [_nav](#).

34.84.4.19 **const Availability_T& stdair::LegCabin::getGrossAvailability () const**
[inline]

Gross Availability (GAV).

Definition at line 140 of file [LegCabin.hpp](#).

References [_gav](#).

34.84.4.20 `const OverbookingRate_T& stdair::LegCabin::getAvgCancellationPercentage ()
const [inline]`

Average Cancellation Percentage (ACP).

Definition at line 145 of file [LegCabin.hpp](#).

References [_acp](#).

34.84.4.21 `const NbOfSeats_T& stdair::LegCabin::getETB () const [inline]`

Expected to Board (ETB).

Definition at line 150 of file [LegCabin.hpp](#).

References [_etb](#).

34.84.4.22 `const NbOfSeats_T& stdair::LegCabin::getStaffNbOfSeats () const
[inline]`

Number of staff bookings.

Definition at line 155 of file [LegCabin.hpp](#).

References [_staffNbOfBookings](#).

34.84.4.23 `const NbOfSeats_T& stdair::LegCabin::getWLNbOfSeats () const
[inline]`

Number of wait-listed bookings.

Definition at line 160 of file [LegCabin.hpp](#).

References [_wLNbOfBookings](#).

34.84.4.24 `const NbOfSeats_T& stdair::LegCabin::getGroupNbOfSeats () const
[inline]`

Number of group bookings.

Definition at line 165 of file [LegCabin.hpp](#).

References [_groupNbOfBookings](#).

34.84.4.25 `VirtualClassList_T& stdair::LegCabin::getVirtualClassList () [inline]`

The virtual class list.

Definition at line 170 of file [LegCabin.hpp](#).

References [_virtualClassList](#).

34.84.4.26 BidPriceVector_T& stdair::LegCabin::getBidPriceVector () [inline]

Reset the bid price vector and return it.

Definition at line 175 of file [LegCabin.hpp](#).

References [_bidPriceVector](#).

34.84.4.27 const YieldLevelDemandMap_T stdair::LegCabin::getYieldLevelDemandMap () [inline]

Get the yield-demand map.

Definition at line 181 of file [LegCabin.hpp](#).

References [_yieldLevelDemandMap](#).

34.84.4.28 void stdair::LegCabin::setCapacities (const CabinCapacity_T & iCapacity)

Set the offered and physical capacities.

Definition at line 47 of file [LegCabin.cpp](#).

References [_committedSpace](#), [_offeredCapacity](#), [_physicalCapacity](#), and [setAvailabilityPool\(\)](#).

34.84.4.29 void stdair::LegCabin::setSoldSeat (const NbOfSeats_T & iSoldSeat) [inline]

Set the number of sold seat.

Definition at line 192 of file [LegCabin.hpp](#).

References [_soldSeat](#).

34.84.4.30 void stdair::LegCabin::setCommittedSpace (const CommittedSpace_T & iCommittedSpace) [inline]

Set the value of committed space.

Definition at line 197 of file [LegCabin.hpp](#).

References [_committedSpace](#).

34.84.4.31 void stdair::LegCabin::setAvailabilityPool (const Availability_T & iAvailabilityPool) [inline]

Set the value of availability pool.

Definition at line 202 of file [LegCabin.hpp](#).

References [_availabilityPool](#).

Referenced by [setCapacities\(\)](#).

34.84.4.32 void stdair::LegCabin::setAvailability (const Availability_T & iAvailability)
[inline]

Set the value of availability.

Definition at line 207 of file [LegCabin.hpp](#).

References [_availability](#).

34.84.4.33 void stdair::LegCabin::setCurrentBidPrice (const BidPrice_T & iBidPrice)
[inline]

Set the current Bid-Price.

Definition at line 212 of file [LegCabin.hpp](#).

References [_currentBidPrice](#).

34.84.4.34 void stdair::LegCabin::setPreviousBidPrice (const BidPrice_T & iBidPrice)
[inline]

Set the previous Bid-Price.

Definition at line 217 of file [LegCabin.hpp](#).

References [_previousBidPrice](#).

34.84.4.35 void stdair::LegCabin::updatePreviousBidPrice () [inline]

Update the previous bid price value with the current one.

Definition at line 222 of file [LegCabin.hpp](#).

References [_currentBidPrice](#), and [_previousBidPrice](#).

34.84.4.36 void stdair::LegCabin::setRegradeAdjustment (const CapacityAdjustment_T & iRegradeAdjustment) [inline]

Get the capacity adjustment due to check-in (DCS) regrade.

Definition at line 227 of file [LegCabin.hpp](#).

References [_dcsRegrade](#).

34.84.4.37 void stdair::LegCabin::setAuthorizationLevel (const AuthorizationLevel_T & iAU) [inline]

Set the Authorisation Level (AU).

Definition at line 232 of file [LegCabin.hpp](#).

References [_au](#).

34.84.4.38 void stdair::LegCabin::setUPR (const UPR_T & iUPR) [inline]

Set the Unsold Protection (UPR).

Definition at line 237 of file [LegCabin.hpp](#).

References [_upr](#).

34.84.4.39 void stdair::LegCabin::setNetAvailability (const Availability_T & iNAV)
[inline]

Set the Net Availability (NAV).

Definition at line 242 of file [LegCabin.hpp](#).

References [_nav](#).

34.84.4.40 void stdair::LegCabin::setGrossAvailability (const Availability_T & iGAV)
[inline]

Set the Gross Availability (GAV).

Definition at line 247 of file [LegCabin.hpp](#).

References [_gav](#).

34.84.4.41 void stdair::LegCabin::setAvgCancellationPercentage (const
OverbookingRate_T & iACP) [inline]

Set the Average Cancellation Percentage (ACP).

Definition at line 252 of file [LegCabin.hpp](#).

References [_acp](#).

34.84.4.42 void stdair::LegCabin::setETB (const NbOfSeats_T & iETB) [inline]

Set the Expected to Board (ETB).

Definition at line 257 of file [LegCabin.hpp](#).

References [_etb](#).

34.84.4.43 void stdair::LegCabin::setStaffNbOfSeats (const NbOfSeats_T & iStaffSeats)
[inline]

Set the Number of staff sold seats.

Definition at line 262 of file [LegCabin.hpp](#).

References [_staffNbOfBookings](#).

34.84.4.44 void stdair::LegCabin::setWLNbOfSeats (const NbOfSeats_T & iWLSets)
[inline]

Set the Number of wait-listed sold seats.

Definition at line 267 of file [LegCabin.hpp](#).

References [_wLNbOfBookings](#).

34.84.4.45 void stdair::LegCabin::setGroupNbOfSeats (const NbOfSeats_T & iGroupSeats)
[inline]

Set the Number of group sold seats.

Definition at line 272 of file [LegCabin.hpp](#).

References [_groupNbOfBookings](#).

34.84.4.46 void stdair::LegCabin::updateCurrentBidPrice ()

Update the bid price (from bid price vector if not empty).

Definition at line 94 of file [LegCabin.cpp](#).

References [_availabilityPool](#), [_bidPriceVector](#), and [_currentBidPrice](#).

34.84.4.47 void stdair::LegCabin::toStream (std::ostream & ioOut) const [inline,
virtual]

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Implements [stdair::BomAbstract](#).

Definition at line 286 of file [LegCabin.hpp](#).

References [toString\(\)](#).

34.84.4.48 void stdair::LegCabin::fromStream (std::istream & ioIn) [inline,
virtual]

Read a Business Object from an input stream.

Parameters

<i>istream&</i> the input stream.

Implements [stdair::BomAbstract](#).

Definition at line 294 of file [LegCabin.hpp](#).

34.84.4.49 std::string stdair::LegCabin::toString () const [virtual]

Get the serialised version of the Business Object.

Implements [stdair::BomAbstract](#).

Definition at line 63 of file [LegCabin.cpp](#).

References [describeKey\(\)](#).

Referenced by [toStream\(\)](#).

34.84.4.50 `const std::string stdair::LegCabin::describeKey () const [inline]`

Get a string describing the key.

Definition at line 305 of file [LegCabin.hpp](#).

References [_key](#), and [stdair::LegCabinKey::toString\(\)](#).

Referenced by [toString\(\)](#).

34.84.4.51 `const std::string stdair::LegCabin::displayVirtualClassList () const`

Display the virtual class list content.

Definition at line 70 of file [LegCabin.cpp](#).

References [_virtualClassList](#), [stdair::VirtualClassStruct::getCumulatedBookingLimit\(\)](#), [stdair::VirtualClassStruct::getCumulatedBookingLimit\(\)](#), and [stdair::VirtualClassStruct::getYield\(\)](#).

34.84.4.52 `void stdair::LegCabin::updateFromReservation (const NbOfBookings_T & iNbOfBookings)`

Register a sale.

Definition at line 88 of file [LegCabin.cpp](#).

References [_availabilityPool](#), [_committedSpace](#), and [_offeredCapacity](#).

34.84.4.53 `void stdair::LegCabin::addVirtualClass (const VirtualClassStruct & iVC) [inline]`

Add a virtual class to the list.

Definition at line 325 of file [LegCabin.hpp](#).

References [_virtualClassList](#).

34.84.4.54 `void stdair::LegCabin::emptyVirtualClassList () [inline]`

Empty the virtual class list.

Definition at line 332 of file [LegCabin.hpp](#).

References [_virtualClassList](#).

34.84.4.55 `void stdair::LegCabin::emptyBidPriceVector () [inline]`

Empty the bid price vector.

Definition at line 339 of file [LegCabin.hpp](#).

References [_bidPriceVector](#).

34.84.4.56 `void stdair::LegCabin::addDemandInformation (const YieldValue_T & iYield, const MeanValue_T & iMeanValue, const StdDevValue_T & iStdDevValue)`

Add demand information.

Definition at line 107 of file [LegCabin.cpp](#).

References [_yieldLevelDemandMap](#).

34.84.4.57 void stdair::LegCabin::emptyYieldLevelDemandMap () [inline]

Reset the (yield level,demand) map.

Definition at line 352 of file [LegCabin.hpp](#).

References [_yieldLevelDemandMap](#).

34.84.5 Friends And Related Function Documentation

34.84.5.1 friend class FacBom [friend]

Definition at line 25 of file [LegCabin.hpp](#).

34.84.5.2 friend class FacBomManager [friend]

Definition at line 26 of file [LegCabin.hpp](#).

34.84.6 Member Data Documentation

34.84.6.1 Key_T stdair::LegCabin::_key [protected]

Primary key (cabin code).

Definition at line 385 of file [LegCabin.hpp](#).

Referenced by [describeKey\(\)](#), [getCabinCode\(\)](#), and [getKey\(\)](#).

34.84.6.2 BomAbstract* stdair::LegCabin::_parent [protected]

Pointer on the parent class ([LegDate](#)).

Definition at line 390 of file [LegCabin.hpp](#).

Referenced by [getParent\(\)](#).

34.84.6.3 HolderMap_T stdair::LegCabin::_holderMap [protected]

Map holding the children ([Bucket](#) objects).

Definition at line 395 of file [LegCabin.hpp](#).

Referenced by [getHolderMap\(\)](#).

34.84.6.4 CabinCapacity_T stdair::LegCabin::_offeredCapacity [protected]

Saleable capacity of the cabin.

Definition at line 398 of file [LegCabin.hpp](#).

Referenced by [getOfferedCapacity\(\)](#), [setCapacities\(\)](#), and [updateFromReservation\(\)](#).

34.84.6.5 CabinCapacity_T stdair::LegCabin::_physicalCapacity [protected]

Physical capacity of the cabin.

Definition at line 401 of file [LegCabin.hpp](#).

Referenced by [getPhysicalCapacity\(\)](#), and [setCapacities\(\)](#).

34.84.6.6 NbOfSeats_T stdair::LegCabin::_soldSeat [protected]

Aggregated number of sold seats.

Definition at line 404 of file [LegCabin.hpp](#).

Referenced by [getSoldSeat\(\)](#), and [setSoldSeat\(\)](#).

34.84.6.7 CommittedSpace_T stdair::LegCabin::_committedSpace
[protected]

Definition at line 407 of file [LegCabin.hpp](#).

Referenced by [getCommittedSpace\(\)](#), [setCapacities\(\)](#), [setCommittedSpace\(\)](#), and [updateFromReservation\(\)](#).

34.84.6.8 Availability_T stdair::LegCabin::_availabilityPool [protected]

Availability pool.

Definition at line 410 of file [LegCabin.hpp](#).

Referenced by [getAvailabilityPool\(\)](#), [setAvailabilityPool\(\)](#), [updateCurrentBidPrice\(\)](#), and [updateFromReservation\(\)](#).

34.84.6.9 Availability_T stdair::LegCabin::_availability [protected]

Availability.

Definition at line 413 of file [LegCabin.hpp](#).

Referenced by [getAvailability\(\)](#), and [setAvailability\(\)](#).

34.84.6.10 BidPrice_T stdair::LegCabin::_currentBidPrice [protected]

Current Bid-Price (BP).

Definition at line 416 of file [LegCabin.hpp](#).

Referenced by [getCurrentBidPrice\(\)](#), [setCurrentBidPrice\(\)](#), [updateCurrentBidPrice\(\)](#), and [updatePreviousBidPrice\(\)](#).

34.84.6.11 BidPrice_T stdair::LegCabin::_previousBidPrice [protected]

Previous Bid-Price (BP).

Definition at line 419 of file [LegCabin.hpp](#).

Referenced by [getPreviousBidPrice\(\)](#), [setPreviousBidPrice\(\)](#), and [updatePreviousBidPrice\(\)](#).

34.84.6.12 BidPriceVector_T stdair::LegCabin::_bidPriceVector [protected]

Bid-Price Vector (BPV).

Definition at line 422 of file [LegCabin.hpp](#).

Referenced by [emptyBidPriceVector\(\)](#), [getBidPriceVector\(\)](#), and [updateCurrentBidPrice\(\)](#).

34.84.6.13 VirtualClassList_T stdair::LegCabin::_virtualClassList
[protected]

List of virtual classes (for revenue management optimisation).

Definition at line 425 of file [LegCabin.hpp](#).

Referenced by [addVirtualClass\(\)](#), [displayVirtualClassList\(\)](#), [emptyVirtualClassList\(\)](#), and [getVirtualClassList\(\)](#).

34.84.6.14 YieldLevelDemandMap_T stdair::LegCabin::_yieldLevelDemandMap
[protected]

Map holding the demand information indexed by yield.

Definition at line 428 of file [LegCabin.hpp](#).

Referenced by [addDemandInformation\(\)](#), [emptyYieldLevelDemandMap\(\)](#), and [getYieldLevelDemandMap\(\)](#).

34.84.6.15 CapacityAdjustment_T stdair::LegCabin::_dcsRegrade

Capacity adjustment of the cabin, due to check-in (DCS) regrade.

Definition at line 433 of file [LegCabin.hpp](#).

Referenced by [getRegradeAdjustment\(\)](#), and [setRegradeAdjustment\(\)](#).

34.84.6.16 AuthorizationLevel_T stdair::LegCabin::_au

Authorisation Level (AU).

Definition at line 436 of file [LegCabin.hpp](#).

Referenced by [getAuthorizationLevel\(\)](#), and [setAuthorizationLevel\(\)](#).

34.84.6.17 UPR_T stdair::LegCabin::_upr

Unsold Protection (UPR).

Definition at line 439 of file [LegCabin.hpp](#).

Referenced by [getUPR\(\)](#), and [setUPR\(\)](#).

34.84.6.18 Availability_T stdair::LegCabin::_nav

Net Availability (NAV).

Definition at line 442 of file [LegCabin.hpp](#).

Referenced by [getNetAvailability\(\)](#), and [setNetAvailability\(\)](#).

34.84.6.19 Availability_T stdair::LegCabin::_gav

Gross Availability (GAV).

Definition at line 445 of file [LegCabin.hpp](#).

Referenced by [getGrossAvailability\(\)](#), and [setGrossAvailability\(\)](#).

34.84.6.20 OverbookingRate_T stdair::LegCabin::_acp

Average Cancellation Percentage (ACP).

Definition at line 448 of file [LegCabin.hpp](#).

Referenced by [getAvgCancellationPercentage\(\)](#), and [setAvgCancellationPercentage\(\)](#).

34.84.6.21 NbOfSeats_T stdair::LegCabin::_etb

Expected to Board (ETB).

Definition at line 451 of file [LegCabin.hpp](#).

Referenced by [getETB\(\)](#), and [setETB\(\)](#).

34.84.6.22 NbOfSeats_T stdair::LegCabin::_staffNbOfBookings

Number of staff bookings.

Definition at line 454 of file [LegCabin.hpp](#).

Referenced by [getStaffNbOfSeats\(\)](#), and [setStaffNbOfSeats\(\)](#).

34.84.6.23 NbOfSeats_T stdair::LegCabin::_wlnNbOfBookings

Number of wait-listed bookings.

Definition at line 457 of file [LegCabin.hpp](#).

Referenced by [getWLNbOfSeats\(\)](#), and [setWLNbOfSeats\(\)](#).

34.84.6.24 NbOfSeats_T stdair::LegCabin::_groupNbOfBookings

Number of group bookings.

Definition at line 460 of file [LegCabin.hpp](#).

Referenced by [getGroupNbOfSeats\(\)](#), and [setGroupNbOfSeats\(\)](#).

The documentation for this class was generated from the following files:

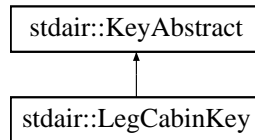
- [stdair/bom/LegCabin.hpp](#)
- [stdair/bom/LegCabin.cpp](#)

34.85 stdair::LegCabinKey Struct Reference

Key of a given leg-cabin, made of a cabin code (only).

```
#include <stdair/bom/LegCabinKey.hpp>
```

Inheritance diagram for stdair::LegCabinKey:



Public Member Functions

- [LegCabinKey](#) (const [CabinCode_T](#) &iCabinCode)
- [LegCabinKey](#) (const [LegCabinKey](#) &)
- [~LegCabinKey](#) ()
- const [CabinCode_T](#) & [getCabinCode](#) () const
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- const std::string [toString](#) () const
- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int iFileVersion)

Friends

- class [boost::serialization::access](#)

34.85.1 Detailed Description

Key of a given leg-cabin, made of a cabin code (only).

Definition at line 26 of file [LegCabinKey.hpp](#).

34.85.2 Constructor & Destructor Documentation

34.85.2.1 stdair::LegCabinKey::LegCabinKey (const CabinCode_T & iCabinCode)

Constructor.

Definition at line 23 of file [LegCabinKey.cpp](#).

34.85.2.2 stdair::LegCabinKey::LegCabinKey (const LegCabinKey & iKey)

Copy constructor.

Definition at line 28 of file [LegCabinKey.cpp](#).

34.85.2.3 stdair::LegCabinKey::~~LegCabinKey ()

Destructor.

Definition at line 33 of file [LegCabinKey.cpp](#).

34.85.3 Member Function Documentation

34.85.3.1 `const CabinCode_T& stdair::LegCabinKey::getCabinCode () const`
[inline]

Get the cabin code.

Definition at line 56 of file [LegCabinKey.hpp](#).

Referenced by [stdair::LegCabin::getCabinCode\(\)](#).

34.85.3.2 `void stdair::LegCabinKey::toStream (std::ostream & ioOut) const` [virtual]

Dump a Business Object Key into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 37 of file [LegCabinKey.cpp](#).

References [toString\(\)](#).

34.85.3.3 `void stdair::LegCabinKey::fromStream (std::istream & ioin)` [virtual]

Read a Business Object Key from an input stream.

Parameters

<i>istream&</i> the input stream.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 42 of file [LegCabinKey.cpp](#).

34.85.3.4 `const std::string stdair::LegCabinKey::toString () const` [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 46 of file [LegCabinKey.cpp](#).

Referenced by [stdair::LegCabin::describeKey\(\)](#), [stdair::LegDate::getLegCabin\(\)](#), and [toStream\(\)](#).

34.85.3.5 `template<class Archive > void stdair::LegCabinKey::serialize (Archive & ar, const unsigned int iFileVersion)`

Serialisation.

Definition at line 68 of file [LegCabinKey.cpp](#).

34.85.4 Friends And Related Function Documentation

34.85.4.1 friend class boost::serialization::access [friend]

Definition at line 27 of file [LegCabinKey.hpp](#).

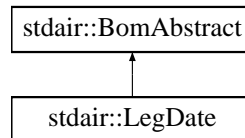
The documentation for this struct was generated from the following files:

- [stdair/bom/LegCabinKey.hpp](#)
- [stdair/bom/LegCabinKey.cpp](#)

34.86 stdair::LegDate Class Reference

```
#include <stdair/bom/LegDate.hpp>
```

Inheritance diagram for stdair::LegDate:



Public Types

- typedef [LegDateKey](#) Key_T

Public Member Functions

- const [Key_T](#) & [getKey](#) () const
- [BomAbstract](#) *const [getParent](#) () const
- const [AirportCode_T](#) & [getBoardingPoint](#) () const
- const [AirlineCode_T](#) & [getAirlineCode](#) () const
- const [HolderMap_T](#) & [getHolderMap](#) () const
- [LegCabin](#) * [getLegCabin](#) (const std::string &iLegCabinKeyStr) const
- [LegCabin](#) * [getLegCabin](#) (const [LegCabinKey](#) &) const
- const [AirportCode_T](#) & [getOffPoint](#) () const
- const [Date_T](#) & [getBoardingDate](#) () const
- const [Duration_T](#) & [getBoardingTime](#) () const
- const [Date_T](#) & [getOffDate](#) () const
- const [Duration_T](#) & [getOffTime](#) () const
- const [Duration_T](#) & [getElapsedTime](#) () const
- const [Distance_T](#) & [getDistance](#) () const
- const [CabinCapacity_T](#) & [getCapacity](#) () const

- const [DateOffset_T](#) [getDateOffset](#) () const
- const [Duration_T](#) [getTimeOffset](#) () const
- void [setOffPoint](#) (const [AirportCode_T](#) &iOffPoint)
- void [setBoardingDate](#) (const [Date_T](#) &iBoardingDate)
- void [setBoardingTime](#) (const [Duration_T](#) &iBoardingTime)
- void [setOffDate](#) (const [Date_T](#) &iOffDate)
- void [setOffTime](#) (const [Duration_T](#) &iOffTime)
- void [setElapsedTime](#) (const [Duration_T](#) &)
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- std::string [toString](#) () const
- const std::string [describeKey](#) () const

Protected Member Functions

- [LegDate](#) (const [Key_T](#) &)
- virtual [~LegDate](#) ()

Protected Attributes

- [Key_T_key](#)
- [BomAbstract](#) * [_parent](#)
- [HolderMap_T_holderMap](#)
- [AirportCode_T_offPoint](#)
- [Date_T_boardingDate](#)
- [Duration_T_boardingTime](#)
- [Date_T_offDate](#)
- [Duration_T_offTime](#)
- [Duration_T_elapsedTime](#)
- [Distance_T_distance](#)
- [CabinCapacity_T_capacity](#)

Friends

- class [FacBom](#)
- class [FacBomManager](#)

34.86.1 Detailed Description

Class representing the actual attributes for an airline leg-date.

Definition at line 25 of file [LegDate.hpp](#).

34.86.2 Member Typedef Documentation

34.86.2.1 typedef LegDateKey stdair::LegDate::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 32 of file [LegDate.hpp](#).

34.86.3 Constructor & Destructor Documentation

34.86.3.1 stdair::LegDate::LegDate (const Key_T & iKey) [protected]

Constructor.

Definition at line 28 of file [LegDate.cpp](#).

34.86.3.2 stdair::LegDate::~~LegDate () [protected, virtual]

Destructor.

Definition at line 34 of file [LegDate.cpp](#).

34.86.4 Member Function Documentation

34.86.4.1 const Key_T& stdair::LegDate::getKey () const [inline]

Get the leg-date key.

Definition at line 38 of file [LegDate.hpp](#).

References [_key](#).

34.86.4.2 BomAbstract* const stdair::LegDate::getParent () const [inline]

Get the parent object.

Definition at line 43 of file [LegDate.hpp](#).

References [_parent](#).

Referenced by [getAirlineCode\(\)](#).

34.86.4.3 const AirportCode_T& stdair::LegDate::getBoardingPoint () const [inline]

Get the boarding point (part of the primary key).

Definition at line 48 of file [LegDate.hpp](#).

References [_key](#), and [stdair::LegDateKey::getBoardingPoint\(\)](#).

Referenced by [stdair::jsonLegDateExport\(\)](#).

34.86.4.4 const AirlineCode_T & stdair::LegDate::getAirlineCode () const

Get the airline code (key of the parent object).

Note

That method assumes that the parent object derives from the [Inventory](#) class, as it needs to have access to the [getAirlineCode\(\)](#) method.

Definition at line 38 of file [LegDate.cpp](#).

References [stdair::FlightDate::getAirlineCode\(\)](#), and [getParent\(\)](#).

34.86.4.5 `const HolderMap_T& stdair::LegDate::getHolderMap () const` [inline]

Get the map of children holders.

Definition at line 64 of file [LegDate.hpp](#).

References [_holderMap](#).

34.86.4.6 `LegCabin * stdair::LegDate::getLegCabin (const std::string & iLegCabinKeyStr) const`

Get a pointer on the [LegCabin](#) object corresponding to the given key.

Note

The [LegCabin](#) object can be inherited from, if needed. In that case, a `dynamic_cast<>` may be needed.

Parameters

<code>const</code> std::string& The leg-cabin key.
--

Returns

[LegCabin*](#) Found [LegCabin](#) object. NULL if not found.

Definition at line 53 of file [LegDate.cpp](#).

Referenced by [getLegCabin\(\)](#), and [stdair::BomRetriever::retrieveDummyLegCabin\(\)](#).

34.86.4.7 `LegCabin * stdair::LegDate::getLegCabin (const LegCabinKey & iLegCabinKey) const`

Get a pointer on the [LegCabin](#) object corresponding to the given key.

Note

The [LegCabin](#) object can be inherited from, if needed. In that case, a `dynamic_cast<>` may be needed.

Parameters

<code>const</code> LegCabinKey& The leg-cabin key

Returns

[LegCabin*](#) Found [LegCabin](#) object. NULL if not found.

Definition at line 60 of file [LegDate.cpp](#).

References [getLegCabin\(\)](#), and [stdair::LegCabinKey::toString\(\)](#).

34.86.4.8 `const AirportCode_T& stdair::LegDate::getOffPoint () const` `[inline]`

Get the off point.

Definition at line 93 of file [LegDate.hpp](#).

References [_offPoint](#).

Referenced by [stdair::jsonLegDateExport\(\)](#).

34.86.4.9 `const Date_T& stdair::LegDate::getBoardingDate () const` `[inline]`

Get the boarding date.

Definition at line 98 of file [LegDate.hpp](#).

References [_boardingDate](#).

Referenced by [stdair::jsonLegDateExport\(\)](#).

34.86.4.10 `const Duration_T& stdair::LegDate::getBoardingTime () const` `[inline]`

Get the boarding time.

Definition at line 103 of file [LegDate.hpp](#).

References [_boardingTime](#).

Referenced by [stdair::jsonLegDateExport\(\)](#).

34.86.4.11 `const Date_T& stdair::LegDate::getOffDate () const` `[inline]`

Get the off date.

Definition at line 108 of file [LegDate.hpp](#).

References [_offDate](#).

Referenced by [stdair::jsonLegDateExport\(\)](#).

34.86.4.12 `const Duration_T& stdair::LegDate::getOffTime () const` `[inline]`

Get the off time.

Definition at line 113 of file [LegDate.hpp](#).

References [_offTime](#).

Referenced by [stdair::jsonLegDateExport\(\)](#).

34.86.4.13 `const Duration_T& stdair::LegDate::getElapsedTime () const` `[inline]`

Get the elapsed time.

Definition at line 118 of file [LegDate.hpp](#).

References [_elapsedTime](#).

Referenced by [stdair::jsonLegDateExport\(\)](#).

34.86.4.14 `const Distance_T& stdair::LegDate::getDistance () const` `[inline]`

Get the distance.

Definition at line 123 of file [LegDate.hpp](#).

References [_distance](#).

Referenced by [stdair::jsonLegDateExport\(\)](#).

34.86.4.15 `const CabinCapacity_T& stdair::LegDate::getCapacity () const` `[inline]`

Get the leg capacity.

Definition at line 128 of file [LegDate.hpp](#).

References [_capacity](#).

Referenced by [stdair::jsonLegDateExport\(\)](#).

34.86.4.16 `const DateOffset_T stdair::LegDate::getDateOffset () const` `[inline]`

Get the date offset (off date - boarding date).

Definition at line 133 of file [LegDate.hpp](#).

References [_boardingDate](#), and [_offDate](#).

Referenced by [getTimeOffset\(\)](#), and [stdair::jsonLegDateExport\(\)](#).

34.86.4.17 `const Duration_T stdair::LegDate::getTimeOffset () const`

Get the time off set between boarding and off points.

It is defined as being: $\text{TimeOffset} = (\text{OffTime} - \text{BoardingTime}) + (\text{OffDate} - \text{BoardingDate}) * 24$

- [ElapsedTime](#).

Definition at line 65 of file [LegDate.cpp](#).

References [_boardingTime](#), [_elapsedTime](#), [_offTime](#), and [getDateOffset\(\)](#).

Referenced by [stdair::jsonLegDateExport\(\)](#).

34.86.4.18 `void stdair::LegDate::setOffPoint (const AirportCode_T & iOffPoint)`
`[inline]`

Set the off point.

Definition at line 147 of file [LegDate.hpp](#).

References [_offPoint](#).

34.86.4.19 void stdair::LegDate::setBoardingDate (const Date_T & iBoardingDate)
[inline]

Set the boarding date.

Definition at line 152 of file [LegDate.hpp](#).

References [_boardingDate](#).

34.86.4.20 void stdair::LegDate::setBoardingTime (const Duration_T & iBoardingTime)
[inline]

Set the boarding time.

Definition at line 157 of file [LegDate.hpp](#).

References [_boardingTime](#).

34.86.4.21 void stdair::LegDate::setOffDate (const Date_T & iOffDate) [inline]

Set the off date.

Definition at line 162 of file [LegDate.hpp](#).

References [_offDate](#).

34.86.4.22 void stdair::LegDate::setOffTime (const Duration_T & iOffTime) [inline]

Set the off time.

Definition at line 167 of file [LegDate.hpp](#).

References [_offTime](#).

34.86.4.23 void stdair::LegDate::setElapsedTime (const Duration_T & iElapsedTime)

Set the elapsed time.

Definition at line 80 of file [LegDate.cpp](#).

References [_elapsedTime](#).

34.86.4.24 void stdair::LegDate::toStream (std::ostream & ioOut) const [inline,
virtual]

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Implements [stdair::BomAbstract](#).

Definition at line 183 of file [LegDate.hpp](#).

References [toString\(\)](#).

34.86.4.25 `void stdair::LegDate::fromStream (std::istream & ioln) [inline, virtual]`

Read a Business Object from an input stream.

Parameters

<i>istream&</i> the input stream.

Implements [stdair::BomAbstract](#).

Definition at line 189 of file [LegDate.hpp](#).

34.86.4.26 `std::string stdair::LegDate::toString () const [virtual]`

Get the serialised version of the Business Object.

Implements [stdair::BomAbstract](#).

Definition at line 46 of file [LegDate.cpp](#).

References [describeKey\(\)](#).

Referenced by [toStream\(\)](#).

34.86.4.27 `const std::string stdair::LegDate::describeKey () const [inline]`

Get a string describing the key.

Definition at line 196 of file [LegDate.hpp](#).

References [_key](#), and [stdair::LegDateKey::toString\(\)](#).

Referenced by [stdair::LegCabin::getFullerKey\(\)](#), and [toString\(\)](#).

34.86.5 Friends And Related Function Documentation

34.86.5.1 `friend class FacBom [friend]`

Definition at line 26 of file [LegDate.hpp](#).

34.86.5.2 `friend class FacBomManager [friend]`

Definition at line 27 of file [LegDate.hpp](#).

34.86.6 Member Data Documentation

34.86.6.1 `Key_T stdair::LegDate::_key [protected]`

Primary key (origin airport).

Definition at line 218 of file [LegDate.hpp](#).

Referenced by [describeKey\(\)](#), [getBoardingPoint\(\)](#), and [getKey\(\)](#).

34.86.6.2 BomAbstract* stdair::LegDate::_parent [protected]

Pointer on the parent class ([FlightDate](#)).

Definition at line 221 of file [LegDate.hpp](#).

Referenced by [getParent\(\)](#).

34.86.6.3 HolderMap_T stdair::LegDate::_holderMap [protected]

Map holding the children ([LegCabin](#) objects).

Definition at line 224 of file [LegDate.hpp](#).

Referenced by [getHolderMap\(\)](#).

34.86.6.4 AirportCode_T stdair::LegDate::_offPoint [protected]

Landing airport.

Definition at line 227 of file [LegDate.hpp](#).

Referenced by [getOffPoint\(\)](#), and [setOffPoint\(\)](#).

34.86.6.5 Date_T stdair::LegDate::_boardingDate [protected]

Boarding date.

Definition at line 230 of file [LegDate.hpp](#).

Referenced by [getBoardingDate\(\)](#), [getDateOffset\(\)](#), and [setBoardingDate\(\)](#).

34.86.6.6 Duration_T stdair::LegDate::_boardingTime [protected]

Boarding time.

Definition at line 233 of file [LegDate.hpp](#).

Referenced by [getBoardingTime\(\)](#), [getTimeOffset\(\)](#), and [setBoardingTime\(\)](#).

34.86.6.7 Date_T stdair::LegDate::_offDate [protected]

Landing date.

Definition at line 236 of file [LegDate.hpp](#).

Referenced by [getDateOffset\(\)](#), [getOffDate\(\)](#), and [setOffDate\(\)](#).

34.86.6.8 Duration_T stdair::LegDate::_offTime [protected]

Landing time.

Definition at line 239 of file [LegDate.hpp](#).

Referenced by [getOffTime\(\)](#), [getTimeOffset\(\)](#), and [setOffTime\(\)](#).

34.86.6.9 Duration_T stdair::LegDate::_elapsedTime [protected]

Trip elapsed time.

Definition at line 242 of file [LegDate.hpp](#).

Referenced by [getElapsedTime\(\)](#), [getTimeOffset\(\)](#), and [setElapsedTime\(\)](#).

34.86.6.10 Distance_T stdair::LegDate::_distance [protected]

Trip distance.

Definition at line 245 of file [LegDate.hpp](#).

Referenced by [getDistance\(\)](#).

34.86.6.11 CabinCapacity_T stdair::LegDate::_capacity [protected]

Aggregated capacity for all the leg-cabins.

Definition at line 248 of file [LegDate.hpp](#).

Referenced by [getCapacity\(\)](#).

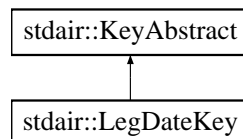
The documentation for this class was generated from the following files:

- [stdair/bom/LegDate.hpp](#)
- [stdair/bom/LegDate.cpp](#)

34.87 stdair::LegDateKey Struct Reference

```
#include <stdair/bom/LegDateKey.hpp>
```

Inheritance diagram for stdair::LegDateKey:



Public Member Functions

- [LegDateKey](#) (const [AirportCode_T](#) &iBoardingPoint)
- [LegDateKey](#) (const [LegDateKey](#) &)
- [~LegDateKey](#) ()
- const [AirportCode_T](#) & [getBoardingPoint](#) () const
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- const std::string [toString](#) () const

34.87.1 Detailed Description

Key of a given leg-date, made of an origin airport.

Definition at line 16 of file [LegDateKey.hpp](#).

34.87.2 Constructor & Destructor Documentation

34.87.2.1 stdair::LegDateKey::LegDateKey (const AirportCode_T & iBoardingPoint)

Constructor.

Definition at line 19 of file [LegDateKey.cpp](#).

34.87.2.2 stdair::LegDateKey::LegDateKey (const LegDateKey & iKey)

Default copy constructor.

Definition at line 24 of file [LegDateKey.cpp](#).

34.87.2.3 stdair::LegDateKey::~~LegDateKey ()

Destructor.

Definition at line 29 of file [LegDateKey.cpp](#).

34.87.3 Member Function Documentation

34.87.3.1 const AirportCode_T& stdair::LegDateKey::getBoardingPoint () const
[inline]

Get the boarding point.

Definition at line 34 of file [LegDateKey.hpp](#).

Referenced by [stdair::LegDate::getBoardingPoint\(\)](#).

34.87.3.2 void stdair::LegDateKey::toStream (std::ostream & ioOut) const [virtual]

Dump a Business Object Key into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 33 of file [LegDateKey.cpp](#).

References [toString\(\)](#).

34.87.3.3 void stdair::LegDateKey::fromStream (std::istream & ioIn) [virtual]

Read a Business Object Key from an input stream.

Parameters

<i>istream&</i> the input stream.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 38 of file [LegDateKey.cpp](#).

34.87.3.4 `const std::string stdair::LegDateKey::toString () const` `[virtual]`

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same leg-date.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 42 of file [LegDateKey.cpp](#).

Referenced by [stdair::LegDate::describeKey\(\)](#), [stdair::FlightDate::getLegDate\(\)](#), and [toStream\(\)](#).

The documentation for this struct was generated from the following files:

- [stdair/bom/LegDateKey.hpp](#)
- [stdair/bom/LegDateKey.cpp](#)

34.88 stdair::Logger Class Reference

```
#include <stdair/service/Logger.hpp>
```

Public Member Functions

- `template<typename T >`
void [log](#) (const [LOG::EN_LogLevel](#) iLevel, const int iLineNumber, const std::string &iFileName, const T &iToBeLogged)

Static Public Member Functions

- static [Logger](#) & [instance](#) ()

Friends

- class [FacSupervisor](#)
Friend classes.
- class [STDAIR_Service](#)

34.88.1 Detailed Description

Class holding the stream for logs.

Note that the error logs are seen as standard output logs, but with a higher level of visibility.

Definition at line 48 of file [Logger.hpp](#).

34.88.2 Member Function Documentation

34.88.2.1 `template<typename T> void stdair::Logger::log (const LOG::EN_LogLevel
iLevel, const int iLineNumber, const std::string & iFileName, const T & iToBeLogged
) [inline]`

Main log entry.

Definition at line 59 of file [Logger.hpp](#).

References [stdair::LOG::_logLevels](#).

34.88.2.2 `Logger & stdair::Logger::instance () [static]`

Return the static [Logger](#) instance.

Definition at line 48 of file [Logger.cpp](#).

34.88.3 Friends And Related Function Documentation

34.88.3.1 `friend class FacSupervisor [friend]`

Friend classes.

Definition at line 50 of file [Logger.hpp](#).

34.88.3.2 `friend class STDAIR_Service [friend]`

Definition at line 51 of file [Logger.hpp](#).

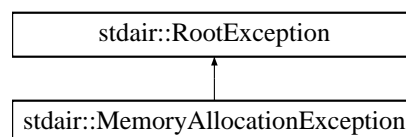
The documentation for this class was generated from the following files:

- [stdair/service/Logger.hpp](#)
- [stdair/service/Logger.cpp](#)

34.89 stdair::MemoryAllocationException Class Reference

```
#include <stdair/stdair_exceptions.hpp>
```

Inheritance diagram for `stdair::MemoryAllocationException`:



Public Member Functions

- [MemoryAllocationException](#) (const std::string &iWhat)
- `const char * what () const throw ()`

Protected Attributes

- `std::string _what`

34.89.1 Detailed Description

Memory allocation.

Definition at line 89 of file [stdair_exceptions.hpp](#).

34.89.2 Constructor & Destructor Documentation

34.89.2.1 `stdair::MemoryAllocationException::MemoryAllocationException (const std::string & iWhat)` `[inline]`

Constructor.

Definition at line 92 of file [stdair_exceptions.hpp](#).

34.89.3 Member Function Documentation

34.89.3.1 `const char* stdair::RootException::what () const throw ()` `[inline, inherited]`

Give the details of the exception.

Definition at line 38 of file [stdair_exceptions.hpp](#).

References [stdair::RootException::_what](#).

34.89.4 Member Data Documentation

34.89.4.1 `std::string stdair::RootException::_what` `[protected, inherited]`

Details for the exception.

Definition at line 46 of file [stdair_exceptions.hpp](#).

Referenced by [stdair::RootException::what\(\)](#).

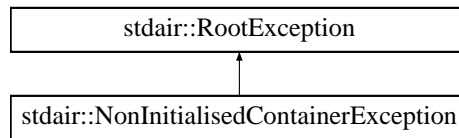
The documentation for this class was generated from the following file:

- [stdair/stdair_exceptions.hpp](#)

34.90 `stdair::NonInitialisedContainerException` Class Reference

```
#include <stdair/stdair_exceptions.hpp>
```

Inheritance diagram for `stdair::NonInitialisedContainerException`:



Public Member Functions

- [NonInitialisedContainerException](#) (const std::string &iWhat)
- const char * [what](#) () const throw ()

Protected Attributes

- std::string [_what](#)

34.90.1 Detailed Description

Non initialised container.

Definition at line 73 of file [stdair_exceptions.hpp](#).

34.90.2 Constructor & Destructor Documentation

34.90.2.1 `stdair::NonInitialisedContainerException::NonInitialisedContainerException (const std::string & iWhat) [inline]`

Constructor.

Definition at line 76 of file [stdair_exceptions.hpp](#).

34.90.3 Member Function Documentation

34.90.3.1 `const char* stdair::RootException::what () const throw () [inline, inherited]`

Give the details of the exception.

Definition at line 38 of file [stdair_exceptions.hpp](#).

References [stdair::RootException::_what](#).

34.90.4 Member Data Documentation

34.90.4.1 `std::string stdair::RootException::_what [protected, inherited]`

Details for the exception.

34.91 stdair::NonInitialisedDBSessionManagerException Class Reference 517

Definition at line 46 of file [stdair_exceptions.hpp](#).

Referenced by [stdair::RootException::what\(\)](#).

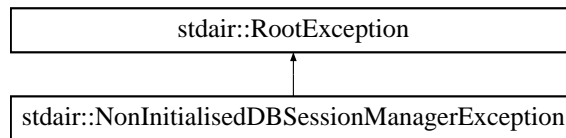
The documentation for this class was generated from the following file:

- [stdair/stdair_exceptions.hpp](#)

34.91 stdair::NonInitialisedDBSessionManagerException Class Reference

```
#include <stdair/stdair_exceptions.hpp>
```

Inheritance diagram for `stdair::NonInitialisedDBSessionManagerException`:



Public Member Functions

- [NonInitialisedDBSessionManagerException](#) (const std::string &iWhat)
- const char * [what](#) () const throw ()

Protected Attributes

- std::string [_what](#)

34.91.1 Detailed Description

Non initialised database session.

Definition at line 180 of file [stdair_exceptions.hpp](#).

34.91.2 Constructor & Destructor Documentation

34.91.2.1 `stdair::NonInitialisedDBSessionManagerException::NonInitialisedDBSessionManagerException (const std::string & iWhat) [inline]`

Constructor.

Definition at line 183 of file [stdair_exceptions.hpp](#).

34.91.3 Member Function Documentation

34.91.3.1 `const char* stdair::RootException::what () const throw ()` [*inline, inherited*]

Give the details of the exception.

Definition at line 38 of file [stdair_exceptions.hpp](#).

References [stdair::RootException::_what](#).

34.91.4 Member Data Documentation

34.91.4.1 `std::string stdair::RootException::_what` [*protected, inherited*]

Details for the exception.

Definition at line 46 of file [stdair_exceptions.hpp](#).

Referenced by [stdair::RootException::what\(\)](#).

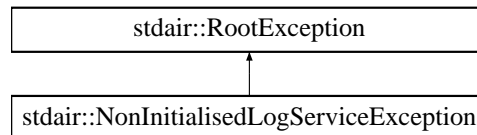
The documentation for this class was generated from the following file:

- [stdair/stdair_exceptions.hpp](#)

34.92 stdair::NonInitialisedLogServiceException Class Reference

```
#include <stdair/stdair_exceptions.hpp>
```

Inheritance diagram for `stdair::NonInitialisedLogServiceException`:



Public Member Functions

- [NonInitialisedLogServiceException](#) (const std::string &iWhat)
- const char * [what](#) () const throw ()

Protected Attributes

- std::string [_what](#)

34.92.1 Detailed Description

Non initialised log service.

Definition at line 57 of file [stdair_exceptions.hpp](#).

34.92.2 Constructor & Destructor Documentation

34.92.2.1 `stdair::NonInitialisedLogServiceException::NonInitialisedLogServiceException (const std::string & iWhat)` `[inline]`

Constructor.

Definition at line 60 of file [stdair_exceptions.hpp](#).

34.92.3 Member Function Documentation

34.92.3.1 `const char* stdair::RootException::what () const throw ()` `[inline, inherited]`

Give the details of the exception.

Definition at line 38 of file [stdair_exceptions.hpp](#).

References [stdair::RootException::_what](#).

34.92.4 Member Data Documentation

34.92.4.1 `std::string stdair::RootException::_what` `[protected, inherited]`

Details for the exception.

Definition at line 46 of file [stdair_exceptions.hpp](#).

Referenced by [stdair::RootException::what\(\)](#).

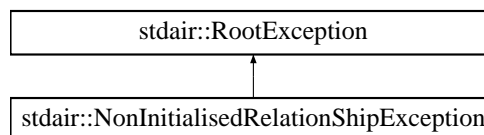
The documentation for this class was generated from the following file:

- [stdair/stdair_exceptions.hpp](#)

34.93 stdair::NonInitialisedRelationshipException Class Reference

```
#include <stdair/stdair_exceptions.hpp>
```

Inheritance diagram for `stdair::NonInitialisedRelationshipException`:



Public Member Functions

- [NonInitialisedRelationshipException](#) (const std::string &iWhat)
- const char * [what](#) () const throw ()

Protected Attributes

- `std::string _what`

34.93.1 Detailed Description

Non initialised relationship.

Definition at line 81 of file [stdair_exceptions.hpp](#).

34.93.2 Constructor & Destructor Documentation

34.93.2.1 `stdair::NonInitialisedRelationshipException::NonInitialisedRelationshipException (const std::string & iWhat) [inline]`

Constructor.

Definition at line 84 of file [stdair_exceptions.hpp](#).

34.93.3 Member Function Documentation

34.93.3.1 `const char* stdair::RootException::what () const throw () [inline, inherited]`

Give the details of the exception.

Definition at line 38 of file [stdair_exceptions.hpp](#).

References [stdair::RootException::_what](#).

34.93.4 Member Data Documentation

34.93.4.1 `std::string stdair::RootException::_what [protected, inherited]`

Details for the exception.

Definition at line 46 of file [stdair_exceptions.hpp](#).

Referenced by [stdair::RootException::what\(\)](#).

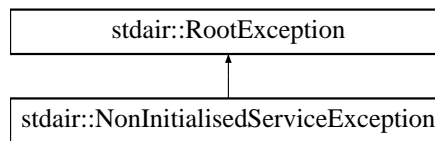
The documentation for this class was generated from the following file:

- [stdair/stdair_exceptions.hpp](#)

34.94 stdair::NonInitialisedServiceException Class Reference

```
#include <stdair/stdair_exceptions.hpp>
```

Inheritance diagram for `stdair::NonInitialisedServiceException`:



Public Member Functions

- [NonInitialisedServiceException](#) (const std::string &iWhat)
- const char * [what](#) () const throw ()

Protected Attributes

- std::string [_what](#)

34.94.1 Detailed Description

Non initialised service.

Definition at line 65 of file [stdair_exceptions.hpp](#).

34.94.2 Constructor & Destructor Documentation

34.94.2.1 `stdair::NonInitialisedServiceException::NonInitialisedServiceException (const std::string & iWhat) [inline]`

Constructor.

Definition at line 68 of file [stdair_exceptions.hpp](#).

34.94.3 Member Function Documentation

34.94.3.1 `const char* stdair::RootException::what () const throw () [inline, inherited]`

Give the details of the exception.

Definition at line 38 of file [stdair_exceptions.hpp](#).

References [stdair::RootException::_what](#).

34.94.4 Member Data Documentation

34.94.4.1 `std::string stdair::RootException::_what [protected, inherited]`

Details for the exception.

Definition at line 46 of file [stdair_exceptions.hpp](#).

Referenced by [stdair::RootException::what\(\)](#).

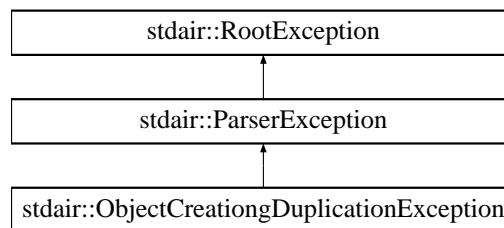
The documentation for this class was generated from the following file:

- [stdair/stdair_exceptions.hpp](#)

34.95 stdair::ObjectCreationgDuplicationException Class Reference

```
#include <stdair/stdair_exceptions.hpp>
```

Inheritance diagram for stdair::ObjectCreationgDuplicationException:



Public Member Functions

- [ObjectCreationgDuplicationException](#) (const std::string &iWhat)
- const char * [what](#) () const throw ()

Protected Attributes

- std::string [_what](#)

34.95.1 Detailed Description

Duplicated object.

Definition at line 149 of file [stdair_exceptions.hpp](#).

34.95.2 Constructor & Destructor Documentation

34.95.2.1 [stdair::ObjectCreationgDuplicationException::ObjectCreationgDuplicationException \(const std::string &iWhat\)](#) [inline]

Constructor.

Definition at line 152 of file [stdair_exceptions.hpp](#).

34.95.3 Member Function Documentation

34.95.3.1 `const char* stdair::RootException::what () const throw ()` [inline, inherited]

Give the details of the exception.

Definition at line 38 of file [stdair_exceptions.hpp](#).

References [stdair::RootException::_what](#).

34.95.4 Member Data Documentation

34.95.4.1 `std::string stdair::RootException::_what` [protected, inherited]

Details for the exception.

Definition at line 46 of file [stdair_exceptions.hpp](#).

Referenced by [stdair::RootException::what\(\)](#).

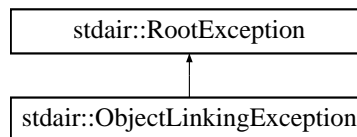
The documentation for this class was generated from the following file:

- [stdair/stdair_exceptions.hpp](#)

34.96 stdair::ObjectLinkingException Class Reference

```
#include <stdair/stdair_exceptions.hpp>
```

Inheritance diagram for `stdair::ObjectLinkingException`:



Public Member Functions

- [ObjectLinkingException](#) (const std::string &iWhat)
- const char * [what](#) () const throw ()

Protected Attributes

- std::string [_what](#)

34.96.1 Detailed Description

Object link.

Definition at line 97 of file [stdair_exceptions.hpp](#).

34.96.2 Constructor & Destructor Documentation

34.96.2.1 `stdair::ObjectLinkingException::ObjectLinkingException (const std::string & iWhat)`
[inline]

Constructor.

Definition at line 100 of file [stdair_exceptions.hpp](#).

34.96.3 Member Function Documentation

34.96.3.1 `const char* stdair::RootException::what () const throw ()` [inline,
inherited]

Give the details of the exception.

Definition at line 38 of file [stdair_exceptions.hpp](#).

References [stdair::RootException::_what](#).

34.96.4 Member Data Documentation

34.96.4.1 `std::string stdair::RootException::_what` [protected, inherited]

Details for the exception.

Definition at line 46 of file [stdair_exceptions.hpp](#).

Referenced by [stdair::RootException::what\(\)](#).

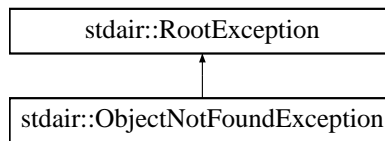
The documentation for this class was generated from the following file:

- [stdair/stdair_exceptions.hpp](#)

34.97 stdair::ObjectNotFoundException Class Reference

```
#include <stdair/stdair_exceptions.hpp>
```

Inheritance diagram for `stdair::ObjectNotFoundException`:



Public Member Functions

- [ObjectNotFoundException](#) (const std::string &iWhat)
- const char * [what](#) () const throw ()

Protected Attributes

- std::string [_what](#)

34.97.1 Detailed Description

Not found object.

Definition at line 157 of file [stdair_exceptions.hpp](#).

34.97.2 Constructor & Destructor Documentation

34.97.2.1 `stdair::ObjectNotFoundException::ObjectNotFoundException (const std::string &iWhat) [inline]`

Constructor.

Definition at line 160 of file [stdair_exceptions.hpp](#).

34.97.3 Member Function Documentation

34.97.3.1 `const char* stdair::RootException::what () const throw () [inline, inherited]`

Give the details of the exception.

Definition at line 38 of file [stdair_exceptions.hpp](#).

References [stdair::RootException::_what](#).

34.97.4 Member Data Documentation

34.97.4.1 `std::string stdair::RootException::_what [protected, inherited]`

Details for the exception.

Definition at line 46 of file [stdair_exceptions.hpp](#).

Referenced by [stdair::RootException::what\(\)](#).

The documentation for this class was generated from the following file:

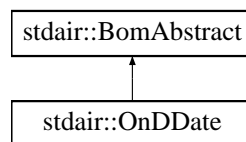
- [stdair/stdair_exceptions.hpp](#)

34.98 stdair::OnDDate Class Reference

Class representing the actual attributes for an airline flight-date.

```
#include <stdair/bom/OnDDate.hpp>
```

Inheritance diagram for stdair::OnDDate:



Public Types

- typedef [OnDDateKey](#) Key_T

Public Member Functions

- const [Key_T](#) & [getKey](#) () const
- [BomAbstract](#) *const [getParent](#) () const
- const [AirlineCode_T](#) & [getAirlineCode](#) () const
- const [stdair::Date_T](#) [getDate](#) () const
- const [stdair::AirportCode_T](#) [getOrigin](#) () const
- const [stdair::AirportCode_T](#) [getDestination](#) () const
- const [HolderMap_T](#) & [getHolderMap](#) () const
- const [StringDemandStructMap_T](#) & [getDemandInfoMap](#) () const
- const [CabinForecastMap_T](#) & [getTotalForecastMap](#) () const
- const [WTPDemandPair_T](#) & [getTotalForecast](#) (const [CabinCode_T](#) &iCC) const
- const [CabinClassPairList_T](#) & [getCabinClassPairList](#) (const std::string &iStr) const
- const short [getNbOfSegments](#) () const
- void [setDemandInformation](#) (const [CabinClassPairList_T](#) &, const [YieldDemandPair_T](#) &)
- void [setTotalForecast](#) (const [CabinCode_T](#) &, const [WTPDemandPair_T](#) &)
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- std::string [toString](#) () const
- const std::string [describeKey](#) () const
- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int iFileVersion)

Protected Member Functions

- [OnDDate](#) (const [Key_T](#) &)
- virtual [~OnDDate](#) ()

Protected Attributes

- [Key_T _key](#)
- [BomAbstract](#) * [_parent](#)
- [HolderMap_T _holderMap](#)
- [StringDemandStructMap_T _classPathDemandMap](#)
- [StringCabinClassPairListMap_T _stringCabinClassPairListMap](#)
- [CabinForecastMap_T _cabinForecastMap](#)

Friends

- class [FacBom](#)
- class [FacBomManager](#)
- class [boost::serialization::access](#)

34.98.1 Detailed Description

Class representing the actual attributes for an airline flight-date.

Definition at line 33 of file [OnDDate.hpp](#).

34.98.2 Member Typedef Documentation

34.98.2.1 typedef OnDDateKey stdair::OnDDate::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 43 of file [OnDDate.hpp](#).

34.98.3 Constructor & Destructor Documentation

34.98.3.1 stdair::OnDDate::OnDDate (const Key_T & iKey) [protected]

Main constructor.

Definition at line 29 of file [OnDDate.cpp](#).

34.98.3.2 stdair::OnDDate::~~OnDDate () [protected, virtual]

Destructor.

Definition at line 34 of file [OnDDate.cpp](#).

34.98.4 Member Function Documentation

34.98.4.1 `const Key_T& stdair::OnDDate::getKey () const` `[inline]`

Get the O&D date key.

Definition at line 49 of file [OnDDate.hpp](#).

References [_key](#).

34.98.4.2 `BomAbstract* const stdair::OnDDate::getParent () const` `[inline]`

Get the parent object.

Definition at line 54 of file [OnDDate.hpp](#).

References [_parent](#).

Referenced by [getAirlineCode\(\)](#).

34.98.4.3 `const AirlineCode_T & stdair::OnDDate::getAirlineCode () const`

Get the airline code (key of the parent object).

Note

That method assumes that the parent object derives from the [Inventory](#) class, as it needs to have access to the [getAirlineCode\(\)](#) method.

Definition at line 45 of file [OnDDate.cpp](#).

References [stdair::Inventory::getAirlineCode\(\)](#), and [getParent\(\)](#).

34.98.4.4 `const stdair::Date_T stdair::OnDDate::getDate () const` `[inline]`

Get the boarding date.

Definition at line 69 of file [OnDDate.hpp](#).

References [_key](#), and [stdair::OnDDateKey::getDate\(\)](#).

34.98.4.5 `const stdair::AirportCode_T stdair::OnDDate::getOrigin () const` `[inline]`

Get the origin.

Definition at line 74 of file [OnDDate.hpp](#).

References [_key](#), and [stdair::OnDDateKey::getOrigin\(\)](#).

34.98.4.6 `const stdair::AirportCode_T stdair::OnDDate::getDestination () const`
`[inline]`

Get the destination.

Definition at line 79 of file [OnDDate.hpp](#).

References [_key](#), and [stdair::OnDDateKey::getDestination\(\)](#).

34.98.4.7 `const HolderMap_T& stdair::OnDDate::getHolderMap () const [inline]`

Get the map of children holders.

Definition at line 86 of file [OnDDate.hpp](#).

References [_holderMap](#).

34.98.4.8 `const StringDemandStructMap_T& stdair::OnDDate::getDemandInfoMap () const [inline]`

Get the map of demand information.

Definition at line 93 of file [OnDDate.hpp](#).

References [_classPathDemandMap](#).

34.98.4.9 `const CabinForecastMap_T& stdair::OnDDate::getTotalForecastMap () const [inline]`

Get the map of total forecast.

Definition at line 100 of file [OnDDate.hpp](#).

References [_cabinForecastMap](#).

34.98.4.10 `const WTPDemandPair_T& stdair::OnDDate::getTotalForecast (const CabinCode_T & iCC) const [inline]`

Get the total forecast for a given cabin.

Definition at line 107 of file [OnDDate.hpp](#).

References [_cabinForecastMap](#).

34.98.4.11 `const CabinClassPairList_T& stdair::OnDDate::getCabinClassPairList (const std::string & iStr) const [inline]`

Get the cabin-class pair out of a string.

Definition at line 115 of file [OnDDate.hpp](#).

References [_stringCabinClassPairListMap](#).

34.98.4.12 `const short stdair::OnDDate::getNbOfSegments () const [inline]`

Get the number of segments of the O&D.

Definition at line 123 of file [OnDDate.hpp](#).

References [_key](#), and [stdair::OnDDateKey::getNbOfSegments\(\)](#).

34.98.4.13 `void stdair::OnDDate::setDemandInformation (const CabinClassPairList_T & iCabinClassPairList, const YieldDemandPair_T & iYieldDemandPair)`

Set demand information.

Definition at line 54 of file [OnDDate.cpp](#).

References [_classPathDemandMap](#), and [_stringCabinClassPairListMap](#).

34.98.4.14 `void stdair::OnDDate::setTotalForecast (const CabinCode_T & iCabinCode, const WTPDemandPair_T & iWTPDemandPair)`

Set forecast information per cabin.

Definition at line 77 of file [OnDDate.cpp](#).

References [_cabinForecastMap](#).

34.98.4.15 `void stdair::OnDDate::toStream (std::ostream & ioOut) const [inline, virtual]`

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Implements [stdair::BomAbstract](#).

Definition at line 146 of file [OnDDate.hpp](#).

References [toString\(\)](#).

34.98.4.16 `void stdair::OnDDate::fromStream (std::istream & ioin) [inline, virtual]`

Read a Business Object from an input stream.

Parameters

<i>istream&</i> the input stream.

Implements [stdair::BomAbstract](#).

Definition at line 155 of file [OnDDate.hpp](#).

34.98.4.17 `std::string stdair::OnDDate::toString () const [virtual]`

Get the serialised version of the Business Object.

Implements [stdair::BomAbstract](#).

Definition at line 38 of file [OnDDate.cpp](#).

References [describeKey\(\)](#).

Referenced by [toStream\(\)](#).

34.98.4.18 `const std::string stdair::OnDDate::describeKey () const [inline]`

Get a string describing the key.

Definition at line 166 of file [OnDDate.hpp](#).

References [_key](#), and [stdair::OnDDateKey::toString\(\)](#).

Referenced by [toString\(\)](#).

34.98.4.19 `template<class Archive > void stdair::OnDDate::serialize (Archive & ar, const unsigned int iFileVersion)`

Serialisation.

34.98.5 Friends And Related Function Documentation

34.98.5.1 `friend class FacBom` `[friend]`

Definition at line 34 of file [OnDDate.hpp](#).

34.98.5.2 `friend class FacBomManager` `[friend]`

Definition at line 35 of file [OnDDate.hpp](#).

34.98.5.3 `friend class boost::serialization::access` `[friend]`

Definition at line 36 of file [OnDDate.hpp](#).

34.98.6 Member Data Documentation

34.98.6.1 `Key_T stdair::OnDDate::_key` `[protected]`

Primary key (list of OnD string keys).

Definition at line 216 of file [OnDDate.hpp](#).

Referenced by [describeKey\(\)](#), [getDate\(\)](#), [getDestination\(\)](#), [getKey\(\)](#), [getNbOfSegments\(\)](#), and [getOrigin\(\)](#).

34.98.6.2 `BomAbstract* stdair::OnDDate::_parent` `[protected]`

Pointer on the parent class ([Inventory](#)).

Definition at line 221 of file [OnDDate.hpp](#).

Referenced by [getParent\(\)](#).

34.98.6.3 `HolderMap_T stdair::OnDDate::_holderMap` `[protected]`

Map holding the children ([SegmentDate](#) and [LegDate](#) objects).

Definition at line 226 of file [OnDDate.hpp](#).

Referenced by [getHolderMap\(\)](#).

34.98.6.4 `StringDemandStructMap_T stdair::OnDDate::_classPathDemandMap` `[protected]`

O&D demand information.

Definition at line 231 of file [OnDDate.hpp](#).

Referenced by [getDemandInfoMap\(\)](#), and [setDemandInformation\(\)](#).

34.98.6.5 StringCabinClassPairListMap_T stdair::OnDDate::_-stringCabinClassPairListMap [protected]

O&D cabin and associated class map.

Definition at line 236 of file [OnDDate.hpp](#).

Referenced by [getCabinClassPairList\(\)](#), and [setDemandInformation\(\)](#).

34.98.6.6 CabinForecastMap_T stdair::OnDDate::_cabinForecastMap [protected]

O&D demand total forecast.

Definition at line 241 of file [OnDDate.hpp](#).

Referenced by [getTotalForecast\(\)](#), [getTotalForecastMap\(\)](#), and [setTotalForecast\(\)](#).

The documentation for this class was generated from the following files:

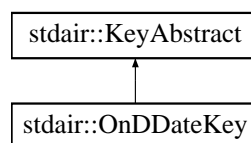
- [stdair/bom/OnDDate.hpp](#)
- [stdair/bom/OnDDate.cpp](#)

34.99 stdair::OnDDateKey Struct Reference

Key of a given O&D-date, made of a list of OnD strings. a OnD string contains the airline code, the flight number, the date and the segment (origin and destination).

```
#include <stdair/bom/OnDDateKey.hpp>
```

Inheritance diagram for stdair::OnDDateKey:



Public Member Functions

- [OnDDateKey](#) (const [OnDStringList_T](#) &)
- [OnDDateKey](#) (const [OnDDateKey](#) &)
- [~OnDDateKey](#) ()
- const [Date_T](#) [getDate](#) () const
- const [AirportCode_T](#) [getOrigin](#) () const
- const [AirportCode_T](#) [getDestination](#) () const
- const short [getNbOfSegments](#) () const

- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- const std::string [toString](#) () const
- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int iFileVersion)

Friends

- class [boost::serialization::access](#)

34.99.1 Detailed Description

Key of a given O&D-date, made of a list of OnD strings. a OnD string contains the airline code, the flight number, the date and the segment (origin and destination).

Definition at line 23 of file [OnDDateKey.hpp](#).

34.99.2 Constructor & Destructor Documentation

34.99.2.1 stdair::OnDDateKey::OnDDateKey (const OnDStringList_T & iOnDStringList)

Constructor.

Definition at line 33 of file [OnDDateKey.cpp](#).

34.99.2.2 stdair::OnDDateKey::OnDDateKey (const OnDDateKey & iKey)

Copy constructor.

Definition at line 38 of file [OnDDateKey.cpp](#).

34.99.2.3 stdair::OnDDateKey::~~OnDDateKey ()

Destructor.

Definition at line 43 of file [OnDDateKey.cpp](#).

34.99.3 Member Function Documentation

34.99.3.1 const Date_T stdair::OnDDateKey::getDate () const

Get the boarding date.

Definition at line 47 of file [OnDDateKey.cpp](#).

References [stdair::BomKeyManager::extractFlightDateKey\(\)](#), and [stdair::FlightDateKey::getDepartureDate\(\)](#).

Referenced by [stdair::OnDDate::getDate\(\)](#).

34.99.3.2 `const AirportCode_T stdair::OnDDateKey::getOrigin () const`

Get the origin.

Definition at line 54 of file [OnDDateKey.cpp](#).

References [stdair::BomKeyManager::extractSegmentDateKey\(\)](#), and [stdair::SegmentDateKey::getBoardingPoint\(\)](#).

Referenced by [stdair::OnDDate::getOrigin\(\)](#).

34.99.3.3 `const AirportCode_T stdair::OnDDateKey::getDestination () const`

Get the destination.

Definition at line 61 of file [OnDDateKey.cpp](#).

References [stdair::BomKeyManager::extractSegmentDateKey\(\)](#), and [stdair::SegmentDateKey::getOffPoint\(\)](#).

Referenced by [stdair::OnDDate::getDestination\(\)](#).

34.99.3.4 `const short stdair::OnDDateKey::getNbOfSegments () const [inline]`

Get the number of segments.

Definition at line 70 of file [OnDDateKey.hpp](#).

Referenced by [stdair::OnDDate::getNbOfSegments\(\)](#).

34.99.3.5 `void stdair::OnDDateKey::toStream (std::ostream & ioOut) const [virtual]`

Dump a Business Object Key into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 68 of file [OnDDateKey.cpp](#).

References [toString\(\)](#).

34.99.3.6 `void stdair::OnDDateKey::fromStream (std::istream & ioIn) [virtual]`

Read a Business Object Key from an input stream.

Parameters

<i>istream&</i> the input stream.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 73 of file [OnDDateKey.cpp](#).

34.99.3.7 `const std::string stdair::OnDDateKey::toString () const [virtual]`

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 77 of file [OnDDateKey.cpp](#).

Referenced by [stdair::OnDDate::describeKey\(\)](#), and [toStream\(\)](#).

34.99.3.8 `template<class Archive > void stdair::OnDDateKey::serialize (Archive & ar, const unsigned int iFileVersion)`

Serialisation.

Definition at line 102 of file [OnDDateKey.cpp](#).

34.99.4 Friends And Related Function Documentation

34.99.4.1 `friend class boost::serialization::access [friend]`

Definition at line 24 of file [OnDDateKey.hpp](#).

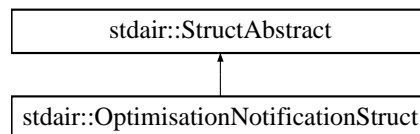
The documentation for this struct was generated from the following files:

- [stdair/bom/OnDDateKey.hpp](#)
- [stdair/bom/OnDDateKey.cpp](#)

34.100 stdair::OptimisationNotificationStruct Struct Reference

```
#include <stdair/bom/OptimisationNotificationStruct.hpp>
```

Inheritance diagram for `stdair::OptimisationNotificationStruct`:



Public Member Functions

- `const AirportCode_T & getOrigin () const`
- `const AirportCode_T & getDestination () const`
- `const CityCode_T & getPOS () const`
- `const Date_T & getPreferedDepartureDate () const`
- `const DateTime_T & getNotificationDateTime () const`
- `const CabinCode_T & getPreferredCabin () const`

- const [NbOfSeats_T](#) & [getPartySize](#) () const
- const [ChannellLabel_T](#) & [getOptimisationChannel](#) () const
- const [TripType_T](#) & [getTripType](#) () const
- const [DayDuration_T](#) & [getStayDuration](#) () const
- const [FrequentFlyer_T](#) & [getFrequentFlyerType](#) () const
- const [Duration_T](#) & [getPreferredDepartureTime](#) () const
- const [WTP_T](#) & [getWTP](#) () const
- const [PriceValue_T](#) & [getValueOfTime](#) () const
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- const std::string [describe](#) () const
- [OptimisationNotificationStruct](#) (const [AirportCode_T](#) &iOrigin, const [AirportCode_T](#) &iDestination, const [CityCode_T](#) &iPOS, const [Date_T](#) &iDepartureDate, const [DateTime_T](#) &iNotificationDateTime, const [CabinCode_T](#) &iPreferredCabin, const [NbOfSeats_T](#) &iPartySize, const [ChannellLabel_T](#) &iChannel, const [TripType_T](#) &iTripType, const [DayDuration_T](#) &iStayDuration, const [FrequentFlyer_T](#) &iFrequentFlyerType, const [Duration_T](#) &iPreferredDepartureTime, const [WTP_T](#) &iWTP, const [PriceValue_T](#) &iValueOfTime)
- [OptimisationNotificationStruct](#) (const [OptimisationNotificationStruct](#) &)
- [~OptimisationNotificationStruct](#) ()

34.100.1 Detailed Description

Structure holding the elements of a optimisation notification.

Definition at line 19 of file [OptimisationNotificationStruct.hpp](#).

34.100.2 Constructor & Destructor Documentation

34.100.2.1 `stdair::OptimisationNotificationStruct::OptimisationNotificationStruct (const AirportCode_T & iOrigin, const AirportCode_T & iDestination, const CityCode_T & iPOS, const Date_T & iDepartureDate, const DateTime_T & iNotificationDateTime, const CabinCode_T & iPreferredCabin, const NbOfSeats_T & iPartySize, const ChannellLabel_T & iChannel, const TripType_T & iTripType, const DayDuration_T & iStayDuration, const FrequentFlyer_T & iFrequentFlyerType, const Duration_T & iPreferredDepartureTime, const WTP_T & iWTP, const PriceValue_T & iValueOfTime)`

Constructor.

Definition at line 39 of file [OptimisationNotificationStruct.cpp](#).

34.100.2.2 `stdair::OptimisationNotificationStruct::OptimisationNotificationStruct (const OptimisationNotificationStruct & iOptimisationNotification)`

Copy constructor.

Definition at line 20 of file [OptimisationNotificationStruct.cpp](#).

34.100.2.3 **stdair::OptimisationNotificationStruct::~~OptimisationNotificationStruct** ()

Destructor.

Definition at line 64 of file [OptimisationNotificationStruct.cpp](#).

34.100.3 Member Function Documentation

34.100.3.1 **const AirportCode_T& stdair::OptimisationNotificationStruct::getOrigin** () const [inline]

Get the notificationed origin.

Definition at line 23 of file [OptimisationNotificationStruct.hpp](#).

34.100.3.2 **const AirportCode_T& stdair::OptimisationNotificationStruct::getDestination** () const [inline]

Get the notificationed destination.

Definition at line 28 of file [OptimisationNotificationStruct.hpp](#).

34.100.3.3 **const CityCode_T& stdair::OptimisationNotificationStruct::getPOS** () const [inline]

Get the point-of-sale.

Definition at line 33 of file [OptimisationNotificationStruct.hpp](#).

34.100.3.4 **const Date_T& stdair::OptimisationNotificationStruct::getPreferredDepartureDate** () const [inline]

Get the notificationed departure date.

Definition at line 38 of file [OptimisationNotificationStruct.hpp](#).

34.100.3.5 **const DateTime_T& stdair::OptimisationNotificationStruct::getNotificationDateTime** () const [inline]

Get the notification datetime.

Definition at line 43 of file [OptimisationNotificationStruct.hpp](#).

34.100.3.6 **const CabinCode_T& stdair::OptimisationNotificationStruct::getPreferredCabin** () const [inline]

Get the preferred cabin.

Definition at line 48 of file [OptimisationNotificationStruct.hpp](#).

34.100.3.7 **const NbOfSeats_T& stdair::OptimisationNotificationStruct::getPartySize** () const [inline]

Get the party size.

Definition at line 53 of file [OptimisationNotificationStruct.hpp](#).

34.100.3.8 **const ChannelLabel_T&** stdair::OptimisationNotificationStruct::getOptimisationChannel () const [\[inline\]](#)

Get the reservation channel.

Definition at line 58 of file [OptimisationNotificationStruct.hpp](#).

34.100.3.9 **const TripType_T&** stdair::OptimisationNotificationStruct::getTripType () const [\[inline\]](#)

Get the trip type.

Definition at line 63 of file [OptimisationNotificationStruct.hpp](#).

34.100.3.10 **const DayDuration_T&** stdair::OptimisationNotificationStruct::getStayDuration () const [\[inline\]](#)

Get the duration of stay.

Definition at line 68 of file [OptimisationNotificationStruct.hpp](#).

34.100.3.11 **const FrequentFlyer_T&** stdair::OptimisationNotificationStruct::getFrequentFlyerType () const [\[inline\]](#)

Get the frequent flyer type.

Definition at line 73 of file [OptimisationNotificationStruct.hpp](#).

34.100.3.12 **const Duration_T&** stdair::OptimisationNotificationStruct::getPreferredDepartureTime () const [\[inline\]](#)

Get the preferred departure time.

Definition at line 78 of file [OptimisationNotificationStruct.hpp](#).

34.100.3.13 **const WTP_T&** stdair::OptimisationNotificationStruct::getWTP () const [\[inline\]](#)

Get the willingness-to-pay.

Definition at line 83 of file [OptimisationNotificationStruct.hpp](#).

34.100.3.14 **const PriceValue_T&** stdair::OptimisationNotificationStruct::getValueOfTime () const [\[inline\]](#)

Get the value of time.

Definition at line 88 of file [OptimisationNotificationStruct.hpp](#).

34.100.3.15 **void** stdair::OptimisationNotificationStruct::toStream (std::ostream & *ioOut*) const

Dump a Business Object into an output stream.

Parameters

<i>ostream</i> & the output stream.

Reimplemented from [stdair::StructAbstract](#).

Definition at line 68 of file [OptimisationNotificationStruct.cpp](#).

References [describe\(\)](#).

34.100.3.16 void stdair::OptimisationNotificationStruct::fromStream (std::istream & *ioIn*)
[virtual]

Read a Business Object from an input stream.

Parameters

<i>istream</i> & the input stream.

Reimplemented from [stdair::StructAbstract](#).

Definition at line 73 of file [OptimisationNotificationStruct.cpp](#).

34.100.3.17 const std::string stdair::OptimisationNotificationStruct::describe () const
[virtual]

Display of the structure.

Implements [stdair::StructAbstract](#).

Definition at line 77 of file [OptimisationNotificationStruct.cpp](#).

Referenced by [toStream\(\)](#).

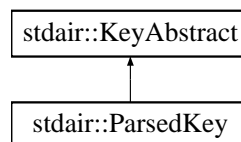
The documentation for this struct was generated from the following files:

- [stdair/bom/OptimisationNotificationStruct.hpp](#)
- [stdair/bom/OptimisationNotificationStruct.cpp](#)

34.101 stdair::ParsedKey Struct Reference

```
#include <stdair/bom/ParsedKey.hpp>
```

Inheritance diagram for stdair::ParsedKey:

**Public Member Functions**

- [InventoryKey](#) getInventoryKey () const

- [FlightDateKey](#) [getFlightDateKey](#) () const
- [SegmentDateKey](#) [getSegmentKey](#) () const
- const [Duration_T](#) [getBoardingTime](#) () const
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- const std::string [toString](#) () const
- [ParsedKey](#) ()
- [~ParsedKey](#) ()

Public Attributes

- std::string [_fullKey](#)
- std::string [_airlineCode](#)
- std::string [_flightNumber](#)
- std::string [_departureDate](#)
- std::string [_boardingPoint](#)
- std::string [_offPoint](#)
- std::string [_boardingTime](#)

34.101.1 Detailed Description

Structure which holds the results/keys after the parsing.

Definition at line 21 of file [ParsedKey.hpp](#).

34.101.2 Constructor & Destructor Documentation

34.101.2.1 stdair::ParsedKey::ParsedKey ()

Definition at line 40 of file [ParsedKey.cpp](#).

34.101.2.2 stdair::ParsedKey::~~ParsedKey ()

Definition at line 46 of file [ParsedKey.cpp](#).

34.101.3 Member Function Documentation

34.101.3.1 InventoryKey stdair::ParsedKey::getInventoryKey () const

[Inventory](#) key.

Definition at line 50 of file [ParsedKey.cpp](#).

References [_airlineCode](#), [_fullKey](#), [STDAIR_LOG_DEBUG](#), [STDAIR_LOG_ERROR](#), and [toString\(\)](#).

Referenced by [stdair::BomKeyManager::extractInventoryKey\(\)](#).

34.101.3.2 FlightDateKey stdair::ParsedKey::getFlightDateKey () const

Flight-date key.

Definition at line 61 of file [ParsedKey.cpp](#).

References [_departureDate](#), [_flightNumber](#), [_fullKey](#), [STDAIR_LOG_DEBUG](#), [STDAIR_LOG_ERROR](#), [stdair::TokeniserDashSeparator\(\)](#), and [toString\(\)](#).

Referenced by [stdair::BomKeyManager::extractFlightDateKey\(\)](#), and [stdair::BomRetriever::retrieveSegmentDateFromLo](#)

34.101.3.3 SegmentDateKey stdair::ParsedKey::getSegmentKey () const

Segment-date key.

Definition at line 83 of file [ParsedKey.cpp](#).

References [_boardingPoint](#), [_fullKey](#), [_offPoint](#), [STDAIR_LOG_DEBUG](#), [STDAIR_LOG_ERROR](#), and [toString\(\)](#).

Referenced by [stdair::BomKeyManager::extractSegmentDateKey\(\)](#), and [stdair::BomRetriever::retrieveSegmentDateFrom](#)

34.101.3.4 const Duration_T stdair::ParsedKey::getBoardingTime () const

Boarding time.

Definition at line 97 of file [ParsedKey.cpp](#).

References [_boardingTime](#), [_fullKey](#), [STDAIR_LOG_DEBUG](#), [STDAIR_LOG_ERROR](#), [stdair::TokeniserTimeSeparator\(\)](#), and [toString\(\)](#).

34.101.3.5 void stdair::ParsedKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 115 of file [ParsedKey.cpp](#).

References [toString\(\)](#).

34.101.3.6 void stdair::ParsedKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters

<i>istream&</i> the input stream.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 120 of file [ParsedKey.cpp](#).

34.101.3.7 const std::string stdair::ParsedKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 124 of file [ParsedKey.cpp](#).

References [_airlineCode](#), [_boardingPoint](#), [_boardingTime](#), [_departureDate](#), [_flightNumber](#), [_offPoint](#), [stdair::DEFAULT_KEY_FLD_DELIMITER](#), and [stdair::DEFAULT_KEY_SUB_FLD_DELIMITER](#).

Referenced by [stdair::TravelSolutionStruct::describe\(\)](#), [stdair::TravelSolutionStruct::display\(\)](#), [getBoardingTime\(\)](#), [getFlightDateKey\(\)](#), [getInventoryKey\(\)](#), [getSegmentKey\(\)](#), and [toString\(\)](#).

34.101.4 Member Data Documentation

34.101.4.1 std::string stdair::ParsedKey::_fullKey

Definition at line 72 of file [ParsedKey.hpp](#).

Referenced by [stdair::BomKeyManager::extractKeys\(\)](#), [getBoardingTime\(\)](#), [getFlightDateKey\(\)](#), [getInventoryKey\(\)](#), and [getSegmentKey\(\)](#).

34.101.4.2 std::string stdair::ParsedKey::_airlineCode

Definition at line 73 of file [ParsedKey.hpp](#).

Referenced by [stdair::BomKeyManager::extractKeys\(\)](#), [getInventoryKey\(\)](#), [stdair::BomRetriever::retrieveSegmentDateFromKey\(\)](#), and [toString\(\)](#).

34.101.4.3 std::string stdair::ParsedKey::_flightNumber

Definition at line 74 of file [ParsedKey.hpp](#).

Referenced by [stdair::BomKeyManager::extractKeys\(\)](#), [getFlightDateKey\(\)](#), and [toString\(\)](#).

34.101.4.4 std::string stdair::ParsedKey::_departureDate

Definition at line 75 of file [ParsedKey.hpp](#).

Referenced by [stdair::BomKeyManager::extractKeys\(\)](#), [getFlightDateKey\(\)](#), and [toString\(\)](#).

34.101.4.5 std::string stdair::ParsedKey::_boardingPoint

Definition at line 76 of file [ParsedKey.hpp](#).

Referenced by [stdair::BomKeyManager::extractKeys\(\)](#), [getSegmentKey\(\)](#), and [toString\(\)](#).

34.101.4.6 std::string stdair::ParsedKey::_offPoint

Definition at line 77 of file [ParsedKey.hpp](#).

Referenced by [stdair::BomKeyManager::extractKeys\(\)](#), [getSegmentKey\(\)](#), and [toString\(\)](#).

34.101.4.7 std::string stdair::ParsedKey::_boardingTime

Definition at line 78 of file [ParsedKey.hpp](#).

Referenced by [stdair::BomKeyManager::extractKeys\(\)](#), [getBoardingTime\(\)](#), and [toString\(\)](#).

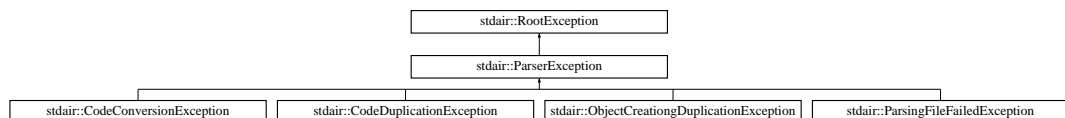
The documentation for this struct was generated from the following files:

- [stdair/bom/ParsedKey.hpp](#)
- [stdair/bom/ParsedKey.cpp](#)

34.102 stdair::ParserException Class Reference

```
#include <stdair/stdair_exceptions.hpp>
```

Inheritance diagram for stdair::ParserException:



Public Member Functions

- [ParserException](#) (const std::string &iWhat)
- const char * [what](#) () const throw ()

Protected Attributes

- std::string [_what](#)

34.102.1 Detailed Description

Parser.

Definition at line 112 of file [stdair_exceptions.hpp](#).

34.102.2 Constructor & Destructor Documentation

34.102.2.1 stdair::ParserException::ParserException (const std::string & iWhat)
[inline]

Constructor.

Definition at line 115 of file [stdair_exceptions.hpp](#).

34.102.3 Member Function Documentation

34.102.3.1 `const char* stdair::RootException::what () const throw ()` `[inline, inherited]`

Give the details of the exception.

Definition at line 38 of file [stdair_exceptions.hpp](#).

References [stdair::RootException::_what](#).

34.102.4 Member Data Documentation

34.102.4.1 `std::string stdair::RootException::_what` `[protected, inherited]`

Details for the exception.

Definition at line 46 of file [stdair_exceptions.hpp](#).

Referenced by [stdair::RootException::what\(\)](#).

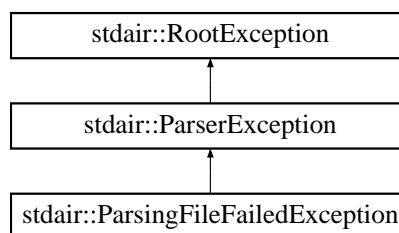
The documentation for this class was generated from the following file:

- [stdair/stdair_exceptions.hpp](#)

34.103 stdair::ParsingFileFailedException Class Reference

```
#include <stdair/stdair_exceptions.hpp>
```

Inheritance diagram for `stdair::ParsingFileFailedException`:



Public Member Functions

- [ParsingFileFailedException](#) (const std::string &iWhat)
- const char * [what](#) () const throw ()

Protected Attributes

- `std::string _what`

34.103.1 Detailed Description

Input file parsing failure.

Definition at line 165 of file [stdair_exceptions.hpp](#).

34.103.2 Constructor & Destructor Documentation

34.103.2.1 `stdair::ParsingFileFailedException::ParsingFileFailedException (const std::string & iWhat) [inline]`

Constructor.

Definition at line 168 of file [stdair_exceptions.hpp](#).

34.103.3 Member Function Documentation

34.103.3.1 `const char* stdair::RootException::what () const throw () [inline, inherited]`

Give the details of the exception.

Definition at line 38 of file [stdair_exceptions.hpp](#).

References [stdair::RootException::_what](#).

34.103.4 Member Data Documentation

34.103.4.1 `std::string stdair::RootException::_what [protected, inherited]`

Details for the exception.

Definition at line 46 of file [stdair_exceptions.hpp](#).

Referenced by [stdair::RootException::what\(\)](#).

The documentation for this class was generated from the following file:

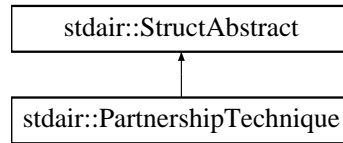
- [stdair/stdair_exceptions.hpp](#)

34.104 stdair::PartnershipTechnique Struct Reference

Enumeration of partnership techniques.

```
#include <stdair/basic/PartnershipTechnique.hpp>
```

Inheritance diagram for stdair::PartnershipTechnique:



Public Types

- enum [EN_PartnershipTechnique](#) {
[NONE](#) = 0, [RAE_DA](#), [RAE_YP](#), [IBP_DA](#),
[IBP_YP](#), [IBP_YP_U](#), [RMC](#), [A_RMC](#),
[LAST_VALUE](#) }

Public Member Functions

- [EN_PartnershipTechnique](#) [getTechnique](#) () const
- char [getTechniqueAsChar](#) () const
- std::string [getTechniqueAsString](#) () const
- const std::string [describe](#) () const
- bool [operator==](#) (const [EN_PartnershipTechnique](#) &) const
- [PartnershipTechnique](#) (const [EN_PartnershipTechnique](#) &)
- [PartnershipTechnique](#) (const char iTechnique)
- [PartnershipTechnique](#) (const std::string &iTechnique)
- [PartnershipTechnique](#) (const [PartnershipTechnique](#) &)
- void [toStream](#) (std::ostream &ioOut) const
- virtual void [fromStream](#) (std::istream &ioIn)

Static Public Member Functions

- static const std::string & [getLabel](#) (const [EN_PartnershipTechnique](#) &)
- static [EN_PartnershipTechnique](#) [getTechnique](#) (const char)
- static char [getTechniqueLabel](#) (const [EN_PartnershipTechnique](#) &)
- static std::string [getTechniqueLabelAsString](#) (const [EN_PartnershipTechnique](#) &)
- static std::string [describeLabels](#) ()

34.104.1 Detailed Description

Enumeration of partnership techniques.

Definition at line 17 of file [PartnershipTechnique.hpp](#).

34.104.2 Member Enumeration Documentation

34.104.2.1 enum stdair::PartnershipTechnique::EN_PartnershipTechnique

Enumerator:

NONE
RAE_DA
RAE_YP
IBP_DA
IBP_YP
IBP_YP_U
RMC
A_RMC
LAST_VALUE

Definition at line 19 of file [PartnershipTechnique.hpp](#).

34.104.3 Constructor & Destructor Documentation

34.104.3.1 stdair::PartnershipTechnique::PartnershipTechnique (const EN_PartnershipTechnique & *iPartnershipTechnique*)

Main constructor.

Definition at line 48 of file [PartnershipTechnique.cpp](#).

34.104.3.2 stdair::PartnershipTechnique::PartnershipTechnique (const char *iTechnique*)

Alternative constructor.

Definition at line 82 of file [PartnershipTechnique.cpp](#).

34.104.3.3 stdair::PartnershipTechnique::PartnershipTechnique (const std::string & *iTechnique*)

Alternative constructor.

Definition at line 88 of file [PartnershipTechnique.cpp](#).

References [getTechnique\(\)](#).

34.104.3.4 stdair::PartnershipTechnique::PartnershipTechnique (const PartnershipTechnique & *iPartnershipTechnique*)

Default copy constructor.

Definition at line 42 of file [PartnershipTechnique.cpp](#).

34.104.4 Member Function Documentation

34.104.4.1 `const std::string & stdair::PartnershipTechnique::getLabel (const EN_PartnershipTechnique & iTechnique) [static]`

Get the label as a string (e.g., "RevenueManagementCooperation").

Definition at line 98 of file [PartnershipTechnique.cpp](#).

34.104.4.2 `PartnershipTechnique::EN_PartnershipTechnique stdair::PartnershipTechnique::getTechnique (const char iTechniqueChar) [static]`

Get the technique value from parsing a single char (e.g., 'r' or 'C').

Definition at line 54 of file [PartnershipTechnique.cpp](#).

References [A_RMC](#), [describeLabels\(\)](#), [IBP_DA](#), [IBP_YP](#), [IBP_YP_U](#), [LAST_VALUE](#), [NONE](#), [RAE_DA](#), [RAE_YP](#), and [RMC](#).

34.104.4.3 `char stdair::PartnershipTechnique::getTechniqueLabel (const EN_PartnershipTechnique & iTechnique) [static]`

Get the label as a single char (e.g., 'r' or 'C').

Definition at line 104 of file [PartnershipTechnique.cpp](#).

34.104.4.4 `std::string stdair::PartnershipTechnique::getTechniqueLabelAsString (const EN_PartnershipTechnique & iTechnique) [static]`

Get the label as a string of a single char (e.g., "r" or "C").

Definition at line 110 of file [PartnershipTechnique.cpp](#).

34.104.4.5 `std::string stdair::PartnershipTechnique::describeLabels () [static]`

List the labels.

Definition at line 117 of file [PartnershipTechnique.cpp](#).

References [LAST_VALUE](#).

Referenced by [getTechnique\(\)](#).

34.104.4.6 `PartnershipTechnique::EN_PartnershipTechnique stdair::PartnershipTechnique::getTechnique () const`

Get the enumerated value.

Definition at line 130 of file [PartnershipTechnique.cpp](#).

Referenced by [PartnershipTechnique\(\)](#).

34.104.4.7 `char stdair::PartnershipTechnique::getTechniqueAsChar () const`

Get the enumerated value as a short string (e.g., 'r' or 'C').

Definition at line 135 of file [PartnershipTechnique.cpp](#).

34.104.4.8 `std::string stdair::PartnershipTechnique::getTechniqueAsString () const`

Get the enumerated value as a short string (e.g., "r" or "C").

Definition at line 141 of file [PartnershipTechnique.cpp](#).

34.104.4.9 `const std::string stdair::PartnershipTechnique::describe () const` `[virtual]`

Give a description of the structure (e.g., "RevenueManagementCooperation" or "Inter-lineBidPriceYieldProration").

Implements [stdair::StructAbstract](#).

Definition at line 148 of file [PartnershipTechnique.cpp](#).

34.104.4.10 `bool stdair::PartnershipTechnique::operator== (const EN_PartnershipTechnique & iTechnique) const`

Comparison operator.

Definition at line 156 of file [PartnershipTechnique.cpp](#).

34.104.4.11 `void stdair::StructAbstract::toStream (std::ostream & ioOut) const` `[inline, inherited]`

Dump a Business Object into an output stream.

Parameters

<code>ostream&</code> the output stream.
--

Reimplemented in [stdair::YieldRange](#), [stdair::AirlineStruct](#), [stdair::BookingRequestStruct](#), [stdair::CancellationStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::TravelSolutionStruct](#), and [stdair::VirtualClassStruct](#).

Definition at line 29 of file [StructAbstract.hpp](#).

References [stdair::StructAbstract::describe\(\)](#).

Referenced by [operator<<\(\)](#).

34.104.4.12 `virtual void stdair::StructAbstract::fromStream (std::istream & ioIn)` `[inline, virtual, inherited]`

Read a Business Object from an input stream.

Parameters

<code>istream&</code> the input stream.

Reimplemented in [stdair::ProgressStatusSet](#), [stdair::YieldRange](#), [stdair::AirlineStruct](#), [stdair::BookingRequestStruct](#), [stdair::CancellationStruct](#), [stdair::EventStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::TravelSolutionStruct](#),

and [stdair::VirtualClassStruct](#).

Definition at line 38 of file [StructAbstract.hpp](#).

Referenced by [operator>>\(\)](#).

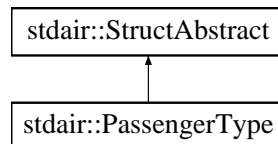
The documentation for this struct was generated from the following files:

- [stdair/basic/PartnershipTechnique.hpp](#)
- [stdair/basic/PartnershipTechnique.cpp](#)

34.105 stdair::PassengerType Struct Reference

```
#include <stdair/basic/PassengerType.hpp>
```

Inheritance diagram for `stdair::PassengerType`:



Public Types

- enum [EN_PassengerType](#) { [LEISURE](#) = 0, [BUSINESS](#), [FIRST](#), [LAST_VALUE](#) }

Public Member Functions

- [EN_PassengerType](#) [getType](#) () const
- `std::string` [getTypeAsString](#) () const
- `const std::string` [describe](#) () const
- `bool` [operator==](#) (const [EN_PassengerType](#) &) const
- [PassengerType](#) (const [EN_PassengerType](#) &)
- [PassengerType](#) (const char iType)
- `void` [toStream](#) (std::ostream &ioOut) const
- `virtual void` [fromStream](#) (std::istream &ioIn)

Static Public Member Functions

- `static const std::string` & [getLabel](#) (const [EN_PassengerType](#) &)
- `static char` [getTypeLabel](#) (const [EN_PassengerType](#) &)
- `static std::string` [getTypeLabelAsString](#) (const [EN_PassengerType](#) &)
- `static std::string` [describeLabels](#) ()

34.105.1 Detailed Description

Enumeration of Frequent Flyer types.

Definition at line 15 of file [PassengerType.hpp](#).

34.105.2 Member Enumeration Documentation

34.105.2.1 enum stdair::PassengerType::EN_PassengerType

Enumerator:

LEISURE

BUSINESS

FIRST

LAST_VALUE

Definition at line 17 of file [PassengerType.hpp](#).

34.105.3 Constructor & Destructor Documentation

34.105.3.1 stdair::PassengerType::PassengerType (const EN_PassengerType & iPassengerType)

Constructor.

Definition at line 21 of file [PassengerType.cpp](#).

34.105.3.2 stdair::PassengerType::PassengerType (const char iType)

Constructor.

Definition at line 26 of file [PassengerType.cpp](#).

References [BUSINESS](#), [describeLabels\(\)](#), [FIRST](#), [LAST_VALUE](#), and [LEISURE](#).

34.105.4 Member Function Documentation

34.105.4.1 const std::string & stdair::PassengerType::getLabel (const EN_PassengerType & iType) [static]

Get the label as a string (e.g., "Leisure" or "Business").

Definition at line 44 of file [PassengerType.cpp](#).

34.105.4.2 char stdair::PassengerType::getTypeLabel (const EN_PassengerType & iType) [static]

Get the label as a single char (e.g., 'L' or 'B').

Definition at line 49 of file [PassengerType.cpp](#).

34.105.4.3 `std::string stdair::PassengerType::getTypeLabelAsString (const EN_PassengerType & iType) [static]`

Get the label as a single char (e.g., 'L' or 'B').

Definition at line 55 of file [PassengerType.cpp](#).

34.105.4.4 `std::string stdair::PassengerType::describeLabels () [static]`

List the labels.

Definition at line 62 of file [PassengerType.cpp](#).

References [LAST_VALUE](#).

Referenced by [PassengerType\(\)](#).

34.105.4.5 `PassengerType::EN_PassengerType stdair::PassengerType::getType () const`

Get the enumerated value.

Definition at line 74 of file [PassengerType.cpp](#).

34.105.4.6 `std::string stdair::PassengerType::getTypeAsString () const`

Get the enumerated value as a short string (e.g., 'L' or 'B').

Definition at line 79 of file [PassengerType.cpp](#).

34.105.4.7 `const std::string stdair::PassengerType::describe () const [virtual]`

Give a description of the structure (e.g., "Leisure" or "Business").

Implements [stdair::StructAbstract](#).

Definition at line 86 of file [PassengerType.cpp](#).

34.105.4.8 `bool stdair::PassengerType::operator== (const EN_PassengerType & iType) const`

Comparison operator.

Definition at line 93 of file [PassengerType.cpp](#).

34.105.4.9 `void stdair::StructAbstract::toStream (std::ostream & ioOut) const [inline, inherited]`

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Reimplemented in [stdair::YieldRange](#), [stdair::AirlineStruct](#), [stdair::BookingRequestStruct](#), [stdair::CancellationStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::TravelSolutionStruct](#), and [stdair::VirtualClassStruct](#).

Definition at line 29 of file [StructAbstract.hpp](#).

References [stdair::StructAbstract::describe\(\)](#).

Referenced by [operator<<\(\)](#).

34.105.4.10 virtual void stdair::StructAbstract::fromStream (std::istream & *ioIn*)
[inline, virtual, inherited]

Read a Business Object from an input stream.

Parameters

<i>istream&</i> the input stream.

Reimplemented in [stdair::ProgressStatusSet](#), [stdair::YieldRange](#), [stdair::AirlineStruct](#), [stdair::BookingRequestStruct](#), [stdair::CancellationStruct](#), [stdair::EventStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::TravelSolutionStruct](#), and [stdair::VirtualClassStruct](#).

Definition at line 38 of file [StructAbstract.hpp](#).

Referenced by [operator>>\(\)](#).

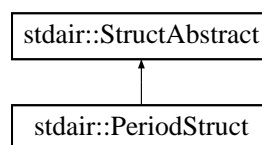
The documentation for this struct was generated from the following files:

- [stdair/basic/PassengerType.hpp](#)
- [stdair/basic/PassengerType.cpp](#)

34.106 stdair::PeriodStruct Struct Reference

```
#include <stdair/bom/PeriodStruct.hpp>
```

Inheritance diagram for stdair::PeriodStruct:



Public Member Functions

- const [DatePeriod_T](#) & [getDateRange](#) () const
- const [DoWStruct](#) & [getDoW](#) () const
- void [setDateRange](#) (const [DatePeriod_T](#) & iDateRange)
- void [setDoW](#) (const [DoWStruct](#) & iDoW)
- const std::string [describe](#) () const
- const std::string [describeShort](#) () const
- [PeriodStruct](#) [addDateOffset](#) (const [DateOffset_T](#) &) const

- [PeriodStruct intersection](#) (const [PeriodStruct](#) &) const
- const bool [isValid](#) () const
- [PeriodStruct](#) (const [DatePeriod_T](#) &, const [DoWStruct](#) &)
- [PeriodStruct](#) ()
- [PeriodStruct](#) (const [PeriodStruct](#) &)
- [~PeriodStruct](#) ()
- void [toStream](#) (std::ostream &ioOut) const
- virtual void [fromStream](#) (std::istream &ioIn)

34.106.1 Detailed Description

Define a departure period

A period is defined by a date range and a day-of-week struct.

Definition at line 19 of file [PeriodStruct.hpp](#).

34.106.2 Constructor & Destructor Documentation

34.106.2.1 `stdair::PeriodStruct::PeriodStruct (const DatePeriod_T & iDateRange, const DoWStruct & iDoW)`

Constructor.

Definition at line 19 of file [PeriodStruct.cpp](#).

34.106.2.2 `stdair::PeriodStruct::PeriodStruct ()`

Default constructors.

Definition at line 14 of file [PeriodStruct.cpp](#).

Referenced by [addDateOffset\(\)](#), and [intersection\(\)](#).

34.106.2.3 `stdair::PeriodStruct::PeriodStruct (const PeriodStruct & iPeriodStruct)`

Definition at line 25 of file [PeriodStruct.cpp](#).

34.106.2.4 `stdair::PeriodStruct::~~PeriodStruct () [inline]`

Default destructor.

Definition at line 64 of file [PeriodStruct.hpp](#).

34.106.3 Member Function Documentation

34.106.3.1 `const DatePeriod_T& stdair::PeriodStruct::getDateRange () const [inline]`

Retrieve the attributes.

Definition at line 23 of file [PeriodStruct.hpp](#).

Referenced by [addDateOffset\(\)](#).

34.106.3.2 `const DoWStruct& stdair::PeriodStruct::getDoW () const` `[inline]`

Definition at line 26 of file [PeriodStruct.hpp](#).

Referenced by [addDateOffset\(\)](#).

34.106.3.3 `void stdair::PeriodStruct::setDateRange (const DatePeriod_T & iDateRange)`
`[inline]`

Set the new value for the attributes.

Definition at line 33 of file [PeriodStruct.hpp](#).

34.106.3.4 `void stdair::PeriodStruct::setDoW (const DoWStruct & iDoW)` `[inline]`

Definition at line 36 of file [PeriodStruct.hpp](#).

34.106.3.5 `const std::string stdair::PeriodStruct::describe () const` `[virtual]`

Display explicitly (e.g., "Mon.Tue.Wed.Thu.Fri.").

Implements [stdair::StructAbstract](#).

Definition at line 38 of file [PeriodStruct.cpp](#).

References [stdair::DoWStruct::describe\(\)](#).

34.106.3.6 `const std::string stdair::PeriodStruct::describeShort () const`

Display as a bit set (e.g., "1111100").

Definition at line 31 of file [PeriodStruct.cpp](#).

References [stdair::DoWStruct::describeShort\(\)](#).

Referenced by [stdair::FlightPeriodKey::toString\(\)](#).

34.106.3.7 `PeriodStruct stdair::PeriodStruct::addDateOffset (const DateOffset_T & iDateOffset) const`

Build a period struct from this period struct by adding a date offset.

Definition at line 46 of file [PeriodStruct.cpp](#).

References [getDateRange\(\)](#), [getDoW\(\)](#), [PeriodStruct\(\)](#), and [stdair::DoWStruct::shift\(\)](#).

34.106.3.8 `PeriodStruct stdair::PeriodStruct::intersection (const PeriodStruct & iPeriodStruct) const`

Build a new period struct which is the intersection of two period structs.

Definition at line 63 of file [PeriodStruct.cpp](#).

References [stdair::DoWStruct::intersection\(\)](#), and [PeriodStruct\(\)](#).

34.106.3.9 const bool stdair::PeriodStruct::isValid () const

Return if the period is valid (i.e., valid date range and valid DoW).

Definition at line 72 of file [PeriodStruct.cpp](#).

References [stdair::DoWStruct::isValid\(\)](#).

34.106.3.10 void stdair::StructAbstract::toStream (std::ostream & ioOut) const [inline, inherited]

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Reimplemented in [stdair::YieldRange](#), [stdair::AirlineStruct](#), [stdair::BookingRequestStruct](#), [stdair::CancellationStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::TravelSolutionStruct](#), and [stdair::VirtualClassStruct](#).

Definition at line 29 of file [StructAbstract.hpp](#).

References [stdair::StructAbstract::describe\(\)](#).

Referenced by [operator<<\(\)](#).

34.106.3.11 virtual void stdair::StructAbstract::fromStream (std::istream & ioIn) [inline, virtual, inherited]

Read a Business Object from an input stream.

Parameters

<i>istream&</i> the input stream.

Reimplemented in [stdair::ProgressStatusSet](#), [stdair::YieldRange](#), [stdair::AirlineStruct](#), [stdair::BookingRequestStruct](#), [stdair::CancellationStruct](#), [stdair::EventStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::TravelSolutionStruct](#), and [stdair::VirtualClassStruct](#).

Definition at line 38 of file [StructAbstract.hpp](#).

Referenced by [operator>>\(\)](#).

The documentation for this struct was generated from the following files:

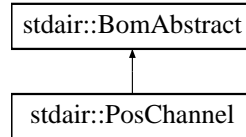
- [stdair/bom/PeriodStruct.hpp](#)
- [stdair/bom/PeriodStruct.cpp](#)

34.107 stdair::PosChannel Class Reference

Class representing the actual attributes for a fare point of sale.

```
#include <stdair/bom/PosChannel.hpp>
```

Inheritance diagram for stdair::PosChannel:



Public Types

- typedef [PosChannelKey](#) [Key_T](#)

Public Member Functions

- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- std::string [toString](#) () const
- const std::string [describeKey](#) () const
- const [Key_T](#) & [getKey](#) () const
- [BomAbstract](#) *const [getParent](#) () const
- const [stdair::HolderMap_T](#) & [getHolderMap](#) () const
- const [CityCode_T](#) & [getPos](#) () const
- const [ChannelLabel_T](#) & [getChannel](#) () const

Protected Member Functions

- [PosChannel](#) (const [Key_T](#) &)
- virtual [~PosChannel](#) ()

Protected Attributes

- [Key_T _key](#)
- [BomAbstract](#) * [_parent](#)
- [HolderMap_T _holderMap](#)

Friends

- class [FacBom](#)
- class [FacBomManager](#)

34.107.1 Detailed Description

Class representing the actual attributes for a fare point of sale.

Definition at line 19 of file [PosChannel.hpp](#).

34.107.2 Member Typedef Documentation

34.107.2.1 typedef PosChannelKey stdair::PosChannel::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 28 of file [PosChannel.hpp](#).

34.107.3 Constructor & Destructor Documentation

34.107.3.1 stdair::PosChannel::PosChannel (const Key_T & iKey) [protected]

Main constructor.

Definition at line 29 of file [PosChannel.cpp](#).

34.107.3.2 stdair::PosChannel::~~PosChannel () [protected, virtual]

Destructor.

Definition at line 34 of file [PosChannel.cpp](#).

34.107.4 Member Function Documentation

34.107.4.1 void stdair::PosChannel::toStream (std::ostream & ioOut) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Implements [stdair::BomAbstract](#).

Definition at line 37 of file [PosChannel.hpp](#).

References [toString\(\)](#).

34.107.4.2 void stdair::PosChannel::fromStream (std::istream & ioin) [inline, virtual]

Read a Business Object from an input stream.

Parameters

<i>istream&</i> the input stream.

Implements [stdair::BomAbstract](#).

Definition at line 46 of file [PosChannel.hpp](#).

34.107.4.3 std::string stdair::PosChannel::toString () const [virtual]

Get the serialised version of the Business Object.

Implements [stdair::BomAbstract](#).

Definition at line 38 of file [PosChannel.cpp](#).

References [describeKey\(\)](#).

Referenced by [toStream\(\)](#).

34.107.4.4 const std::string stdair::PosChannel::describeKey () const [inline]

Get a string describing the key.

Definition at line 57 of file [PosChannel.hpp](#).

References [_key](#), and [stdair::PosChannelKey::toString\(\)](#).

Referenced by [toString\(\)](#).

34.107.4.5 const Key_T& stdair::PosChannel::getKey () const [inline]

Get the primary key (pos, channel).

Definition at line 66 of file [PosChannel.hpp](#).

References [_key](#).

34.107.4.6 BomAbstract* const stdair::PosChannel::getParent () const [inline]

Get a reference on the parent object instance.

Definition at line 73 of file [PosChannel.hpp](#).

References [_parent](#).

34.107.4.7 const stdair::HolderMap_T& stdair::PosChannel::getHolderMap () const [inline]

Get a reference on the children holder.

Definition at line 80 of file [PosChannel.hpp](#).

References [_holderMap](#).

34.107.4.8 const CityCode_T& stdair::PosChannel::getPos () const [inline]

Get the point-of-sale.

Definition at line 87 of file [PosChannel.hpp](#).

References [_key](#), and [stdair::PosChannelKey::getPos\(\)](#).

34.107.4.9 const ChannelLabel_T& stdair::PosChannel::getChannel () const [inline]

Get the channel.

Definition at line 94 of file [PosChannel.hpp](#).

References [_key](#), and [stdair::PosChannelKey::getChannel\(\)](#).

34.107.5 Friends And Related Function Documentation

34.107.5.1 friend class **FacBom** `[friend]`

Definition at line 20 of file [PosChannel.hpp](#).

34.107.5.2 friend class **FacBomManager** `[friend]`

Definition at line 21 of file [PosChannel.hpp](#).

34.107.6 Member Data Documentation

34.107.6.1 **Key_T** **stdair::PosChannel::_key** `[protected]`

Primary key (flight number and departure date).

Definition at line 126 of file [PosChannel.hpp](#).

Referenced by [describeKey\(\)](#), [getChannel\(\)](#), [getKey\(\)](#), and [getPos\(\)](#).

34.107.6.2 **BomAbstract*** **stdair::PosChannel::_parent** `[protected]`

Pointer on the parent class.

Definition at line 131 of file [PosChannel.hpp](#).

Referenced by [getParent\(\)](#).

34.107.6.3 **HolderMap_T** **stdair::PosChannel::_holderMap** `[protected]`

Map holding the children.

Definition at line 136 of file [PosChannel.hpp](#).

Referenced by [getHolderMap\(\)](#).

The documentation for this class was generated from the following files:

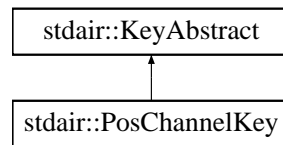
- [stdair/bom/PosChannel.hpp](#)
- [stdair/bom/PosChannel.cpp](#)

34.108 stdair::PosChannelKey Struct Reference

Key of point of sale and channel.

```
#include <stdair/bom/PosChannelKey.hpp>
```

Inheritance diagram for `stdair::PosChannelKey`:



Public Member Functions

- [PosChannelKey](#) (const [stdair::CityCode_T](#) &, const [stdair::ChannelLabel_T](#) &)
- [PosChannelKey](#) (const [PosChannelKey](#) &)
- [~PosChannelKey](#) ()
- const [stdair::CityCode_T](#) & [getPos](#) () const
- const [stdair::ChannelLabel_T](#) & [getChannel](#) () const
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- const std::string [toString](#) () const

34.108.1 Detailed Description

Key of point of sale and channel.

Definition at line 15 of file [PosChannelKey.hpp](#).

34.108.2 Constructor & Destructor Documentation

34.108.2.1 stdair::PosChannelKey::PosChannelKey (const stdair::CityCode_T & iPos, const stdair::ChannelLabel_T & iChannel)

Main constructor.

Definition at line 22 of file [PosChannelKey.cpp](#).

34.108.2.2 stdair::PosChannelKey::PosChannelKey (const PosChannelKey & iKey)

Copy constructor.

Definition at line 28 of file [PosChannelKey.cpp](#).

34.108.2.3 stdair::PosChannelKey::~~PosChannelKey ()

Destructor.

Definition at line 33 of file [PosChannelKey.cpp](#).

34.108.3 Member Function Documentation

34.108.3.1 `const stdair::CityCode_T& stdair::PosChannelKey::getPos () const`
[inline]

Get the point of sale.

Definition at line 43 of file [PosChannelKey.hpp](#).

Referenced by [stdair::PosChannel::getPos\(\)](#).

34.108.3.2 `const stdair::ChannelLabel_T& stdair::PosChannelKey::getChannel () const`
[inline]

Get the channel.

Definition at line 50 of file [PosChannelKey.hpp](#).

Referenced by [stdair::PosChannel::getChannel\(\)](#).

34.108.3.3 `void stdair::PosChannelKey::toStream (std::ostream & ioOut) const`
[virtual]

Dump a Business Object Key into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 37 of file [PosChannelKey.cpp](#).

References [toString\(\)](#).

34.108.3.4 `void stdair::PosChannelKey::fromStream (std::istream & ioIn)` [virtual]

Read a Business Object Key from an input stream.

Parameters

<i>istream&</i> the input stream.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 42 of file [PosChannelKey.cpp](#).

34.108.3.5 `const std::string stdair::PosChannelKey::toString () const` [virtual]

Get the serialised version of the Business Object Key. That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 46 of file [PosChannelKey.cpp](#).

References [stdair::DEFAULT_KEY_SUB_FLD_DELIMITER](#).

Referenced by [stdair::PosChannel::describeKey\(\)](#), and [toStream\(\)](#).

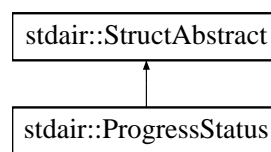
The documentation for this struct was generated from the following files:

- stdair/bom/[PosChannelKey.hpp](#)
- stdair/bom/[PosChannelKey.cpp](#)

34.109 stdair::ProgressStatus Struct Reference

```
#include <stdair/basic/ProgressStatus.hpp>
```

Inheritance diagram for stdair::ProgressStatus:



Public Member Functions

- const [Count_T](#) & [count](#) () const
- const [Count_T](#) & [getCurrentNb](#) () const
- const [Count_T](#) & [getExpectedNb](#) () const
- const [Count_T](#) & [getActualNb](#) () const
- const [ProgressPercentage_T](#) [progress](#) () const
- void [setCurrentNb](#) (const [Count_T](#) &iCurrentNb)
- void [setExpectedNb](#) (const [Count_T](#) &iExpectedNb)
- void [setActualNb](#) (const [Count_T](#) &iActualNb)
- void [reset](#) ()
- [Count_T](#) operator+= ([Count_T](#) iIncrement)
- [Count_T](#) operator++ ()
- const std::string [describe](#) () const
- [ProgressStatus](#) (const [Count_T](#) &iCurrentNb, const [Count_T](#) &iExpectedNb, const [Count_T](#) &iActualNb)
- [ProgressStatus](#) (const [Count_T](#) &iExpectedNb, const [Count_T](#) &iActualNb)
- [ProgressStatus](#) (const [Count_T](#) &iActualNb)
- [ProgressStatus](#) ()
- [ProgressStatus](#) (const [ProgressStatus](#) &)
- void [toStream](#) (std::ostream &ioOut) const
- virtual void [fromStream](#) (std::istream &ioIn)

34.109.1 Detailed Description

Structure holding the details of a progress status.

The progress status is given by the ratio between the "current" and the "expected" (or "actual") numbers. For instance, when the expected/actual number is 1000 and the current number is 200, then the progress status is 20% (= 200 / 1000).

Definition at line 25 of file [ProgressStatus.hpp](#).

34.109.2 Constructor & Destructor Documentation

34.109.2.1 `stdair::ProgressStatus::ProgressStatus (const Count_T & iCurrentNb, const Count_T & iExpectedNb, const Count_T & iActualNb)`

Constructor.

Parameters

<i>const</i>	Count_T& The current number.
<i>const</i>	Count_T& The expected number.
<i>const</i>	Count_T& The actual number.

Definition at line 15 of file [ProgressStatus.cpp](#).

34.109.2.2 `stdair::ProgressStatus::ProgressStatus (const Count_T & iExpectedNb, const Count_T & iActualNb)`

Constructor.

As no current number is given, it is set to 0.

Parameters

<i>const</i>	Count_T& The expected number.
<i>const</i>	Count_T& The actual number.

Definition at line 23 of file [ProgressStatus.cpp](#).

34.109.2.3 `stdair::ProgressStatus::ProgressStatus (const Count_T & iActualNb)`

Constructor.

As no expected number is given, it is assumed to be equal to the actual one. The current number is set to 0.

Parameters

<i>const</i>	Count_T& The actual number.
--------------	-----------------------------

Definition at line 30 of file [ProgressStatus.cpp](#).

34.109.2.4 `stdair::ProgressStatus::ProgressStatus ()`

Constructor.

All the numbers are set to 0.

Definition at line 36 of file [ProgressStatus.cpp](#).

34.109.2.5 `stdair::ProgressStatus::ProgressStatus (const ProgressStatus & iProgressStatus)`

Copy Constructor.

Definition at line 43 of file [ProgressStatus.cpp](#).

34.109.3 Member Function Documentation

34.109.3.1 `const Count_T& stdair::ProgressStatus::count () const [inline]`

Get the current number.

Definition at line 29 of file [ProgressStatus.hpp](#).

34.109.3.2 `const Count_T& stdair::ProgressStatus::getCurrentNb () const [inline]`

Get the current number.

Definition at line 34 of file [ProgressStatus.hpp](#).

Referenced by [stdair::ProgressStatusSet::describe\(\)](#), [stdair::EventQueue::getCurrentNbOfEvents\(\)](#), [stdair::EventQueue::toString\(\)](#), and [stdair::EventQueue::updateStatus\(\)](#).

34.109.3.3 `const Count_T& stdair::ProgressStatus::getExpectedNb () const [inline]`

Get the expected number.

Definition at line 39 of file [ProgressStatus.hpp](#).

Referenced by [stdair::EventQueue::addStatus\(\)](#), [stdair::ProgressStatusSet::describe\(\)](#), [stdair::EventQueue::getExpectedTotalNbOfEvents\(\)](#), [stdair::EventQueue::toString\(\)](#), and [stdair::EventQueue::updateStatus\(\)](#).

34.109.3.4 `const Count_T& stdair::ProgressStatus::getActualNb () const [inline]`

Get the actual number.

Definition at line 44 of file [ProgressStatus.hpp](#).

Referenced by [stdair::EventQueue::addStatus\(\)](#), [stdair::ProgressStatusSet::describe\(\)](#), [stdair::EventQueue::getActualTotalNbOfEvents\(\)](#), [stdair::EventQueue::toString\(\)](#), and [stdair::EventQueue::updateStatus\(\)](#).

34.109.3.5 `const ProgressPercentage_T stdair::ProgressStatus::progress () const [inline]`

Get the progress as a percentage.

Definition at line 49 of file [ProgressStatus.hpp](#).

Referenced by [stdair::EventQueue::calculateProgress\(\)](#).

34.109.3.6 `void stdair::ProgressStatus::setCurrentNb (const Count_T & iCurrentNb) [inline]`

Set the current number.

Definition at line 60 of file [ProgressStatus.hpp](#).

Referenced by [stdair::EventQueue::setCurrentNbOfEvents\(\)](#), [stdair::EventQueue::setStatus\(\)](#), and [stdair::EventQueue::updateStatus\(\)](#).

34.109.3.7 void stdair::ProgressStatus::setExpectedNb (const Count_T & *iExpectedNb*)
[inline]

Set the expected number.

Definition at line 65 of file [ProgressStatus.hpp](#).

Referenced by [stdair::EventQueue::addStatus\(\)](#), [stdair::EventQueue::setExpectedTotalNbOfEvents\(\)](#), [stdair::EventQueue::setStatus\(\)](#), and [stdair::EventQueue::updateStatus\(\)](#).

34.109.3.8 void stdair::ProgressStatus::setActualNb (const Count_T & *iActualNb*)
[inline]

Set the actual number.

Definition at line 70 of file [ProgressStatus.hpp](#).

Referenced by [stdair::EventQueue::addStatus\(\)](#), [stdair::EventQueue::setActualTotalNbOfEvents\(\)](#), [stdair::EventQueue::setStatus\(\)](#), and [stdair::EventQueue::updateStatus\(\)](#).

34.109.3.9 void stdair::ProgressStatus::reset ()

Reset the current number (to 0).

Definition at line 50 of file [ProgressStatus.cpp](#).

References [stdair::DEFAULT_PROGRESS_STATUS](#).

Referenced by [stdair::EventQueue::reset\(\)](#).

34.109.3.10 Count_T stdair::ProgressStatus::operator+= (Count_T *iIncrement*)
[inline]

Increment the current number.

Definition at line 78 of file [ProgressStatus.hpp](#).

34.109.3.11 Count_T stdair::ProgressStatus::operator++ () [inline]

Increment the current number.

Definition at line 84 of file [ProgressStatus.hpp](#).

34.109.3.12 const std::string stdair::ProgressStatus::describe () const [virtual]

Give a description of the structure (e.g., "4 / 100 / 101").

Implements [stdair::StructAbstract](#).

Definition at line 56 of file [ProgressStatus.cpp](#).

34.109.3.13 `void stdair::StructAbstract::toStream (std::ostream & ioOut) const` [`inline`, `inherited`]

Dump a Business Object into an output stream.

Parameters

<code>ostream&</code> the output stream.
--

Reimplemented in [stdair::YieldRange](#), [stdair::AirlineStruct](#), [stdair::BookingRequestStruct](#), [stdair::CancellationStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::TravelSolutionStruct](#), and [stdair::VirtualClassStruct](#).

Definition at line 29 of file [StructAbstract.hpp](#).

References [stdair::StructAbstract::describe\(\)](#).

Referenced by [operator<<\(\)](#).

34.109.3.14 `virtual void stdair::StructAbstract::fromStream (std::istream & ioln)` [`inline`, `virtual`, `inherited`]

Read a Business Object from an input stream.

Parameters

<code>istream&</code> the input stream.

Reimplemented in [stdair::ProgressStatusSet](#), [stdair::YieldRange](#), [stdair::AirlineStruct](#), [stdair::BookingRequestStruct](#), [stdair::CancellationStruct](#), [stdair::EventStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::TravelSolutionStruct](#), and [stdair::VirtualClassStruct](#).

Definition at line 38 of file [StructAbstract.hpp](#).

Referenced by [operator>>\(\)](#).

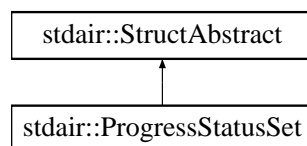
The documentation for this struct was generated from the following files:

- [stdair/basic/ProgressStatus.hpp](#)
- [stdair/basic/ProgressStatus.cpp](#)

34.110 stdair::ProgressStatusSet Struct Reference

```
#include <stdair/basic/ProgressStatusSet.hpp>
```

Inheritance diagram for `stdair::ProgressStatusSet`:



Public Member Functions

- const [ProgressStatus](#) & [getTypeSpecificStatus](#) () const
- const [ProgressStatus](#) & [getSpecificGeneratorStatus](#) () const
- const [ProgressStatus](#) & [getOverallStatus](#) () const
- void [setTypeSpecificStatus](#) (const [ProgressStatus](#) &iProgressStatus)
- void [setSpecificGeneratorStatus](#) (const [ProgressStatus](#) &iProgressStatus, const [EventGeneratorKey_T](#) &iKey)
- void [setOverallStatus](#) (const [ProgressStatus](#) &iProgressStatus)
- void [fromStream](#) (std::istream &iIn)
- const std::string [describe](#) () const
- [ProgressStatusSet](#) (const [EventType::EN_EventType](#) &)
- [ProgressStatusSet](#) (const [ProgressStatusSet](#) &)
- [~ProgressStatusSet](#) ()
- void [toStream](#) (std::ostream &iOut) const

34.110.1 Detailed Description

Structure holding a set of progress status.

Definition at line 22 of file [ProgressStatusSet.hpp](#).

34.110.2 Constructor & Destructor Documentation**34.110.2.1 stdair::ProgressStatusSet::ProgressStatusSet (const [EventType::EN_EventType](#) & iType)**

Constructor .

Definition at line 20 of file [ProgressStatusSet.cpp](#).

34.110.2.2 stdair::ProgressStatusSet::ProgressStatusSet (const [ProgressStatusSet](#) & iProgressStatusSet)

Copy constructor.

Definition at line 27 of file [ProgressStatusSet.cpp](#).

34.110.2.3 stdair::ProgressStatusSet::~~ProgressStatusSet ()

Destructor.

Definition at line 36 of file [ProgressStatusSet.cpp](#).

34.110.3 Member Function Documentation**34.110.3.1 const [ProgressStatus](#)& stdair::ProgressStatusSet::getTypeSpecificStatus () const [inline]**

Get the progress status specific to that event type.

Note that that progress status may not be up-to-date. That attribute is up-to-date only after a call to the [EventQueue::popEvent\(\)](#) method.

Definition at line 31 of file [ProgressStatusSet.hpp](#).

34.110.3.2 `const ProgressStatus& stdair::ProgressStatusSet::getSpecificGeneratorStatus () const [inline]`

Get the progress status specific to the content key for that event.

Note that that progress status may not be up-to-date. That attribute is up-to-date only after a call to the [EventQueue::popEvent\(\)](#) method.

Definition at line 43 of file [ProgressStatusSet.hpp](#).

34.110.3.3 `const ProgressStatus& stdair::ProgressStatusSet::getOverallStatus () const [inline]`

Get the overall progress status (absolute, for all the events).

Note that that progress status may not be up-to-date. That attribute is up-to-date only after a call to the [EventQueue::popEvent\(\)](#) method.

Definition at line 54 of file [ProgressStatusSet.hpp](#).

34.110.3.4 `void stdair::ProgressStatusSet::setTypeSpecificStatus (const ProgressStatus & iProgressStatus) [inline]`

Set/update the progress status specific to that event type.

Definition at line 62 of file [ProgressStatusSet.hpp](#).

Referenced by [stdair::EventQueue::popEvent\(\)](#).

34.110.3.5 `void stdair::ProgressStatusSet::setSpecificGeneratorStatus (const ProgressStatus & iProgressStatus, const EventGeneratorKey_T & iKey) [inline]`

Set/update the progress status specific to the content key for that event.

Definition at line 68 of file [ProgressStatusSet.hpp](#).

34.110.3.6 `void stdair::ProgressStatusSet::setOverallStatus (const ProgressStatus & iProgressStatus) [inline]`

Set/update the overall progress status (absolute, for all the events).

Definition at line 76 of file [ProgressStatusSet.hpp](#).

Referenced by [stdair::EventQueue::popEvent\(\)](#).

34.110.3.7 `void stdair::ProgressStatusSet::fromStream (std::istream & iIn) [virtual]`

Read a Business Object from an input stream.

Parameters

<i>istream</i> & the input stream.

Reimplemented from [stdair::StructAbstract](#).

Definition at line 40 of file [ProgressStatusSet.cpp](#).

34.110.3.8 `const std::string stdair::ProgressStatusSet::describe () const` `[virtual]`

Display of the structure.

Implements [stdair::StructAbstract](#).

Definition at line 44 of file [ProgressStatusSet.cpp](#).

References [stdair::ProgressStatus::getActualNb\(\)](#), [stdair::ProgressStatus::getCurrentNb\(\)](#), and [stdair::ProgressStatus::getExpectedNb\(\)](#).

34.110.3.9 `void stdair::StructAbstract::toStream (std::ostream & ioOut) const` `[inline, inherited]`

Dump a Business Object into an output stream.

Parameters

<i>ostream</i> & the output stream.

Reimplemented in [stdair::YieldRange](#), [stdair::AirlineStruct](#), [stdair::BookingRequestStruct](#), [stdair::CancellationStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::TravelSolutionStruct](#), and [stdair::VirtualClassStruct](#).

Definition at line 29 of file [StructAbstract.hpp](#).

References [stdair::StructAbstract::describe\(\)](#).

Referenced by [operator<<\(\)](#).

The documentation for this struct was generated from the following files:

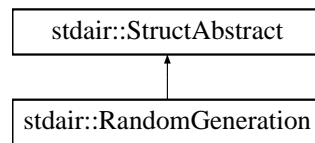
- [stdair/basic/ProgressStatusSet.hpp](#)
- [stdair/basic/ProgressStatusSet.cpp](#)

34.111 stdair::RandomGeneration Struct Reference

Class holding a random generator.

```
#include <stdair/basic/RandomGeneration.hpp>
```

Inheritance diagram for [stdair::RandomGeneration](#):



Public Member Functions

- [RealNumber_T generateUniform01 \(\)](#)
- [RealNumber_T operator\(\) \(\)](#)
- [RealNumber_T generateUniform \(const \[RealNumber_T\]\(#\) &, const \[RealNumber_T\]\(#\) &\)](#)
- [RealNumber_T generateNormal \(const \[RealNumber_T\]\(#\) &, const \[RealNumber_T\]\(#\) &\)](#)
- [RealNumber_T generateExponential \(const \[RealNumber_T\]\(#\) &\)](#)
- [BaseGenerator_T & getBaseGenerator \(\)](#)
- [const std::string describe \(\) const](#)
- [RandomGeneration \(const \[RandomSeed_T\]\(#\) &\)](#)
- [RandomGeneration \(\)](#)
- [~RandomGeneration \(\)](#)
- [void init \(const \[RandomSeed_T\]\(#\) &\)](#)
- [void toStream \(std::ostream &ioOut\) const](#)
- [virtual void fromStream \(std::istream &ioIn\)](#)

Public Attributes

- [BaseGenerator_T _generator](#)

34.111.1 Detailed Description

Class holding a random generator.

Definition at line 17 of file [RandomGeneration.hpp](#).

34.111.2 Constructor & Destructor Documentation

34.111.2.1 stdair::RandomGeneration::RandomGeneration (const [RandomSeed_T](#) & *iSeed*)

Main constructor.

Definition at line 27 of file [RandomGeneration.cpp](#).

34.111.2.2 stdair::RandomGeneration::RandomGeneration ()

Default constructor.

Note

As per Boost bug #3516 (<https://svn.boost.org/trac/boost/ticket/3516>) the seed should not be set to 0 (at least on versions of Boost lower than 1.44).

Definition at line 23 of file [RandomGeneration.cpp](#).

34.111.2.3 stdair::RandomGeneration::~~RandomGeneration ()

Destructor.

Definition at line 37 of file [RandomGeneration.cpp](#).

34.111.3 Member Function Documentation

34.111.3.1 RealNumber_T stdair::RandomGeneration::generateUniform01 ()

Generate a randomised number following a uniform distribution between 0 (included) and 1 (excluded).

Definition at line 53 of file [RandomGeneration.cpp](#).

References [_generator](#).

Referenced by [generateNormal\(\)](#), [generateUniform\(\)](#), and [operator\(\)](#).

34.111.3.2 RealNumber_T stdair::RandomGeneration::operator() () [inline]

Same as [generateUniform01\(\)](#). That operator is provided for convenient reasons.

Definition at line 30 of file [RandomGeneration.hpp](#).

References [generateUniform01\(\)](#).

34.111.3.3 RealNumber_T stdair::RandomGeneration::generateUniform (const RealNumber_T & iMinValue, const RealNumber_T & iMaxValue)

Generate a randomized number following a uniform distribution between a minimum (included) and a maximum (excluded) value.

Definition at line 59 of file [RandomGeneration.cpp](#).

References [generateUniform01\(\)](#).

34.111.3.4 RealNumber_T stdair::RandomGeneration::generateNormal (const RealNumber_T & mu, const RealNumber_T & sigma)

Generate a randomized number following a normal distribution specified by a mean and a standard deviation.

Definition at line 68 of file [RandomGeneration.cpp](#).

References [generateUniform01\(\)](#).

Referenced by [stdair::BookingClass::generateDemandSamples\(\)](#).

34.111.3.5 RealNumber_T stdair::RandomGeneration::generateExponential (const RealNumber_T & lambda)

Generate a randomized number following an exponential distribution specified by a mean and a lambda parameter.

Definition at line 86 of file [RandomGeneration.cpp](#).

References [_generator](#).

34.111.3.6 BaseGenerator_T& stdair::RandomGeneration::getBaseGenerator ()
[inline]

Retrieve the base generator for initialising other random generators.

Definition at line 56 of file [RandomGeneration.hpp](#).

References [_generator](#).

34.111.3.7 const std::string stdair::RandomGeneration::describe () const [virtual]

Give a description of the structure (for display purposes).

Implements [stdair::StructAbstract](#).

Definition at line 46 of file [RandomGeneration.cpp](#).

References [_generator](#).

34.111.3.8 void stdair::RandomGeneration::init (const RandomSeed_T & iSeed)

Initialise the random generator.

A uniform random number distribution is defined, which produces "real" values between 0 and 1 (0 inclusive, 1 exclusive).

Definition at line 41 of file [RandomGeneration.cpp](#).

References [_generator](#).

34.111.3.9 void stdair::StructAbstract::toStream (std::ostream & ioOut) const [inline, inherited]

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Reimplemented in [stdair::YieldRange](#), [stdair::AirlineStruct](#), [stdair::BookingRequestStruct](#), [stdair::CancellationStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::TravelSolutionStruct](#), and [stdair::VirtualClassStruct](#).

Definition at line 29 of file [StructAbstract.hpp](#).

References [stdair::StructAbstract::describe\(\)](#).

Referenced by [operator<<\(\)](#).

34.111.3.10 virtual void stdair::StructAbstract::fromStream (std::istream & *ioIn*)
[inline, virtual, inherited]

Read a Business Object from an input stream.

Parameters

<i>istream&</i> the input stream.

Reimplemented in [stdair::ProgressStatusSet](#), [stdair::YieldRange](#), [stdair::AirlineStruct](#), [stdair::BookingRequestStruct](#), [stdair::CancellationStruct](#), [stdair::EventStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::TravelSolutionStruct](#), and [stdair::VirtualClassStruct](#).

Definition at line 38 of file [StructAbstract.hpp](#).

Referenced by [operator>>\(\)](#).

34.111.4 Member Data Documentation

34.111.4.1 BaseGenerator_T stdair::RandomGeneration::_generator

Random number generator engine.

The random number generator is currently based on boost::minstd_rand. Alternates are boost::mt19937, boost::ecuyer1988.

Definition at line 112 of file [RandomGeneration.hpp](#).

Referenced by [describe\(\)](#), [generateExponential\(\)](#), [generateUniform01\(\)](#), [getBaseGenerator\(\)](#), and [init\(\)](#).

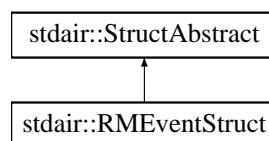
The documentation for this struct was generated from the following files:

- [stdair/basic/RandomGeneration.hpp](#)
- [stdair/basic/RandomGeneration.cpp](#)

34.112 stdair::RMEventStruct Struct Reference

```
#include <stdair/bom/RMEventStruct.hpp>
```

Inheritance diagram for stdair::RMEventStruct:



Public Member Functions

- const [AirlineCode_T](#) & [getAirlineCode](#) () const
- const [KeyDescription_T](#) & [getFlightDateDescription](#) () const
- const [DateTime_T](#) & [getRMEventTime](#) () const
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- const std::string [describe](#) () const
- [RMEventStruct](#) (const [AirlineCode_T](#) &, const [KeyDescription_T](#) &, const [DateTime_T](#) &)
- [RMEventStruct](#) (const [RMEventStruct](#) &)
- [RMEventStruct](#) ()
- [~RMEventStruct](#) ()

34.112.1 Detailed Description

Structure holding the elements of a snapshot .

Definition at line 19 of file [RMEventStruct.hpp](#).

34.112.2 Constructor & Destructor Documentation

34.112.2.1 `stdair::RMEventStruct::RMEventStruct (const AirlineCode_T & iAirlineCode,
const KeyDescription_T & iFlightDateDescription, const DateTime_T &
iRMEventTime)`

Constructor.

Definition at line 27 of file [RMEventStruct.cpp](#).

34.112.2.2 `stdair::RMEventStruct::RMEventStruct (const RMEventStruct & iRMEvent)`

Copy constructor.

Definition at line 19 of file [RMEventStruct.cpp](#).

34.112.2.3 `stdair::RMEventStruct::RMEventStruct ()`

Default constructor.

It is private so that it can not be used.

Definition at line 13 of file [RMEventStruct.cpp](#).

34.112.2.4 `stdair::RMEventStruct::~~RMEventStruct ()`

Destructor.

Definition at line 36 of file [RMEventStruct.cpp](#).

34.112.3 Member Function Documentation

34.112.3.1 `const AirlineCode_T& stdair::RMEventStruct::getAirlineCode () const`
[inline]

Get the airline code.

Definition at line 23 of file [RMEventStruct.hpp](#).

34.112.3.2 `const KeyDescription_T& stdair::RMEventStruct::getFlightDateDescription ()`
`const` [inline]

Get the string describing the flight-date key.

Definition at line 28 of file [RMEventStruct.hpp](#).

34.112.3.3 `const DateTime_T& stdair::RMEventStruct::getRMEventTime () const`
[inline]

Get the snapshot action time.

Definition at line 33 of file [RMEventStruct.hpp](#).

34.112.3.4 `void stdair::RMEventStruct::toStream (std::ostream & ioOut) const`

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Reimplemented from [stdair::StructAbstract](#).

Definition at line 40 of file [RMEventStruct.cpp](#).

References [describe\(\)](#).

34.112.3.5 `void stdair::RMEventStruct::fromStream (std::istream & ioIn)` [virtual]

Read a Business Object from an input stream.

Parameters

<i>istream&</i> the input stream.

Reimplemented from [stdair::StructAbstract](#).

Definition at line 45 of file [RMEventStruct.cpp](#).

34.112.3.6 `const std::string stdair::RMEventStruct::describe () const` [virtual]

Display of the structure.

Implements [stdair::StructAbstract](#).

Definition at line 49 of file [RMEventStruct.cpp](#).

Referenced by [toStream\(\)](#).

The documentation for this struct was generated from the following files:

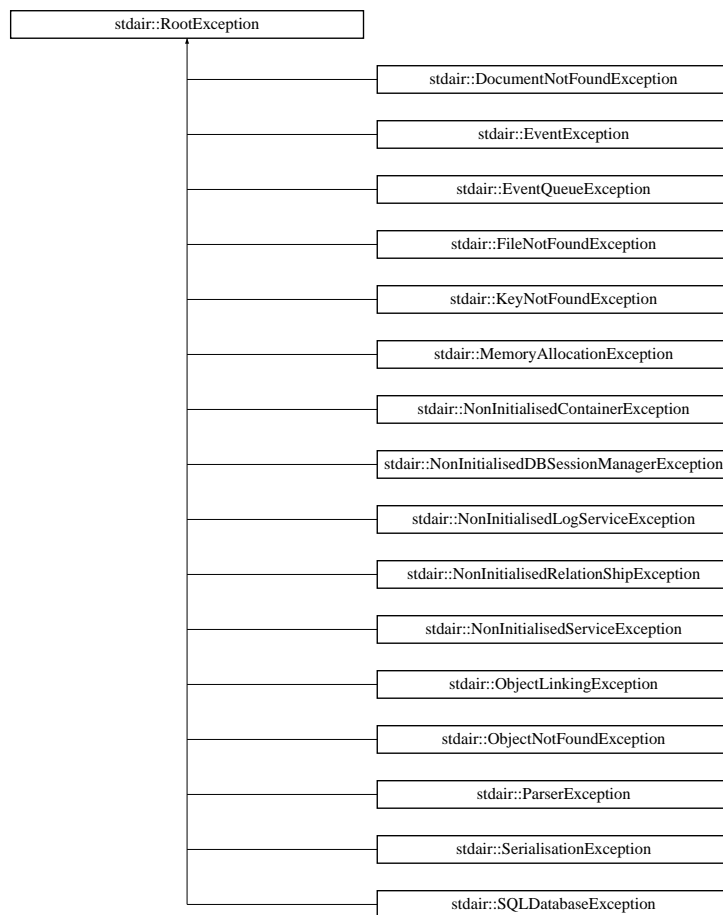
- [stdair/bom/RMEventStruct.hpp](#)
- [stdair/bom/RMEventStruct.cpp](#)

34.113 stdair::RootException Class Reference

Root of the stdair exceptions.

```
#include <stdair/stdair_exceptions.hpp>
```

Inheritance diagram for stdair::RootException:



Public Member Functions

- [RootException](#) (const std::string &iWhat)
- [RootException](#) ()

- virtual [~RootException](#) () throw ()
- const char * [what](#) () const throw ()

Protected Attributes

- std::string [_what](#)

34.113.1 Detailed Description

Root of the stdair exceptions.

All the stdair exceptions inherit from that root, allowing to catch them and to spot them easily when arising in code wrapping the stdair library.

Definition at line 19 of file [stdair_exceptions.hpp](#).

34.113.2 Constructor & Destructor Documentation

34.113.2.1 stdair::RootException::RootException (const std::string & *iWhat*) [inline]

Main Constructor.

Definition at line 24 of file [stdair_exceptions.hpp](#).

34.113.2.2 stdair::RootException::RootException () [inline]

Default constructor.

Definition at line 28 of file [stdair_exceptions.hpp](#).

34.113.2.3 virtual stdair::RootException::~~RootException () throw () [inline, virtual]

Destructor.

Definition at line 33 of file [stdair_exceptions.hpp](#).

34.113.3 Member Function Documentation

34.113.3.1 const char* stdair::RootException::what () const throw () [inline]

Give the details of the exception.

Definition at line 38 of file [stdair_exceptions.hpp](#).

References [_what](#).

34.113.4 Member Data Documentation

34.113.4.1 std::string stdair::RootException::_what [protected]

Details for the exception.

Definition at line 46 of file [stdair_exceptions.hpp](#).

Referenced by [what\(\)](#).

The documentation for this class was generated from the following file:

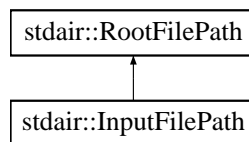
- [stdair/stdair_exceptions.hpp](#)

34.114 stdair::RootFilePath Class Reference

Root of the input and output files.

```
#include <stdair/stdair_file.hpp>
```

Inheritance diagram for stdair::RootFilePath:



Public Member Functions

- [RootFilePath](#) (const [Filename_T](#) &iFilename)
- [RootFilePath](#) ()
- virtual [~RootFilePath](#) ()
- const char * [name](#) () const

Protected Attributes

- const [Filename_T](#) [_filename](#)

34.114.1 Detailed Description

Root of the input and output files.

All the files inherit from that root.

Definition at line 22 of file [stdair_file.hpp](#).

34.114.2 Constructor & Destructor Documentation

34.114.2.1 `stdair::RootFilePath::RootFilePath (const Filename_T & iFilename)`
[inline]

Main Constructor.

Definition at line 27 of file [stdair_file.hpp](#).

34.114.2.2 `stdair::RootFilePath::RootFilePath ()` [inline]

Default constructor.

Definition at line 32 of file [stdair_file.hpp](#).

34.114.2.3 `virtual stdair::RootFilePath::~~RootFilePath ()` [inline, virtual]

Destructor.

Definition at line 37 of file [stdair_file.hpp](#).

34.114.3 Member Function Documentation

34.114.3.1 `const char* stdair::RootFilePath::name () const` [inline]

Give the details of the exception.

Definition at line 42 of file [stdair_file.hpp](#).

References [_filename](#).

34.114.4 Member Data Documentation

34.114.4.1 `const Filename_T stdair::RootFilePath::_filename` [protected]

Name of the file.

Definition at line 50 of file [stdair_file.hpp](#).

Referenced by [name\(\)](#).

The documentation for this class was generated from the following file:

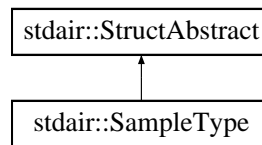
- [stdair/stdair_file.hpp](#)

34.115 stdair::SampleType Struct Reference

Enumeration of BOM sample types.

```
#include <stdair/basic/SampleType.hpp>
```

Inheritance diagram for `stdair::SampleType`:



Public Types

- enum [EN_SampleType](#) {
[ALL](#) = 0, [A4P](#), [RMS](#), [INV](#),
[SCH](#), [RAC](#), [FQT](#), [CRS](#),
[DEM](#), [EVT](#), [CCM](#), [LAST_VALUE](#) }

Public Member Functions

- [EN_SampleType](#) [getType](#) () const
- std::string [getTypeAsString](#) () const
- const std::string [describe](#) () const
- bool [operator==](#) (const [EN_SampleType](#) &) const
- [SampleType](#) (const [EN_SampleType](#) &)
- [SampleType](#) (const char iType)
- [SampleType](#) (const [SampleType](#) &)
- void [toStream](#) (std::ostream &ioOut) const
- virtual void [fromStream](#) (std::istream &ioIn)

Static Public Member Functions

- static const std::string & [getLabel](#) (const [EN_SampleType](#) &)
- static char [getTypeLabel](#) (const [EN_SampleType](#) &)
- static std::string [getTypeLabelAsString](#) (const [EN_SampleType](#) &)
- static std::string [describeLabels](#) ()

34.115.1 Detailed Description

Enumeration of BOM sample types.

In order to test some components, it is often easier to fill the BOM tree with hard-coded structures than set up CSV input files and parsing them. That enumeration structure tells for which component(s) the sample BOM tree should be built. By default, a BOM sample tree is built for all the components, i.e., it contains StdAir objects for all the other components (AirInv, AirSched, etc).

Definition at line 25 of file [SampleType.hpp](#).

34.115.2 Member Enumeration Documentation

34.115.2.1 enum stdair::SampleType::EN_SampleType

Enumerator:

ALL
A4P
RMS
INV
SCH
RAC
FQT
CRS
DEM
EVT
CCM
LAST_VALUE

Definition at line 27 of file [SampleType.hpp](#).

34.115.3 Constructor & Destructor Documentation

34.115.3.1 stdair::SampleType::SampleType (const EN_SampleType & *iSampleType*)

Constructor.

Definition at line 36 of file [SampleType.cpp](#).

34.115.3.2 stdair::SampleType::SampleType (const char *iType*)

Constructor.

Definition at line 41 of file [SampleType.cpp](#).

References [A4P](#), [ALL](#), [CCM](#), [CRS](#), [DEM](#), [describeLabels\(\)](#), [EVT](#), [FQT](#), [INV](#), [LAST_VALUE](#), [RAC](#), [RMS](#), and [SCH](#).

34.115.3.3 stdair::SampleType::SampleType (const SampleType & *iSampleType*)

Default copy constructor.

Definition at line 31 of file [SampleType.cpp](#).

34.115.4 Member Function Documentation

34.115.4.1 `const std::string & stdair::SampleType::getLabel (const EN_SampleType & iType) [static]`

Get the label as a string (e.g., "Inventory" or "Schedule").

Definition at line 67 of file [SampleType.cpp](#).

34.115.4.2 `char stdair::SampleType::getTypeLabel (const EN_SampleType & iType) [static]`

Get the label as a single char (e.g., 'I' or 'S').

Definition at line 72 of file [SampleType.cpp](#).

34.115.4.3 `std::string stdair::SampleType::getTypeLabelAsString (const EN_SampleType & iType) [static]`

Get the label as a string of a single char (e.g., "I" or "S").

Definition at line 77 of file [SampleType.cpp](#).

34.115.4.4 `std::string stdair::SampleType::describeLabels () [static]`

List the labels.

Definition at line 84 of file [SampleType.cpp](#).

References [LAST_VALUE](#).

Referenced by [SampleType\(\)](#).

34.115.4.5 `SampleType::EN_SampleType stdair::SampleType::getType () const`

Get the enumerated value.

Definition at line 96 of file [SampleType.cpp](#).

34.115.4.6 `std::string stdair::SampleType::getTypeAsString () const`

Get the enumerated value as a short string (e.g., "I" or "S").

Definition at line 101 of file [SampleType.cpp](#).

34.115.4.7 `const std::string stdair::SampleType::describe () const [virtual]`

Give a description of the structure (e.g., "Inventory" or "Schedule").

Implements [stdair::StructAbstract](#).

Definition at line 108 of file [SampleType.cpp](#).

34.115.4.8 `bool stdair::SampleType::operator== (const EN_SampleType & iType) const`

Comparison operator.

Definition at line 115 of file [SampleType.cpp](#).

34.115.4.9 `void stdair::StructAbstract::toStream (std::ostream & ioOut) const` [`inline`, `inherited`]

Dump a Business Object into an output stream.

Parameters

<i>ostream</i> & the output stream.

Reimplemented in [stdair::YieldRange](#), [stdair::AirlineStruct](#), [stdair::BookingRequestStruct](#), [stdair::CancellationStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::TravelSolutionStruct](#), and [stdair::VirtualClassStruct](#).

Definition at line 29 of file [StructAbstract.hpp](#).

References [stdair::StructAbstract::describe\(\)](#).

Referenced by [operator<<\(\)](#).

34.115.4.10 `virtual void stdair::StructAbstract::fromStream (std::istream & ioIn)` [`inline`, `virtual`, `inherited`]

Read a Business Object from an input stream.

Parameters

<i>istream</i> & the input stream.

Reimplemented in [stdair::ProgressStatusSet](#), [stdair::YieldRange](#), [stdair::AirlineStruct](#), [stdair::BookingRequestStruct](#), [stdair::CancellationStruct](#), [stdair::EventStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::TravelSolutionStruct](#), and [stdair::VirtualClassStruct](#).

Definition at line 38 of file [StructAbstract.hpp](#).

Referenced by [operator>>\(\)](#).

The documentation for this struct was generated from the following files:

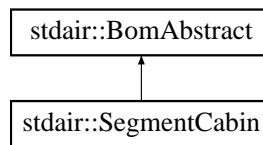
- [stdair/basic/SampleType.hpp](#)
- [stdair/basic/SampleType.cpp](#)

34.116 stdair::SegmentCabin Class Reference

Class representing the actual attributes for an airline segment-cabin.

`#include <stdair/bom/SegmentCabin.hpp>`

Inheritance diagram for `stdair::SegmentCabin`:



Public Types

- typedef [SegmentCabinKey](#) [Key_T](#)

Public Member Functions

- const [Key_T](#) & [getKey](#) () const
- [BomAbstract](#) *const [getParent](#) () const
- const [HolderMap_T](#) & [getHolderMap](#) () const
- const [CabinCode_T](#) & [getCabinCode](#) () const
- const [MapKey_T](#) [getFullerKey](#) () const
- const [GuillotineBlock](#) & [getGuillotineBlock](#) () const
- const [CabinCapacity_T](#) & [getCapacity](#) () const
- const [BlockSpace_T](#) & [getBlockSpace](#) () const
- const [BlockSpace_T](#) & [getMIN](#) () const
- const [UPR_T](#) & [getUPR](#) () const
- const [NbOfBookings_T](#) & [getBookingCounter](#) () const
- const [CommittedSpace_T](#) & [getCommittedSpace](#) () const
- const [Availability_T](#) & [getAvailabilityPool](#) () const
- const [BidPrice_T](#) & [getCurrentBidPrice](#) () const
- const [BidPriceVector_T](#) & [getBidPriceVector](#) () const
- const bool [getFareFamilyStatus](#) () const
- void [setGuillotineBlock](#) ([GuillotineBlock](#) &ioGuillotine)
- void [setCapacity](#) (const [CabinCapacity_T](#) &iCapacity)
- void [setBlockSpace](#) (const [BlockSpace_T](#) &iBlockSpace)
- void [setMIN](#) (const [BlockSpace_T](#) &iMIN)
- void [setUPR](#) (const [UPR_T](#) &iUPR)
- void [setBookingCounter](#) (const [NbOfBookings_T](#) &iBookingCounter)
- void [setCommittedSpace](#) (const [CommittedSpace_T](#) &iCommittedSpace)
- void [setAvailabilityPool](#) (const [Availability_T](#) &iAvailabilityPool)
- void [setBidPriceVector](#) (const [BidPriceVector_T](#) &iBPV)
- void [activateFareFamily](#) ()
- void [updateFromReservation](#) (const [NbOfBookings_T](#) &)
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- std::string [toString](#) () const
- const std::string [describeKey](#) () const
- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int iFileVersion)

Protected Member Functions

- [SegmentCabin](#) (const [Key_T](#) &)
- virtual [~SegmentCabin](#) ()

Protected Attributes

- [Key_T _key](#)
- [BomAbstract](#) * [_parent](#)
- [HolderMap_T](#) [_holderMap](#)
- [GuillotineBlock](#) * [_guillotineBlock](#)
- [CabinCapacity_T](#) [_capacity](#)
- [BlockSpace_T](#) [_blockSpace](#)
- [BlockSpace_T](#) [_min](#)
- [UPR_T](#) [_upr](#)
- [NbOfBookings_T](#) [_bookingCounter](#)
- [CommittedSpace_T](#) [_committedSpace](#)
- [Availability_T](#) [_availabilityPool](#)
- [BidPriceVector_T](#) [_bidPriceVector](#)
- [BidPrice_T](#) [_currentBidPrice](#)
- bool [_fareFamilyActivation](#)

Friends

- class [FacBom](#)
- class [FacBomManager](#)
- class [boost::serialization::access](#)

34.116.1 Detailed Description

Class representing the actual attributes for an airline segment-cabin.

Definition at line 31 of file [SegmentCabin.hpp](#).

34.116.2 Member Typedef Documentation

34.116.2.1 typedef SegmentCabinKey stdair::SegmentCabin::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 41 of file [SegmentCabin.hpp](#).

34.116.3 Constructor & Destructor Documentation

34.116.3.1 stdair::SegmentCabin::SegmentCabin (const Key_T & iKey) [protected]

Constructor.

Definition at line 29 of file [SegmentCabin.cpp](#).

34.116.3.2 stdair::SegmentCabin::~~SegmentCabin () [protected, virtual]

Destructor.

Definition at line 42 of file [SegmentCabin.cpp](#).

34.116.4 Member Function Documentation

34.116.4.1 const Key_T& stdair::SegmentCabin::getKey () const [inline]

Get the segment-cabin key (cabin code).

Definition at line 49 of file [SegmentCabin.hpp](#).

References [_key](#).

34.116.4.2 BomAbstract* const stdair::SegmentCabin::getParent () const [inline]

Get the parent object.

Definition at line 56 of file [SegmentCabin.hpp](#).

References [_parent](#).

34.116.4.3 const HolderMap_T& stdair::SegmentCabin::getHolderMap () const [inline]

Get the map of children holders.

Definition at line 63 of file [SegmentCabin.hpp](#).

References [_holderMap](#).

34.116.4.4 const CabinCode_T& stdair::SegmentCabin::getCabinCode () const [inline]

Get the cabin code (primary key).

Definition at line 70 of file [SegmentCabin.hpp](#).

References [_key](#), and [stdair::SegmentCabinKey::getCabinCode\(\)](#).

Referenced by [getFullerKey\(\)](#).

34.116.4.5 const MapKey_T stdair::SegmentCabin::getFullerKey () const

Get the (segment-date, segment-cabin) key (board point, off point and cabin code).

Note

That method assumes that the parent object derives from the [SegmentDate](#) class, as it needs to have access to the [describeKey\(\)](#) method.

Definition at line 46 of file [SegmentCabin.cpp](#).

References [stdair::DEFAULT_KEY_FLD_DELIMITER](#), [stdair::SegmentDate::describeKey\(\)](#), and [getCabinCode\(\)](#).

34.116.4.6 `const GuillotineBlock& stdair::SegmentCabin::getGuillotineBlock () const`
[inline]

Get the guillotine block.

Definition at line 85 of file [SegmentCabin.hpp](#).

References [_guillotineBlock](#).

34.116.4.7 `const CabinCapacity_T& stdair::SegmentCabin::getCapacity () const`
[inline]

Get the cabin capacity.

Definition at line 91 of file [SegmentCabin.hpp](#).

References [_capacity](#).

34.116.4.8 `const BlockSpace_T& stdair::SegmentCabin::getBlockSpace () const`
[inline]

Get the blocked number of bookings.

Definition at line 96 of file [SegmentCabin.hpp](#).

References [_blockSpace](#).

34.116.4.9 `const BlockSpace_T& stdair::SegmentCabin::getMIN () const` [inline]

Get the blocked number of bookings.

Definition at line 101 of file [SegmentCabin.hpp](#).

References [_min](#).

34.116.4.10 `const UPR_T& stdair::SegmentCabin::getUPR () const` [inline]

Unsold Protection (UPR).

Definition at line 106 of file [SegmentCabin.hpp](#).

References [_upr](#).

34.116.4.11 `const NbOfBookings_T& stdair::SegmentCabin::getBookingCounter () const`
[inline]

Get the booking counter.

Definition at line 111 of file [SegmentCabin.hpp](#).

References [_bookingCounter](#).

34.116.4.12 **const CommittedSpace_T&** stdair::SegmentCabin::getCommittedSpace ()
const [inline]

Get the committed Space value.

Definition at line 116 of file [SegmentCabin.hpp](#).

References [_committedSpace](#).

34.116.4.13 **const Availability_T&** stdair::SegmentCabin::getAvailabilityPool () **const**
[inline]

Get the availability pool value.

Definition at line 121 of file [SegmentCabin.hpp](#).

References [_availabilityPool](#).

34.116.4.14 **const BidPrice_T&** stdair::SegmentCabin::getCurrentBidPrice () **const**
[inline]

Retrieve the current Bid-Price.

Definition at line 126 of file [SegmentCabin.hpp](#).

References [_currentBidPrice](#).

34.116.4.15 **const BidPriceVector_T&** stdair::SegmentCabin::getBidPriceVector () **const**
[inline]

Retrieve the Bid-Price Vector.

Definition at line 131 of file [SegmentCabin.hpp](#).

References [_bidPriceVector](#).

34.116.4.16 **const bool** stdair::SegmentCabin::getFareFamilyStatus () **const** [inline]

Retrieve the status of fare family.

Definition at line 136 of file [SegmentCabin.hpp](#).

References [_fareFamilyActivation](#).

34.116.4.17 **void** stdair::SegmentCabin::setGuillotineBlock (**GuillotineBlock & ioGuillotine**)
[inline]

Set the guillotine block.

Definition at line 143 of file [SegmentCabin.hpp](#).

References [_guillotineBlock](#).

34.116.4.18 void stdair::SegmentCabin::setCapacity (const CabinCapacity_T & iCapacity)
[inline]

Set the cabin capacity.

Definition at line 148 of file [SegmentCabin.hpp](#).

References [_capacity](#).

34.116.4.19 void stdair::SegmentCabin::setBlockSpace (const BlockSpace_T & iBlockSpace) [inline]

Set the blocked number of seats.

Definition at line 153 of file [SegmentCabin.hpp](#).

References [_blockSpace](#).

34.116.4.20 void stdair::SegmentCabin::setMIN (const BlockSpace_T & iMIN)
[inline]

Set the blocked number of seats.

Definition at line 158 of file [SegmentCabin.hpp](#).

References [_min](#).

34.116.4.21 void stdair::SegmentCabin::setUPR (const UPR_T & iUPR) [inline]

Set the Unsold Protection (UPR).

Definition at line 163 of file [SegmentCabin.hpp](#).

References [_upr](#).

34.116.4.22 void stdair::SegmentCabin::setBookingCounter (const NbOfBookings_T & iBookingCounter) [inline]

Set the total number of bookings.

Definition at line 168 of file [SegmentCabin.hpp](#).

References [_bookingCounter](#).

34.116.4.23 void stdair::SegmentCabin::setCommittedSpace (const CommittedSpace_T & iCommittedSpace) [inline]

Set the value of committed space.

Definition at line 173 of file [SegmentCabin.hpp](#).

References [_committedSpace](#).

34.116.4.24 void stdair::SegmentCabin::setAvailabilityPool (const Availability_T & iAvailabilityPool) [inline]

Set the value of availability pool.

Definition at line 178 of file [SegmentCabin.hpp](#).

References [_availabilityPool](#).

34.116.4.25 void stdair::SegmentCabin::setBidPriceVector (const BidPriceVector_T & iBPV) [inline]

Set the Bid-Price Vector.

Definition at line 183 of file [SegmentCabin.hpp](#).

References [_bidPriceVector](#).

34.116.4.26 void stdair::SegmentCabin::activateFareFamily () [inline]

Activate fare family.

Definition at line 188 of file [SegmentCabin.hpp](#).

References [_fareFamilyActivation](#).

34.116.4.27 void stdair::SegmentCabin::updateFromReservation (const NbOfBookings_T & iNbOfBookings)

Register a sale.

Definition at line 63 of file [SegmentCabin.cpp](#).

References [_committedSpace](#).

34.116.4.28 void stdair::SegmentCabin::toStream (std::ostream & ioOut) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Implements [stdair::BomAbstract](#).

Definition at line 205 of file [SegmentCabin.hpp](#).

References [toString\(\)](#).

34.116.4.29 void stdair::SegmentCabin::fromStream (std::istream & iIn) [inline, virtual]

Read a Business Object from an input stream.

Parameters

<i>istream&</i> the input stream.

Implements [stdair::BomAbstract](#).

Definition at line 214 of file [SegmentCabin.hpp](#).

34.116.4.30 `std::string stdair::SegmentCabin::toString () const` `[virtual]`

Get the serialised version of the Business Object.

Implements [stdair::BomAbstract](#).

Definition at line 55 of file [SegmentCabin.cpp](#).

References [describeKey\(\)](#).

Referenced by [toStream\(\)](#).

34.116.4.31 `const std::string stdair::SegmentCabin::describeKey () const` `[inline]`

Get a string describing the key.

Definition at line 225 of file [SegmentCabin.hpp](#).

References [_key](#), and [stdair::SegmentCabinKey::toString\(\)](#).

Referenced by [toString\(\)](#).

34.116.4.32 `template<class Archive > void stdair::SegmentCabin::serialize (Archive & ar,
const unsigned int iFileVersion)`

Serialisation.

Definition at line 229 of file [CmdBomSerialiser.cpp](#).

References [_key](#).

34.116.5 Friends And Related Function Documentation

34.116.5.1 `friend class FacBom` `[friend]`

Definition at line 32 of file [SegmentCabin.hpp](#).

34.116.5.2 `friend class FacBomManager` `[friend]`

Definition at line 33 of file [SegmentCabin.hpp](#).

34.116.5.3 `friend class boost::serialization::access` `[friend]`

Definition at line 34 of file [SegmentCabin.hpp](#).

34.116.6 Member Data Documentation

34.116.6.1 `Key_T stdair::SegmentCabin::_key` `[protected]`

Primary key (cabin code).

Definition at line 279 of file [SegmentCabin.hpp](#).

Referenced by [describeKey\(\)](#), [getCabinCode\(\)](#), [getKey\(\)](#), and [serialize\(\)](#).

34.116.6.2 BomAbstract* stdair::SegmentCabin::_parent [protected]

Pointer on the parent class ([SegmentDate](#)).

Definition at line 284 of file [SegmentCabin.hpp](#).

Referenced by [getParent\(\)](#).

34.116.6.3 HolderMap_T stdair::SegmentCabin::_holderMap [protected]

Map holding the children ([FareFamily](#) or [BookingClass](#) objects).

Definition at line 289 of file [SegmentCabin.hpp](#).

Referenced by [getHolderMap\(\)](#).

34.116.6.4 GuillotineBlock* stdair::SegmentCabin::_guillotineBlock
[protected]

The guillotine block used for Revenue Management activities.

Definition at line 294 of file [SegmentCabin.hpp](#).

Referenced by [getGuillotineBlock\(\)](#), and [setGuillotineBlock\(\)](#).

34.116.6.5 CabinCapacity_T stdair::SegmentCabin::_capacity [protected]

Capacity of the cabin.

Definition at line 297 of file [SegmentCabin.hpp](#).

Referenced by [getCapacity\(\)](#), and [setCapacity\(\)](#).

34.116.6.6 BlockSpace_T stdair::SegmentCabin::_blockSpace [protected]

Blocked capacity.

Definition at line 300 of file [SegmentCabin.hpp](#).

Referenced by [getBlockSpace\(\)](#), and [setBlockSpace\(\)](#).

34.116.6.7 BlockSpace_T stdair::SegmentCabin::_min [protected]

Blocked number of seats.

Definition at line 303 of file [SegmentCabin.hpp](#).

Referenced by [getMIN\(\)](#), and [setMIN\(\)](#).

34.116.6.8 UPR_T stdair::SegmentCabin::_upr [protected]

Unsold Protection (UPR).

Definition at line 306 of file [SegmentCabin.hpp](#).

Referenced by [getUPR\(\)](#), and [setUPR\(\)](#).

34.116.6.9 NbOfBookings_T stdair::SegmentCabin::_bookingCounter
[protected]

Aggregated number of bookings.

Definition at line 309 of file [SegmentCabin.hpp](#).

Referenced by [getBookingCounter\(\)](#), and [setBookingCounter\(\)](#).

34.116.6.10 CommittedSpace_T stdair::SegmentCabin::_committedSpace
[protected]

Aggregated committed space.

Definition at line 312 of file [SegmentCabin.hpp](#).

Referenced by [getCommittedSpace\(\)](#), [setCommittedSpace\(\)](#), and [updateFromReservation\(\)](#).

34.116.6.11 Availability_T stdair::SegmentCabin::_availabilityPool
[protected]

Aggregated availability pool.

Definition at line 315 of file [SegmentCabin.hpp](#).

Referenced by [getAvailabilityPool\(\)](#), and [setAvailabilityPool\(\)](#).

34.116.6.12 BidPriceVector_T stdair::SegmentCabin::_bidPriceVector
[protected]

Bid-Price Vector (BPV).

Definition at line 318 of file [SegmentCabin.hpp](#).

Referenced by [getBidPriceVector\(\)](#), and [setBidPriceVector\(\)](#).

34.116.6.13 BidPrice_T stdair::SegmentCabin::_currentBidPrice [protected]

Current Bid-Price (BP).

Definition at line 321 of file [SegmentCabin.hpp](#).

Referenced by [getCurrentBidPrice\(\)](#).

34.116.6.14 bool stdair::SegmentCabin::_fareFamilyActivation [protected]

Indicate if fare family is in use.

Definition at line 324 of file [SegmentCabin.hpp](#).

Referenced by [activateFareFamily\(\)](#), and [getFareFamilyStatus\(\)](#).

The documentation for this class was generated from the following files:

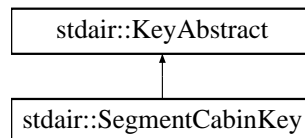
- [stdair/bom/SegmentCabin.hpp](#)
- [stdair/bom/SegmentCabin.cpp](#)
- [stdair/command/CmdBomSerialiser.cpp](#)

34.117 stdair::SegmentCabinKey Struct Reference

Key of a given segment-cabin, made of a cabin code (only).

```
#include <stdair/bom/SegmentCabinKey.hpp>
```

Inheritance diagram for stdair::SegmentCabinKey:



Public Member Functions

- [SegmentCabinKey](#) (const [CabinCode_T](#) &iCabinCode)
- [SegmentCabinKey](#) (const [SegmentCabinKey](#) &)
- [~SegmentCabinKey](#) ()
- const [CabinCode_T](#) & [getCabinCode](#) () const
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- const std::string [toString](#) () const
- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int iFileVersion)

Friends

- class [boost::serialization::access](#)

34.117.1 Detailed Description

Key of a given segment-cabin, made of a cabin code (only).

Definition at line 26 of file [SegmentCabinKey.hpp](#).

34.117.2 Constructor & Destructor Documentation

34.117.2.1 [stdair::SegmentCabinKey::SegmentCabinKey](#) (const [CabinCode_T](#) & *iCabinCode*)

Constructor.

Definition at line 23 of file [SegmentCabinKey.cpp](#).

34.117.2.2 stdair::SegmentCabinKey::SegmentCabinKey (const SegmentCabinKey & iKey)

Copy constructor.

Definition at line 28 of file [SegmentCabinKey.cpp](#).

34.117.2.3 stdair::SegmentCabinKey::~~SegmentCabinKey ()

Destructor.

Definition at line 33 of file [SegmentCabinKey.cpp](#).

34.117.3 Member Function Documentation

34.117.3.1 const CabinCode_T& stdair::SegmentCabinKey::getCabinCode () const [inline]

Get the cabin code.

Definition at line 56 of file [SegmentCabinKey.hpp](#).

Referenced by [stdair::SegmentCabin::getCabinCode\(\)](#).

34.117.3.2 void stdair::SegmentCabinKey::toStream (std::ostream & ioOut) const [virtual]

Dump a Business Object Key into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 37 of file [SegmentCabinKey.cpp](#).

References [toString\(\)](#).

34.117.3.3 void stdair::SegmentCabinKey::fromStream (std::istream & ioIn) [virtual]

Read a Business Object Key from an input stream.

Parameters

<i>istream&</i> the input stream.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 42 of file [SegmentCabinKey.cpp](#).

34.117.3.4 const std::string stdair::SegmentCabinKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a

given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 46 of file [SegmentCabinKey.cpp](#).

Referenced by [stdair::SegmentCabin::describeKey\(\)](#), [stdair::BomRetriever::retrieveDummySegmentCabin\(\)](#), and [toStream\(\)](#).

34.117.3.5 `template<class Archive > void stdair::SegmentCabinKey::serialize (Archive & ar, const unsigned int iFileVersion)`

Serialisation.

Definition at line 68 of file [SegmentCabinKey.cpp](#).

34.117.4 Friends And Related Function Documentation

34.117.4.1 `friend class boost::serialization::access [friend]`

Definition at line 27 of file [SegmentCabinKey.hpp](#).

The documentation for this struct was generated from the following files:

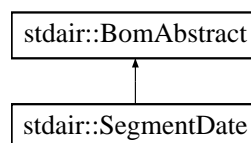
- [stdair/bom/SegmentCabinKey.hpp](#)
- [stdair/bom/SegmentCabinKey.cpp](#)

34.118 stdair::SegmentDate Class Reference

Class representing the actual attributes for an airline segment-date.

```
#include <stdair/bom/SegmentDate.hpp>
```

Inheritance diagram for `stdair::SegmentDate`:



Public Types

- typedef [SegmentDateKey](#) `Key_T`

Public Member Functions

- const [Key_T](#) & `getKey ()` const

- [BomAbstract](#) *const [getParent](#) () const
- const [AirportCode_T](#) & [getBoardingPoint](#) () const
- const [AirportCode_T](#) & [getOffPoint](#) () const
- const [HolderMap_T](#) & [getHolderMap](#) () const
- const [Date_T](#) & [getBoardingDate](#) () const
- const [Duration_T](#) & [getBoardingTime](#) () const
- const [Date_T](#) & [getOffDate](#) () const
- const [Duration_T](#) & [getOffTime](#) () const
- const [Duration_T](#) & [getElapsedTime](#) () const
- const [Distance_T](#) & [getDistance](#) () const
- const [DateOffset_T](#) & [getDateOffset](#) () const
- const [Duration_T](#) & [getTimeOffset](#) () const
- [SegmentDate](#) * [getOperatingSegmentDate](#) () const
- const [SegmentDateList_T](#) & [getMarketingSegmentDateList](#) () const
- void [setBoardingDate](#) (const [Date_T](#) & iBoardingDate)
- void [setBoardingTime](#) (const [Duration_T](#) & iBoardingTime)
- void [setOffDate](#) (const [Date_T](#) & iOffDate)
- void [setOffTime](#) (const [Duration_T](#) & iOffTime)
- void [setElapsedTime](#) (const [Duration_T](#) & iElapsedTime)
- void [setDistance](#) (const [Distance_T](#) & iDistance)
- void [linkWithOperating](#) ([SegmentDate](#) & iSegmentDate)
- void [toStream](#) (std::ostream & ioOut) const
- void [fromStream](#) (std::istream & ioIn)
- std::string [toString](#) () const
- const std::string [describeKey](#) () const
- template<class Archive >
void [serialize](#) (Archive & ar, const unsigned int iFileVersion)

Protected Member Functions

- [SegmentDate](#) (const [Key_T](#) &)
- virtual [~SegmentDate](#) ()

Protected Attributes

- [Key_T](#) _key
- [BomAbstract](#) * _parent
- [HolderMap_T](#) _holderMap
- [SegmentDate](#) * _operatingSegmentDate
- [SegmentDateList_T](#) _marketingSegmentDateList
- [Date_T](#) _boardingDate
- [Duration_T](#) _boardingTime
- [Date_T](#) _offDate
- [Duration_T](#) _offTime
- [Duration_T](#) _elapsedTime
- [Distance_T](#) _distance

Friends

- class [FacBom](#)
- class [FacBomManager](#)
- class [boost::serialization::access](#)

34.118.1 Detailed Description

Class representing the actual attributes for an airline segment-date.

Definition at line 33 of file [SegmentDate.hpp](#).

34.118.2 Member Typedef Documentation

34.118.2.1 typedef SegmentDateKey stdair::SegmentDate::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 43 of file [SegmentDate.hpp](#).

34.118.3 Constructor & Destructor Documentation

34.118.3.1 stdair::SegmentDate::SegmentDate (const Key_T & iKey) [protected]

Constructor.

Definition at line 31 of file [SegmentDate.cpp](#).

34.118.3.2 stdair::SegmentDate::~~SegmentDate () [protected, virtual]

Destructor.

Definition at line 37 of file [SegmentDate.cpp](#).

34.118.4 Member Function Documentation

34.118.4.1 const Key_T& stdair::SegmentDate::getKey () const [inline]

Get the segment-date key.

Definition at line 49 of file [SegmentDate.hpp](#).

References [_key](#).

34.118.4.2 BomAbstract* const stdair::SegmentDate::getParent () const [inline]

Get the parent object.

Definition at line 54 of file [SegmentDate.hpp](#).

References [_parent](#).

34.118.4.3 `const AirportCode_T& stdair::SegmentDate::getBoardingPoint () const`
[inline]

Get the boarding point (part of the primary key).

Definition at line 59 of file [SegmentDate.hpp](#).

References [_key](#), and [stdair::SegmentDateKey::getBoardingPoint\(\)](#).

34.118.4.4 `const AirportCode_T& stdair::SegmentDate::getOffPoint () const` [inline]

Get the off point (part of the primary key).

Definition at line 64 of file [SegmentDate.hpp](#).

References [_key](#), and [stdair::SegmentDateKey::getOffPoint\(\)](#).

34.118.4.5 `const HolderMap_T& stdair::SegmentDate::getHolderMap () const`
[inline]

Get the map of children holders.

Definition at line 69 of file [SegmentDate.hpp](#).

References [_holderMap](#).

34.118.4.6 `const Date_T& stdair::SegmentDate::getBoardingDate () const` [inline]

Get the boarding date.

Definition at line 74 of file [SegmentDate.hpp](#).

References [_boardingDate](#).

34.118.4.7 `const Duration_T& stdair::SegmentDate::getBoardingTime () const`
[inline]

Get the boarding time.

Definition at line 79 of file [SegmentDate.hpp](#).

References [_boardingTime](#).

34.118.4.8 `const Date_T& stdair::SegmentDate::getOffDate () const` [inline]

Get the off date.

Definition at line 84 of file [SegmentDate.hpp](#).

References [_offDate](#).

34.118.4.9 `const Duration_T& stdair::SegmentDate::getOffTime () const` [inline]

Get the off time.

Definition at line 89 of file [SegmentDate.hpp](#).

References [_offTime](#).

34.118.4.10 **const Duration_T& stdair::SegmentDate::getElapsedTime () const**
[inline]

Get the elapsed time.

Definition at line 94 of file [SegmentDate.hpp](#).

References [_elapsedTime](#).

34.118.4.11 **const Distance_T& stdair::SegmentDate::getDistance () const** [inline]

Get the distance.

Definition at line 99 of file [SegmentDate.hpp](#).

References [_distance](#).

34.118.4.12 **const DateOffset_T stdair::SegmentDate::getDateOffset () const**
[inline]

Get the date offset (off date - boarding date).

Definition at line 104 of file [SegmentDate.hpp](#).

References [_boardingDate](#), and [_offDate](#).

Referenced by [getTimeOffset\(\)](#).

34.118.4.13 **const Duration_T stdair::SegmentDate::getTimeOffset () const**

Get the time offset between boarding and off points.

It is defined as being:

$\text{TimeOffset} = (\text{OffTime} - \text{BoardingTime}) + (\text{OffDate} - \text{BoardingDate}) * 24$

- [ElapsedTime](#).

Definition at line 48 of file [SegmentDate.cpp](#).

References [_boardingTime](#), [_elapsedTime](#), [_offTime](#), and [getDateOffset\(\)](#).

34.118.4.14 **SegmentDate* stdair::SegmentDate::getOperatingSegmentDate () const**
[inline]

Get the "operating" segment date.

Definition at line 121 of file [SegmentDate.hpp](#).

References [_operatingSegmentDate](#).

34.118.4.15 **const SegmentDateList_T& stdair::SegmentDate::getMarketingSegmentDateList () const** [inline]

Get the list of marketing segment dates.

Definition at line 128 of file [SegmentDate.hpp](#).

References [_marketingSegmentDateList](#).

34.118.4.16 void stdair::SegmentDate::setBoardingDate (const Date_T & *iBoardingDate*)
[inline]

Set the boarding date.

Definition at line 135 of file [SegmentDate.hpp](#).

References [_boardingDate](#).

34.118.4.17 void stdair::SegmentDate::setBoardingTime (const Duration_T & *iBoardingTime*)
[inline]

Set the boarding time.

Definition at line 140 of file [SegmentDate.hpp](#).

References [_boardingTime](#).

34.118.4.18 void stdair::SegmentDate::setOffDate (const Date_T & *iOffDate*) [inline]

Set the off date.

Definition at line 145 of file [SegmentDate.hpp](#).

References [_offDate](#).

34.118.4.19 void stdair::SegmentDate::setOffTime (const Duration_T & *iOffTime*)
[inline]

Set the off time.

Definition at line 150 of file [SegmentDate.hpp](#).

References [_offTime](#).

34.118.4.20 void stdair::SegmentDate::setElapsedTime (const Duration_T & *iElapsedTime*)
[inline]

Set the elapsed time.

Definition at line 155 of file [SegmentDate.hpp](#).

References [_elapsedTime](#).

34.118.4.21 void stdair::SegmentDate::setDistance (const Distance_T & *iDistance*)
[inline]

Set the distance.

Definition at line 160 of file [SegmentDate.hpp](#).

References [_distance](#).

34.118.4.22 void stdair::SegmentDate::linkWithOperating (SegmentDate & *iSegmentDate*)
[inline]

Set operating segment date.

Definition at line 165 of file [SegmentDate.hpp](#).

References [_operatingSegmentDate](#).

```
34.118.4.23 void stdair::SegmentDate::toStream ( std::ostream & ioOut ) const [inline,
virtual]
```

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Implements [stdair::BomAbstract](#).

Definition at line 176 of file [SegmentDate.hpp](#).

References [toString\(\)](#).

```
34.118.4.24 void stdair::SegmentDate::fromStream ( std::istream & ioIn ) [inline,
virtual]
```

Read a Business Object from an input stream.

Parameters

<i>istream&</i> the input stream.

Implements [stdair::BomAbstract](#).

Definition at line 185 of file [SegmentDate.hpp](#).

```
34.118.4.25 std::string stdair::SegmentDate::toString ( ) const [virtual]
```

Get the serialised version of the Business Object.

Implements [stdair::BomAbstract](#).

Definition at line 41 of file [SegmentDate.cpp](#).

References [describeKey\(\)](#).

Referenced by [toStream\(\)](#).

```
34.118.4.26 const std::string stdair::SegmentDate::describeKey ( ) const [inline]
```

Get a string describing the key.

Definition at line 196 of file [SegmentDate.hpp](#).

References [_key](#), and [stdair::SegmentDateKey::toString\(\)](#).

Referenced by [stdair::SegmentCabin::getFullerKey\(\)](#), and [toString\(\)](#).

34.118.4.27 `template<class Archive > void stdair::SegmentDate::serialize (Archive & ar, const unsigned int iFileVersion)`

Serialisation.

Definition at line 208 of file [CmdBomSerialiser.cpp](#).

References [_key](#).

34.118.5 Friends And Related Function Documentation

34.118.5.1 `friend class FacBom [friend]`

Definition at line 34 of file [SegmentDate.hpp](#).

34.118.5.2 `friend class FacBomManager [friend]`

Definition at line 35 of file [SegmentDate.hpp](#).

34.118.5.3 `friend class boost::serialization::access [friend]`

Definition at line 36 of file [SegmentDate.hpp](#).

34.118.6 Member Data Documentation

34.118.6.1 `Key_T stdair::SegmentDate::_key [protected]`

Primary key (origin and destination).

Definition at line 250 of file [SegmentDate.hpp](#).

Referenced by [describeKey\(\)](#), [getBoardingPoint\(\)](#), [getKey\(\)](#), [getOffPoint\(\)](#), and [serialize\(\)](#).

34.118.6.2 `BomAbstract* stdair::SegmentDate::_parent [protected]`

Pointer on the parent class ([FlightDate](#)).

Definition at line 255 of file [SegmentDate.hpp](#).

Referenced by [getParent\(\)](#).

34.118.6.3 `HolderMap_T stdair::SegmentDate::_holderMap [protected]`

Map holding the children ([SegmentCabin](#) objects).

Definition at line 260 of file [SegmentDate.hpp](#).

Referenced by [getHolderMap\(\)](#).

34.118.6.4 SegmentDate* stdair::SegmentDate::_operatingSegmentDate [protected]

Pointer on the operating [SegmentDate](#). Nota: 1. "operating" refers to the codeshare contract seller. 2. the pointer will be NULL if the segment date is itself the "operating" one.

Definition at line 268 of file [SegmentDate.hpp](#).

Referenced by [getOperatingSegmentDate\(\)](#), and [linkWithOperating\(\)](#).

34.118.6.5 SegmentDateList_T stdair::SegmentDate::_marketingSegmentDateList [protected]

List holding the marketing segment dates. Nota: 1. "marketing" refers to the codeshare contract seller. 2. the list will be empty if the segment date is itself the "marketing" one.

Definition at line 276 of file [SegmentDate.hpp](#).

Referenced by [getMarketingSegmentDateList\(\)](#).

34.118.6.6 Date_T stdair::SegmentDate::_boardingDate [protected]

Boarding date.

Definition at line 281 of file [SegmentDate.hpp](#).

Referenced by [getBoardingDate\(\)](#), [getDateOffset\(\)](#), and [setBoardingDate\(\)](#).

34.118.6.7 Duration_T stdair::SegmentDate::_boardingTime [protected]

Boarding time.

Definition at line 286 of file [SegmentDate.hpp](#).

Referenced by [getBoardingTime\(\)](#), [getTimeOffset\(\)](#), and [setBoardingTime\(\)](#).

34.118.6.8 Date_T stdair::SegmentDate::_offDate [protected]

Landing date.

Definition at line 291 of file [SegmentDate.hpp](#).

Referenced by [getDateOffset\(\)](#), [getOffDate\(\)](#), and [setOffDate\(\)](#).

34.118.6.9 Duration_T stdair::SegmentDate::_offTime [protected]

Landing time.

Definition at line 296 of file [SegmentDate.hpp](#).

Referenced by [getOffTime\(\)](#), [getTimeOffset\(\)](#), and [setOffTime\(\)](#).

34.118.6.10 Duration_T stdair::SegmentDate::_elapsedTime [protected]

Trip elapsed time.

Definition at line 301 of file [SegmentDate.hpp](#).

Referenced by [getElapsedTime\(\)](#), [getTimeOffset\(\)](#), and [setElapsedTime\(\)](#).

34.118.6.11 **Distance_T** stdair::SegmentDate::_distance [protected]

Trip distance.

Definition at line 306 of file [SegmentDate.hpp](#).

Referenced by [getDistance\(\)](#), and [setDistance\(\)](#).

The documentation for this class was generated from the following files:

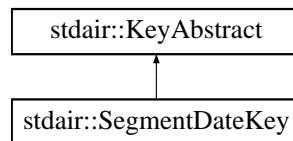
- [stdair/bom/SegmentDate.hpp](#)
- [stdair/bom/SegmentDate.cpp](#)
- [stdair/command/CmdBomSerialiser.cpp](#)

34.119 stdair::SegmentDateKey Struct Reference

Key of a given segment-date, made of an origin and a destination airports.

```
#include <stdair/bom/SegmentDateKey.hpp>
```

Inheritance diagram for stdair::SegmentDateKey:



Public Member Functions

- [SegmentDateKey](#) (const [AirportCode_T](#) &, const [AirportCode_T](#) &)
- [SegmentDateKey](#) (const [SegmentDateKey](#) &)
- [~SegmentDateKey](#) ()
- const [AirportCode_T](#) & [getBoardingPoint](#) () const
- const [AirportCode_T](#) & [getOffPoint](#) () const
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- const std::string [toString](#) () const
- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int iFileVersion)

Friends

- class [boost::serialization::access](#)

34.119.1 Detailed Description

Key of a given segment-date, made of an origin and a destination airports.

Definition at line 24 of file [SegmentDateKey.hpp](#).

34.119.2 Constructor & Destructor Documentation

34.119.2.1 stdair::SegmentDateKey::SegmentDateKey (const AirportCode_T & iBoardingPoint, const AirportCode_T & iOffPoint)

Main constructor.

Definition at line 25 of file [SegmentDateKey.cpp](#).

34.119.2.2 stdair::SegmentDateKey::SegmentDateKey (const SegmentDateKey & iKey)

Copy constructor.

Definition at line 31 of file [SegmentDateKey.cpp](#).

34.119.2.3 stdair::SegmentDateKey::~~SegmentDateKey ()

Destructor.

Definition at line 36 of file [SegmentDateKey.cpp](#).

34.119.3 Member Function Documentation

34.119.3.1 const AirportCode_T& stdair::SegmentDateKey::getBoardingPoint () const [inline]

Get the boarding point.

Definition at line 51 of file [SegmentDateKey.hpp](#).

Referenced by [stdair::SegmentDate::getBoardingPoint\(\)](#), and [stdair::OnDDateKey::getOrigin\(\)](#).

34.119.3.2 const AirportCode_T& stdair::SegmentDateKey::getOffPoint () const [inline]

Get the arrival point.

Definition at line 56 of file [SegmentDateKey.hpp](#).

Referenced by [stdair::OnDDateKey::getDestination\(\)](#), and [stdair::SegmentDate::getOffPoint\(\)](#).

34.119.3.3 void stdair::SegmentDateKey::toStream (std::ostream & ioOut) const [virtual]

Dump a Business Object Key into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 40 of file [SegmentDateKey.cpp](#).

References [toString\(\)](#).

34.119.3.4 `void stdair::SegmentDateKey::fromStream (std::istream & ioIn) [virtual]`

Read a Business Object Key from an input stream.

Parameters

<i>istream&</i> the input stream.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 45 of file [SegmentDateKey.cpp](#).

34.119.3.5 `const std::string stdair::SegmentDateKey::toString () const [virtual]`

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 49 of file [SegmentDateKey.cpp](#).

References [stdair::DEFAULT_KEY_SUB_FLD_DELIMITER](#).

Referenced by [stdair::SegmentDate::describeKey\(\)](#), [stdair::FlightDate::getSegmentDate\(\)](#), [stdair::BomRetriever::retrieveSegmentDateFromLongKey\(\)](#), and [toString\(\)](#).

34.119.3.6 `template<class Archive > void stdair::SegmentDateKey::serialize (Archive & ar, const unsigned int iFileVersion)`

Serialisation.

Definition at line 72 of file [SegmentDateKey.cpp](#).

34.119.4 Friends And Related Function Documentation

34.119.4.1 `friend class boost::serialization::access [friend]`

Definition at line 25 of file [SegmentDateKey.hpp](#).

The documentation for this struct was generated from the following files:

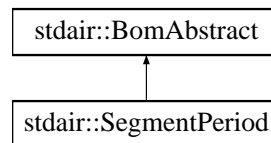
- [stdair/bom/SegmentDateKey.hpp](#)

- stdair/bom/[SegmentDateKey.cpp](#)

34.120 stdair::SegmentPeriod Class Reference

```
#include <stdair/bom/SegmentPeriod.hpp>
```

Inheritance diagram for stdair::SegmentPeriod:



Public Types

- typedef [SegmentPeriodKey](#) Key_T

Public Member Functions

- const [Key_T](#) & [getKey](#) () const
- [BomAbstract](#) *const [getParent](#) () const
- const [AirportCode_T](#) & [getBoardingPoint](#) () const
- const [AirportCode_T](#) & [getOffPoint](#) () const
- const [Duration_T](#) & [getBoardingTime](#) () const
- const [Duration_T](#) & [getOffTime](#) () const
- const [DateOffset_T](#) & [getBoardingDateOffset](#) () const
- const [DateOffset_T](#) & [getOffDateOffset](#) () const
- const [Duration_T](#) & [getElapsedTime](#) () const
- const [CabinBookingClassMap_T](#) & [getCabinBookingClassMap](#) () const
- const [HolderMap_T](#) & [getHolderMap](#) () const
- void [setBoardingTime](#) (const [Duration_T](#) &iBoardingTime)
- void [setOffTime](#) (const [Duration_T](#) &iOffTime)
- void [setBoardingDateOffset](#) (const [DateOffset_T](#) &iDateOffset)
- void [setOffDateOffset](#) (const [DateOffset_T](#) &iDateOffset)
- void [setElapsedTime](#) (const [Duration_T](#) &iElapsedTime)
- void [addCabinBookingClassList](#) (const [CabinCode_T](#) &, const [ClassList_String_T](#) &)
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- std::string [toString](#) () const
- const std::string [describeKey](#) () const

Protected Member Functions

- [SegmentPeriod](#) (const [Key_T](#) &)
- [SegmentPeriod](#) (const [SegmentPeriod](#) &)
- [~SegmentPeriod](#) ()

Protected Attributes

- [Key_T _key](#)
- [BomAbstract](#) * [_parent](#)
- [Duration_T _boardingTime](#)
- [Duration_T _offTime](#)
- [DateOffset_T _boardingDateOffset](#)
- [DateOffset_T _offDateOffset](#)
- [Duration_T _elapsedTime](#)
- [CabinBookingClassMap_T _cabinBookingClassMap](#)
- [HolderMap_T _holderMap](#)

Friends

- class [FacBom](#)
- class [FacBomManager](#)

34.120.1 Detailed Description

Class representing the actual attributes for an airline segment-period.

Definition at line 15 of file [SegmentPeriod.hpp](#).

34.120.2 Member Typedef Documentation

34.120.2.1 typedef SegmentPeriodKey stdair::SegmentPeriod::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 22 of file [SegmentPeriod.hpp](#).

34.120.3 Constructor & Destructor Documentation

34.120.3.1 stdair::SegmentPeriod::SegmentPeriod (const Key_T & iKey) [protected]

Default constructors.

Definition at line 13 of file [SegmentPeriod.cpp](#).

34.120.3.2 **stdair::SegmentPeriod::SegmentPeriod** (const **SegmentPeriod** &)
[protected]

34.120.3.3 **stdair::SegmentPeriod::~~SegmentPeriod** () [protected]

Destructor.

Definition at line 18 of file [SegmentPeriod.cpp](#).

34.120.4 Member Function Documentation

34.120.4.1 **const Key_T& stdair::SegmentPeriod::getKey** () const [inline]

Get the segment-period key.

Definition at line 27 of file [SegmentPeriod.hpp](#).

References [_key](#).

34.120.4.2 **BomAbstract* const stdair::SegmentPeriod::getParent** () const [inline]

Get the parent object.

Definition at line 30 of file [SegmentPeriod.hpp](#).

References [_parent](#).

34.120.4.3 **const AirportCode_T& stdair::SegmentPeriod::getBoardingPoint** () const
[inline]

Get the boarding point (part of the primary key).

Definition at line 33 of file [SegmentPeriod.hpp](#).

References [_key](#), and [stdair::SegmentPeriodKey::getBoardingPoint\(\)](#).

34.120.4.4 **const AirportCode_T& stdair::SegmentPeriod::getOffPoint** () const
[inline]

Get the off point (part of the primary key).

Definition at line 38 of file [SegmentPeriod.hpp](#).

References [_key](#), and [stdair::SegmentPeriodKey::getOffPoint\(\)](#).

34.120.4.5 **const Duration_T& stdair::SegmentPeriod::getBoardingTime** () const
[inline]

Get the boarding time.

Definition at line 41 of file [SegmentPeriod.hpp](#).

References [_boardingTime](#).

34.120.4.6 **const Duration_T& stdair::SegmentPeriod::getOffTime () const** `[inline]`

Get the off time.

Definition at line 44 of file [SegmentPeriod.hpp](#).

References [_offTime](#).

34.120.4.7 **const DateOffset_T& stdair::SegmentPeriod::getBoardingDateOffset () const**
`[inline]`

Get the boarding date offset.

Definition at line 47 of file [SegmentPeriod.hpp](#).

References [_boardingDateOffset](#).

34.120.4.8 **const DateOffset_T& stdair::SegmentPeriod::getOffDateOffset () const**
`[inline]`

Get the off date offset.

Definition at line 52 of file [SegmentPeriod.hpp](#).

References [_offDateOffset](#).

34.120.4.9 **const Duration_T& stdair::SegmentPeriod::getElapsedTime () const**
`[inline]`

Get the elapsed time.

Definition at line 55 of file [SegmentPeriod.hpp](#).

References [_elapsedTime](#).

34.120.4.10 **const CabinBookingClassMap_T& stdair::SegmentPeriod::getCabinBookingClassMap () const**
`[inline]`

Get the cabin booking class map.

Definition at line 58 of file [SegmentPeriod.hpp](#).

References [_cabinBookingClassMap](#).

34.120.4.11 **const HolderMap_T& stdair::SegmentPeriod::getHolderMap () const**
`[inline]`

Get the map of children holders.

Definition at line 63 of file [SegmentPeriod.hpp](#).

References [_holderMap](#).

34.120.4.12 **void stdair::SegmentPeriod::setBoardingTime (const Duration_T & iBoardingTime)** `[inline]`

Set the boarding time.

Definition at line 68 of file [SegmentPeriod.hpp](#).

References [_boardingTime](#).

```
34.120.4.13 void stdair::SegmentPeriod::setOffTime ( const Duration_T & iOffTime )
           [inline]
```

Set the off time.

Definition at line 73 of file [SegmentPeriod.hpp](#).

References [_offTime](#).

```
34.120.4.14 void stdair::SegmentPeriod::setBoardingDateOffset ( const DateOffset_T &
           iDateOffset ) [inline]
```

Set the boarding date offset.

Definition at line 76 of file [SegmentPeriod.hpp](#).

References [_boardingDateOffset](#).

```
34.120.4.15 void stdair::SegmentPeriod::setOffDateOffset ( const DateOffset_T & iDateOffset
           ) [inline]
```

Set the off date offset.

Definition at line 81 of file [SegmentPeriod.hpp](#).

References [_offDateOffset](#).

```
34.120.4.16 void stdair::SegmentPeriod::setElapsedTime ( const Duration_T & iElapsedTime
           ) [inline]
```

Set the elapsed time.

Definition at line 86 of file [SegmentPeriod.hpp](#).

References [_elapsedTime](#).

```
34.120.4.17 void stdair::SegmentPeriod::addCabinBookingClassList ( const CabinCode_T &
           iCabinCode, const ClassList_String_T & iClassCodeList )
```

Add a pair cabin code and list of class codes within the cabin to the cabin booking class map.

Definition at line 30 of file [SegmentPeriod.cpp](#).

References [_cabinBookingClassMap](#).

```
34.120.4.18 void stdair::SegmentPeriod::toStream ( std::ostream & ioOut ) const
           [inline, virtual]
```

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Implements [stdair::BomAbstract](#).

Definition at line 99 of file [SegmentPeriod.hpp](#).

References [toString\(\)](#).

34.120.4.19 `void stdair::SegmentPeriod::fromStream (std::istream & ioIn) [inline, virtual]`

Read a Business Object from an input stream.

Parameters

<i>istream&</i>	the input stream.
---------------------	-------------------

Implements [stdair::BomAbstract](#).

Definition at line 103 of file [SegmentPeriod.hpp](#).

34.120.4.20 `std::string stdair::SegmentPeriod::toString () const [virtual]`

Get the serialised version of the Business Object.

Implements [stdair::BomAbstract](#).

Definition at line 22 of file [SegmentPeriod.cpp](#).

References [describeKey\(\)](#).

Referenced by [toStream\(\)](#).

34.120.4.21 `const std::string stdair::SegmentPeriod::describeKey () const [inline]`

Get a string describing the key.

Definition at line 109 of file [SegmentPeriod.hpp](#).

References [_key](#), and [stdair::SegmentPeriodKey::toString\(\)](#).

Referenced by [toString\(\)](#).

34.120.5 Friends And Related Function Documentation

34.120.5.1 `friend class FacBom [friend]`

Definition at line 16 of file [SegmentPeriod.hpp](#).

34.120.5.2 `friend class FacBomManager [friend]`

Definition at line 17 of file [SegmentPeriod.hpp](#).

34.120.6 Member Data Documentation

34.120.6.1 Key_T stdair::SegmentPeriod::_key [protected]

Definition at line 120 of file [SegmentPeriod.hpp](#).

Referenced by [describeKey\(\)](#), [getBoardingPoint\(\)](#), [getKey\(\)](#), and [getOffPoint\(\)](#).

34.120.6.2 BomAbstract* stdair::SegmentPeriod::_parent [protected]

Definition at line 121 of file [SegmentPeriod.hpp](#).

Referenced by [getParent\(\)](#).

34.120.6.3 Duration_T stdair::SegmentPeriod::_boardingTime [protected]

Definition at line 122 of file [SegmentPeriod.hpp](#).

Referenced by [getBoardingTime\(\)](#), and [setBoardingTime\(\)](#).

34.120.6.4 Duration_T stdair::SegmentPeriod::_offTime [protected]

Definition at line 123 of file [SegmentPeriod.hpp](#).

Referenced by [getOffTime\(\)](#), and [setOffTime\(\)](#).

34.120.6.5 DateOffset_T stdair::SegmentPeriod::_boardingDateOffset
[protected]

Definition at line 124 of file [SegmentPeriod.hpp](#).

Referenced by [getBoardingDateOffset\(\)](#), and [setBoardingDateOffset\(\)](#).

34.120.6.6 DateOffset_T stdair::SegmentPeriod::_offDateOffset [protected]

Definition at line 125 of file [SegmentPeriod.hpp](#).

Referenced by [getOffDateOffset\(\)](#), and [setOffDateOffset\(\)](#).

34.120.6.7 Duration_T stdair::SegmentPeriod::_elapsedTime [protected]

Definition at line 126 of file [SegmentPeriod.hpp](#).

Referenced by [getElapsedTime\(\)](#), and [setElapsedTime\(\)](#).

34.120.6.8 CabinBookingClassMap_T stdair::SegmentPeriod::_cabinBookingClassMap [protected]

Definition at line 127 of file [SegmentPeriod.hpp](#).

Referenced by [addCabinBookingClassList\(\)](#), and [getCabinBookingClassMap\(\)](#).

34.120.6.9 HolderMap_T stdair::SegmentPeriod::_holderMap [protected]

Definition at line 128 of file [SegmentPeriod.hpp](#).

Referenced by [getHolderMap\(\)](#).

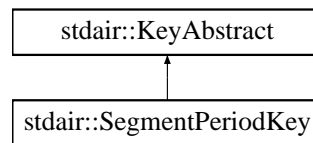
The documentation for this class was generated from the following files:

- [stdair/bom/SegmentPeriod.hpp](#)
- [stdair/bom/SegmentPeriod.cpp](#)

34.121 stdair::SegmentPeriodKey Struct Reference

```
#include <stdair/bom/SegmentPeriodKey.hpp>
```

Inheritance diagram for stdair::SegmentPeriodKey:



Public Member Functions

- [SegmentPeriodKey](#) (const [AirportCode_T](#) &, const [AirportCode_T](#) &)
- [SegmentPeriodKey](#) (const [SegmentPeriodKey](#) &)
- [~SegmentPeriodKey](#) ()
- const [AirportCode_T](#) & [getBoardingPoint](#) () const
- const [AirportCode_T](#) & [getOffPoint](#) () const
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- const std::string [toString](#) () const

34.121.1 Detailed Description

Key of segment-period.

Definition at line 14 of file [SegmentPeriodKey.hpp](#).

34.121.2 Constructor & Destructor Documentation

34.121.2.1 [stdair::SegmentPeriodKey::SegmentPeriodKey](#) (const [AirportCode_T](#) & *iBoardingPoint*, const [AirportCode_T](#) & *iOffPoint*)

Constructors.

Definition at line 12 of file [SegmentPeriodKey.cpp](#).

34.121.2.2 [stdair::SegmentPeriodKey::SegmentPeriodKey](#) (const [SegmentPeriodKey](#) & *iKey*)

Definition at line 18 of file [SegmentPeriodKey.cpp](#).

34.121.2.3 stdair::SegmentPeriodKey::~~SegmentPeriodKey ()

Destructor.

Definition at line 23 of file [SegmentPeriodKey.cpp](#).

34.121.3 Member Function Documentation**34.121.3.1 const AirportCode_T& stdair::SegmentPeriodKey::getBoardingPoint () const**
[inline]

Get the boarding point.

Definition at line 29 of file [SegmentPeriodKey.hpp](#).

Referenced by [stdair::SegmentPeriod::getBoardingPoint\(\)](#).

34.121.3.2 const AirportCode_T& stdair::SegmentPeriodKey::getOffPoint () const
[inline]

Get the arrival point.

Definition at line 34 of file [SegmentPeriodKey.hpp](#).

Referenced by [stdair::SegmentPeriod::getOffPoint\(\)](#).

34.121.3.3 void stdair::SegmentPeriodKey::toStream (std::ostream & ioOut) const
[virtual]

Dump a Business Object Key into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 27 of file [SegmentPeriodKey.cpp](#).

References [toString\(\)](#).

34.121.3.4 void stdair::SegmentPeriodKey::fromStream (std::istream & ioIn) [virtual]

Read a Business Object Key from an input stream.

Parameters

<i>istream&</i> the input stream.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 32 of file [SegmentPeriodKey.cpp](#).

34.121.3.5 const std::string stdair::SegmentPeriodKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-period.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 36 of file [SegmentPeriodKey.cpp](#).

Referenced by [stdair::SegmentPeriod::describeKey\(\)](#), and [toStream\(\)](#).

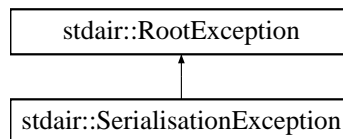
The documentation for this struct was generated from the following files:

- [stdair/bom/SegmentPeriodKey.hpp](#)
- [stdair/bom/SegmentPeriodKey.cpp](#)

34.122 stdair::SerialisationException Class Reference

```
#include <stdair/stdair_exceptions.hpp>
```

Inheritance diagram for stdair::SerialisationException:



Public Member Functions

- [SerialisationException](#) (const std::string &iWhat)
- const char * [what](#) () const throw ()

Protected Attributes

- std::string [_what](#)

34.122.1 Detailed Description

Serialisation.

Definition at line 119 of file [stdair_exceptions.hpp](#).

34.122.2 Constructor & Destructor Documentation

34.122.2.1 stdair::SerialisationException::SerialisationException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 122 of file [stdair_exceptions.hpp](#).

34.122.3 Member Function Documentation

34.122.3.1 const char* stdair::RootException::what () const throw () [inline, inherited]

Give the details of the exception.

Definition at line 38 of file [stdair_exceptions.hpp](#).

References [stdair::RootException::_what](#).

34.122.4 Member Data Documentation

34.122.4.1 std::string stdair::RootException::_what [protected, inherited]

Details for the exception.

Definition at line 46 of file [stdair_exceptions.hpp](#).

Referenced by [stdair::RootException::what\(\)](#).

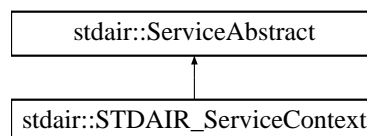
The documentation for this class was generated from the following file:

- [stdair/stdair_exceptions.hpp](#)

34.123 stdair::ServiceAbstract Class Reference

```
#include <stdair/service/ServiceAbstract.hpp>
```

Inheritance diagram for stdair::ServiceAbstract:



Public Member Functions

- virtual [~ServiceAbstract](#) ()

- virtual void [toStream](#) (std::ostream &ioOut) const
- virtual void [fromStream](#) (std::istream &ioIn)

Protected Member Functions

- [ServiceAbstract](#) ()

34.123.1 Detailed Description

Base class for the Service layer.

Definition at line 15 of file [ServiceAbstract.hpp](#).

34.123.2 Constructor & Destructor Documentation

34.123.2.1 virtual stdair::ServiceAbstract::~ServiceAbstract () [inline, virtual]

Destructor.

Definition at line 21 of file [ServiceAbstract.hpp](#).

34.123.2.2 stdair::ServiceAbstract::ServiceAbstract () [inline, protected]

Display of the structure. Protected Default Constructor to ensure this class is abstract.

Definition at line 46 of file [ServiceAbstract.hpp](#).

34.123.3 Member Function Documentation

34.123.3.1 virtual void stdair::ServiceAbstract::toStream (std::ostream & *ioOut*) const
[inline, virtual]

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i>	the output stream.
---------------------	--------------------

Definition at line 28 of file [ServiceAbstract.hpp](#).

Referenced by [operator<<\(\)](#).

34.123.3.2 virtual void stdair::ServiceAbstract::fromStream (std::istream & *ioIn*)
[inline, virtual]

Read a Business Object from an input stream.

Parameters

<i>istream&</i>	the input stream.
---------------------	-------------------

Definition at line 35 of file [ServiceAbstract.hpp](#).

Referenced by [operator>>\(\)](#).

The documentation for this class was generated from the following file:

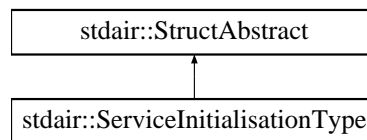
- [stdair/service/ServiceAbstract.hpp](#)

34.124 stdair::ServiceInitialisationType Struct Reference

Enumeration of service initialisation types.

```
#include <stdair/basic/ServiceInitialisationType.hpp>
```

Inheritance diagram for stdair::ServiceInitialisationType:



Public Types

- enum [EN_ServiceInitialisationType](#) { [NOT_YET_INITIALISED](#) = 0, [FILE_PARSING](#), [BUILTIN_SAMPLE](#), [LAST_VALUE](#) }

Public Member Functions

- [EN_ServiceInitialisationType](#) [getType](#) () const
- char [getTypeAsChar](#) () const
- std::string [getTypeAsString](#) () const
- const std::string [describe](#) () const
- bool [operator==](#) (const [EN_ServiceInitialisationType](#) &) const
- [ServiceInitialisationType](#) (const [EN_ServiceInitialisationType](#) &)
- [ServiceInitialisationType](#) (const char iType)
- [ServiceInitialisationType](#) (const std::string &iType)
- [ServiceInitialisationType](#) (const [ServiceInitialisationType](#) &)
- void [toStream](#) (std::ostream &ioOut) const
- virtual void [fromStream](#) (std::istream &ioIn)

Static Public Member Functions

- static const std::string & [getLabel](#) (const [EN_ServiceInitialisationType](#) &)
- static [EN_ServiceInitialisationType](#) [getType](#) (const char)
- static char [getTypeLabel](#) (const [EN_ServiceInitialisationType](#) &)
- static std::string [getTypeLabelAsString](#) (const [EN_ServiceInitialisationType](#) &)
- static std::string [describeLabels](#) ()

34.124.1 Detailed Description

Enumeration of service initialisation types.

Definition at line 17 of file [ServiceInitialisationType.hpp](#).

34.124.2 Member Enumeration Documentation

34.124.2.1 enum stdair::ServiceInitialisationType::EN_ServiceInitialisationType

Enumerator:

NOT_YET_INITIALISED

FILE_PARSING

BUILTIN_SAMPLE

LAST_VALUE

Definition at line 19 of file [ServiceInitialisationType.hpp](#).

34.124.3 Constructor & Destructor Documentation

34.124.3.1 stdair::ServiceInitialisationType::ServiceInitialisationType (const EN_ServiceInitialisationType & iServiceInitialisationType)

Main constructor.

Definition at line 36 of file [ServiceInitialisationType.cpp](#).

34.124.3.2 stdair::ServiceInitialisationType::ServiceInitialisationType (const char iType)

Alternative constructor.

Definition at line 65 of file [ServiceInitialisationType.cpp](#).

34.124.3.3 stdair::ServiceInitialisationType::ServiceInitialisationType (const std::string & iType)

Alternative constructor.

Definition at line 71 of file [ServiceInitialisationType.cpp](#).

References [getType\(\)](#).

34.124.3.4 stdair::ServiceInitialisationType::ServiceInitialisationType (const ServiceInitialisationType & iServiceInitialisationType)

Default copy constructor.

Definition at line 30 of file [ServiceInitialisationType.cpp](#).

34.124.4 Member Function Documentation

34.124.4.1 `const std::string & stdair::ServiceInitialisationType::getLabel (const EN_ServiceInitialisationType & iType) [static]`

Get the label as a string (e.g., "Not yet initialised", "File parsing" or "Built-in sample BOM").

Definition at line 81 of file [ServiceInitialisationType.cpp](#).

34.124.4.2 `ServiceInitialisationType::EN_ServiceInitialisationType stdair::ServiceInitialisationType::getType (const char iTypeChar) [static]`

Get the type value from parsing a single char (e.g., 'N', 'F', 'B').

Definition at line 42 of file [ServiceInitialisationType.cpp](#).

References [BUILTIN_SAMPLE](#), [describeLabels\(\)](#), [FILE_PARSING](#), [LAST_VALUE](#), and [NOT_YET_INITIALISED](#).

34.124.4.3 `char stdair::ServiceInitialisationType::getTypeLabel (const EN_ServiceInitialisationType & iType) [static]`

Get the label as a single char (e.g., 'N', 'F', 'B').

Definition at line 87 of file [ServiceInitialisationType.cpp](#).

34.124.4.4 `std::string stdair::ServiceInitialisationType::getTypeLabelAsString (const EN_ServiceInitialisationType & iType) [static]`

Get the label as a string of a single char (e.g., "N", "F", "B").

Definition at line 93 of file [ServiceInitialisationType.cpp](#).

34.124.4.5 `std::string stdair::ServiceInitialisationType::describeLabels () [static]`

List the labels.

Definition at line 100 of file [ServiceInitialisationType.cpp](#).

References [LAST_VALUE](#).

Referenced by [getType\(\)](#).

34.124.4.6 `ServiceInitialisationType::EN_ServiceInitialisationType stdair::ServiceInitialisationType::getType () const`

Get the enumerated value.

Definition at line 113 of file [ServiceInitialisationType.cpp](#).

Referenced by [ServiceInitialisationType\(\)](#).

34.124.4.7 `char stdair::ServiceInitialisationType::getTypeAsChar () const`

Get the enumerated value as a short string (e.g., 'N', 'F', 'B').

Definition at line 118 of file [ServiceInitialisationType.cpp](#).

34.124.4.8 `std::string stdair::ServiceInitialisationType::getTypeAsString () const`

Get the enumerated value as a short string (e.g., "N", "F", "B").

Definition at line 124 of file [ServiceInitialisationType.cpp](#).

34.124.4.9 `const std::string stdair::ServiceInitialisationType::describe () const`
`[virtual]`

Give a description of the structure (e.g., "Not yet initialised", "File parsing" or "Built-in sample BOM").

Implements [stdair::StructAbstract](#).

Definition at line 131 of file [ServiceInitialisationType.cpp](#).

34.124.4.10 `bool stdair::ServiceInitialisationType::operator== (const`
`EN_ServiceInitialisationType & iType) const`

Comparison operator.

Definition at line 139 of file [ServiceInitialisationType.cpp](#).

34.124.4.11 `void stdair::StructAbstract::toStream (std::ostream & ioOut) const` `[inline,`
`inherited]`

Dump a Business Object into an output stream.

Parameters

<code>ostream&</code> the output stream.
--

Reimplemented in [stdair::YieldRange](#), [stdair::AirlineStruct](#), [stdair::BookingRequestStruct](#), [stdair::CancellationStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::TravelSolutionStruct](#), and [stdair::VirtualClassStruct](#).

Definition at line 29 of file [StructAbstract.hpp](#).

References [stdair::StructAbstract::describe\(\)](#).

Referenced by [operator<<\(\)](#).

34.124.4.12 `virtual void stdair::StructAbstract::fromStream (std::istream & ioIn)`
`[inline, virtual, inherited]`

Read a Business Object from an input stream.

Parameters

<code>istream&</code> the input stream.

Reimplemented in [stdair::ProgressStatusSet](#), [stdair::YieldRange](#), [stdair::AirlineStruct](#), [stdair::BookingRequestStruct](#), [stdair::CancellationStruct](#), [stdair::EventStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::TravelSolutionStruct](#), and [stdair::VirtualClassStruct](#).

Definition at line 38 of file [StructAbstract.hpp](#).

Referenced by [operator>>\(\)](#).

The documentation for this struct was generated from the following files:

- stdair/basic/[ServiceInitialisationType.hpp](#)
- stdair/basic/[ServiceInitialisationType.cpp](#)

34.125 swift::SKeymap Class Reference

The readline keymap wrapper.

```
#include <stdair/ui/cmdline/SReadline.hpp>
```

Public Member Functions

- [SKeymap](#) (bool PrintableBound=false)
Creates a new keymap.
- [SKeymap](#) (Keymap Pattern)
Creates a new keymap which is a copy of Pattern.
- [~SKeymap](#) ()
Frees the allocated keymap.
- void [Bind](#) (int Key, KeyCallback Callback)
Binds the given key to a function.
- void [Unbind](#) (int Key)
Unbinds the given key.
- [SKeymap](#) (const [SKeymap](#) &rhs)
Copy constructor.
- [SKeymap](#) & [operator=](#) (const [SKeymap](#) &rhs)
operator=

Friends

- class [SReadline](#)

34.125.1 Detailed Description

The readline keymap wrapper.

Attention: It is not thread safe! Supports: key binding, key unbinding

Definition at line 307 of file [SReadline.hpp](#).

34.125.2 Constructor & Destructor Documentation

34.125.2.1 `swift::SKeymap::SKeymap (bool PrintableBound = false) [inline, explicit]`

Creates a new keymap.

Parameters

<i>Printable-Bound</i>	if true - the printable characters are bound if false - the keymap is empty
------------------------	---

Definition at line 319 of file [SReadline.hpp](#).

34.125.2.2 `swift::SKeymap::SKeymap (Keymap Pattern) [inline, explicit]`

Creates a new keymap which is a copy of Pattern.

Parameters

<i>Pattern</i>	A keymap to be copied.
----------------	------------------------

Definition at line 342 of file [SReadline.hpp](#).

34.125.2.3 `swift::SKeymap::~~SKeymap () [inline]`

Frees the allocated keymap.

Definition at line 354 of file [SReadline.hpp](#).

34.125.2.4 `swift::SKeymap::SKeymap (const SKeymap & rhs) [inline]`

Copy constructor.

Parameters

<i>rhs</i>	Right hand side object of SKeymap
------------	---

Definition at line 395 of file [SReadline.hpp](#).

34.125.3 Member Function Documentation

34.125.3.1 `void swift::SKeymap::Bind (int Key, KeyCallback Callback) [inline]`

Binds the given key to a function.

Parameters

<i>Key</i>	A key to be bound
<i>Callback</i>	A function to be called when the Key is pressed

Definition at line 366 of file [SReadline.hpp](#).

34.125.3.2 void swift::SKeymap::Unbind (int Key) [inline]

Unbinds the given key.

Parameters

<i>Key</i> A key to be unbound

Definition at line 381 of file [SReadline.hpp](#).

34.125.3.3 SKeymap& swift::SKeymap::operator= (const SKeymap & rhs) [inline]

operator=

Parameters

<i>rhs</i> Right hand side object of SKeymap
--

Definition at line 407 of file [SReadline.hpp](#).

34.125.4 Friends And Related Function Documentation

34.125.4.1 friend class SReadline [friend]

Definition at line 415 of file [SReadline.hpp](#).

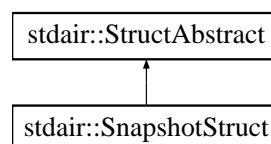
The documentation for this class was generated from the following file:

- [stdair/ui/cmdline/SReadline.hpp](#)

34.126 stdair::SnapshotStruct Struct Reference

```
#include <stdair/bom/SnapshotStruct.hpp>
```

Inheritance diagram for stdair::SnapshotStruct:



Public Member Functions

- const [AirlineCode_T](#) & getAirlineCode () const
- const [DateTime_T](#) & getSnapshotTime () const

- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- const std::string [describe](#) () const
- [SnapshotStruct](#) (const [AirlineCode_T](#) &, const [DateTime_T](#) &)
- [SnapshotStruct](#) (const [SnapshotStruct](#) &)
- [~SnapshotStruct](#) ()

34.126.1 Detailed Description

Structure holding the elements of a snapshot .

Definition at line 19 of file [SnapshotStruct.hpp](#).

34.126.2 Constructor & Destructor Documentation

34.126.2.1 **stdair::SnapshotStruct::SnapshotStruct** (const [AirlineCode_T](#) & *iAirlineCode*,
const [DateTime_T](#) & *iSnapshotTime*)

Constructor.

Definition at line 26 of file [SnapshotStruct.cpp](#).

34.126.2.2 **stdair::SnapshotStruct::SnapshotStruct** (const [SnapshotStruct](#) & *iSnapshot*)

Copy constructor.

Definition at line 19 of file [SnapshotStruct.cpp](#).

34.126.2.3 **stdair::SnapshotStruct::~~SnapshotStruct** ()

Destructor.

Definition at line 32 of file [SnapshotStruct.cpp](#).

34.126.3 Member Function Documentation

34.126.3.1 **const [AirlineCode_T](#)& stdair::SnapshotStruct::getAirlineCode** () const
[inline]

Get the airline code.

Definition at line 23 of file [SnapshotStruct.hpp](#).

34.126.3.2 **const [DateTime_T](#)& stdair::SnapshotStruct::getSnapshotTime** () const
[inline]

Get the snapshot action time.

Definition at line 28 of file [SnapshotStruct.hpp](#).

34.127 stdair::SQLDatabaseConnectionImpossibleException Class Reference

34.126.3.3 void stdair::SnapshotStruct::toStream (std::ostream & *ioOut*) const

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Reimplemented from [stdair::StructAbstract](#).

Definition at line 36 of file [SnapshotStruct.cpp](#).

References [describe\(\)](#).

34.126.3.4 void stdair::SnapshotStruct::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object from an input stream.

Parameters

<i>istream&</i> the input stream.

Reimplemented from [stdair::StructAbstract](#).

Definition at line 41 of file [SnapshotStruct.cpp](#).

34.126.3.5 const std::string stdair::SnapshotStruct::describe () const [virtual]

Display of the structure.

Implements [stdair::StructAbstract](#).

Definition at line 45 of file [SnapshotStruct.cpp](#).

Referenced by [toStream\(\)](#).

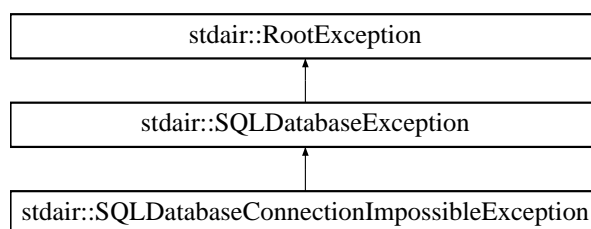
The documentation for this struct was generated from the following files:

- [stdair/bom/SnapshotStruct.hpp](#)
- [stdair/bom/SnapshotStruct.cpp](#)

34.127 stdair::SQLDatabaseConnectionImpossibleException Class Reference

```
#include <stdair/stdair_exceptions.hpp>
```

Inheritance diagram for stdair::SQLDatabaseConnectionImpossibleException:



Public Member Functions

- [SQLDatabaseConnectionImpossibleException](#) (const std::string &iWhat)
- const char * [what](#) () const throw ()

Protected Attributes

- std::string [_what](#)

34.127.1 Detailed Description

Database connection.

Definition at line 188 of file [stdair_exceptions.hpp](#).

34.127.2 Constructor & Destructor Documentation

34.127.2.1 **stdair::SQLDatabaseConnectionImpossibleException::SQLDatabaseConnectionImpossibleException**
(const std::string & *iWhat*) `[inline]`

Constructor.

Definition at line 191 of file [stdair_exceptions.hpp](#).

34.127.3 Member Function Documentation

34.127.3.1 **const char*** **stdair::RootException::what** () const throw () `[inline, inherited]`

Give the details of the exception.

Definition at line 38 of file [stdair_exceptions.hpp](#).

References [stdair::RootException::_what](#).

34.127.4 Member Data Documentation

34.127.4.1 **std::string** **stdair::RootException::_what** `[protected, inherited]`

Details for the exception.

Definition at line 46 of file [stdair_exceptions.hpp](#).

Referenced by [stdair::RootException::what\(\)](#).

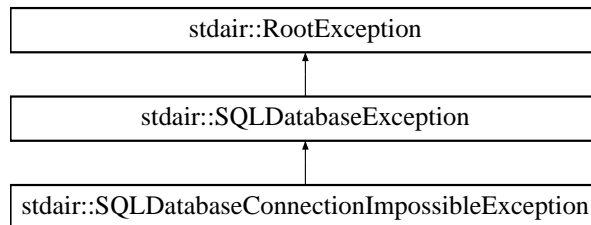
The documentation for this class was generated from the following file:

- [stdair/stdair_exceptions.hpp](#)

34.128 stdair::SQLDatabaseException Class Reference

```
#include <stdair/stdair_exceptions.hpp>
```

Inheritance diagram for stdair::SQLDatabaseException:



Public Member Functions

- [SQLDatabaseException](#) (const std::string &iWhat)
- const char * [what](#) () const throw ()

Protected Attributes

- std::string [_what](#)

34.128.1 Detailed Description

Database.

Definition at line 173 of file [stdair_exceptions.hpp](#).

34.128.2 Constructor & Destructor Documentation

34.128.2.1 [stdair::SQLDatabaseException::SQLDatabaseException \(const std::string & iWhat \)](#)
[inline]

Constructor.

Definition at line 176 of file [stdair_exceptions.hpp](#).

34.128.3 Member Function Documentation

34.128.3.1 [const char* stdair::RootException::what \(\) const throw \(\)](#) [inline, inherited]

Give the details of the exception.

Definition at line 38 of file [stdair_exceptions.hpp](#).

References [stdair::RootException::_what](#).

34.128.4 Member Data Documentation

34.128.4.1 std::string stdair::RootException::_what [protected, inherited]

Details for the exception.

Definition at line 46 of file [stdair_exceptions.hpp](#).

Referenced by [stdair::RootException::what\(\)](#).

The documentation for this class was generated from the following file:

- [stdair/stdair_exceptions.hpp](#)

34.129 swift::SReadline Class Reference

The readline library wrapper.

```
#include <stdair/ui/cmdline/SReadline.hpp>
```

Public Member Functions

- [SReadline](#) (const size_t Limit=DefaultHistoryLimit)
Constructs the object, sets the completion function.
- [SReadline](#) (const std::string &historyFileName, const size_t Limit=DefaultHistoryLimit)
Constructs the object, sets the completion function, loads history.
- [~SReadline](#) ()
Saves the session history (if the file name was provided) and destroys the object.
- std::string [GetLine](#) (const std::string &Prompt)
Gets a single line from a user.
- template<typename Container >
std::string [GetLine](#) (const std::string &Prompt, Container &ReadTokens)
Gets a single line from a user.
- template<typename Container >
std::string [GetLine](#) (const std::string &Prompt, Container &ReadTokens, bool &BreakOut)
Gets a single line from a user.
- std::string [GetLine](#) (const std::string &Prompt, bool &BreakOut)
Gets a single line from a user.
- template<typename ContainerType >
void [GetHistory](#) (ContainerType &Container)
Fills the given container with the current history list.
- bool [SaveHistory](#) (std::ostream &OS)
Saves the history to the given file stream.
- bool [SaveHistory](#) (const std::string &FileName)

Saves the history to the given file.

- void [ClearHistory](#) ()

Clears the history. Does not affect the file where the previous session history is saved.

- bool [LoadHistory](#) (std::istream &IS)

Loads a history from a file stream.

- bool [LoadHistory](#) (const std::string &FileName)

Loads a history from the given file.

- template<typename ContainerType >
void [RegisterCompletions](#) (const ContainerType &Container)

Allows to register custom completers.

- void [SetKeymap](#) (SKeymap &NewKeymap)

Sets the given keymap.

34.129.1 Detailed Description

The readline library wrapper.

Attention: It is not thread safe! Supports: editing, history, custom completers

Definition at line 424 of file [SReadline.hpp](#).

34.129.2 Constructor & Destructor Documentation

34.129.2.1 **swift::SReadline::SReadline** (const size_t *Limit* = DefaultHistoryLimit)
[inline]

Constructs the object, sets the completion function.

Parameters

<i>Limit</i>	History size
--------------	--------------

Definition at line 431 of file [SReadline.hpp](#).

34.129.2.2 **swift::SReadline::SReadline** (const std::string & *historyFileName*, const size_t *Limit* = DefaultHistoryLimit) [inline]

Constructs the object, sets the completion function, loads history.

Parameters

<i>historyFile- Name</i>	File name to load history from
------------------------------	--------------------------------

<i>Limit</i>	History size
--------------	--------------

Definition at line 446 of file [SReadline.hpp](#).

References [LoadHistory\(\)](#).

34.129.2.3 swift::SReadline::~~SReadline () [inline]

Saves the session history (if the file name was provided) and destroys the object.

Definition at line 462 of file [SReadline.hpp](#).

References [SaveHistory\(\)](#).

34.129.3 Member Function Documentation

34.129.3.1 std::string swift::SReadline::GetLine (const std::string & *Prompt*) [inline]

Gets a single line from a user.

Parameters

<i>Prompt</i>	A printed prompt
---------------	------------------

Returns

A string which was actually inputed

Definition at line 473 of file [SReadline.hpp](#).

Referenced by [GetLine\(\)](#).

34.129.3.2 template<typename Container > std::string swift::SReadline::GetLine (const std::string & *Prompt*, Container & *ReadTokens*) [inline]

Gets a single line from a user.

Parameters

<i>Prompt</i>	A printed prompt
<i>ReadTokens</i>	A user inputed string splitted into tokens. The container is cleared first

Returns

A string which was actually inputed

Definition at line 487 of file [SReadline.hpp](#).

References [GetLine\(\)](#).

34.129.3.3 template<typename Container > std::string swift::SReadline::GetLine (const std::string & *Prompt*, Container & *ReadTokens*, bool & *BreakOut*) [inline]

Gets a single line from a user.

Parameters

<i>Prompt</i>	A printed prompt
<i>BreakOut</i>	it is set to true if the EOF found
<i>ReadTokens</i>	A user inputed string splitted into tokens. The container is cleared first

Returns

A string which was actually inputed

Definition at line 502 of file [SReadline.hpp](#).

References [GetLine\(\)](#).

34.129.3.4 `std::string swift::SReadline::GetLine (const std::string & Prompt, bool & BreakOut)`
`[inline]`

Gets a single line from a user.

Parameters

<i>Prompt</i>	A printed prompt
<i>BreakOut</i>	it is set to true if the EOF found

Returns

A string which was actually inputed

Definition at line 517 of file [SReadline.hpp](#).

34.129.3.5 `template<typename ContainerType > void swift::SReadline::GetHistory (`
`ContainerType & Container) [inline]`

Fills the given container with the current history list.

Does not clear the given container

Definition at line 552 of file [SReadline.hpp](#).

34.129.3.6 `bool swift::SReadline::SaveHistory (std::ostream & OS) [inline]`

Saves the history to the given file stream.

Parameters

<i>OS</i>	output file stream
-----------	--------------------

Returns

true if success

Definition at line 564 of file [SReadline.hpp](#).

Referenced by [SaveHistory\(\)](#), and [~SReadline\(\)](#).

34.129.3.7 `bool swift::SReadline::SaveHistory (const std::string & FileName) [inline]`

Saves the history to the given file.

Parameters

<i>FileName</i>	File name to save the history to
-----------------	----------------------------------

Returns

true if success

Definition at line 581 of file [SReadline.hpp](#).

References [SaveHistory\(\)](#).

34.129.3.8 void swift::SReadline::ClearHistory () [inline]

Clears the history. Does not affect the file where the previous session history is saved.

Definition at line 594 of file [SReadline.hpp](#).

Referenced by [LoadHistory\(\)](#).

34.129.3.9 bool swift::SReadline::LoadHistory (std::istream & *IS*) [inline]

Loads a history from a file stream.

Parameters

<i>IS</i> Input file stream

Returns

true if success

Definition at line 604 of file [SReadline.hpp](#).

References [ClearHistory\(\)](#).

Referenced by [LoadHistory\(\)](#), and [SReadline\(\)](#).

34.129.3.10 bool swift::SReadline::LoadHistory (const std::string & *FileName*) [inline]

Loads a history from the given file.

Parameters

<i>FileName</i> File name to be load from

Returns

true if success

Definition at line 629 of file [SReadline.hpp](#).

References [LoadHistory\(\)](#).

34.129.3.11 template<typename ContainerType > void swift::SReadline::RegisterCompletions (const ContainerType & *Container*) [inline]

Allows to register custom completers.

Supports a special keyword: file. It means to use the standard file name completer.

For example the given container elements could be as follows:

- command1 opt1
- command1 opt2 file
- command2
- command2 opt1

Each container element must describe a single possible command line. The container element must have a conversion to std::string operator.

Parameters

<i>Container</i>	A container which has all the user possible commands.
------------------	---

Definition at line 658 of file [SReadline.hpp](#).

34.129.3.12 void swift::SReadline::SetKeymap (SKeymap & NewKeymap) [inline]

Sets the given keymap.

Parameters

<i>NewKeymap</i>	The keymap that should be used from now.
------------------	--

Definition at line 675 of file [SReadline.hpp](#).

The documentation for this class was generated from the following file:

- stdair/ui/cmdline/[SReadline.hpp](#)

34.130 stdair::STDAIR_Service Class Reference

Interface for the STDAIR Services.

```
#include <stdair/STDAIR_Service.hpp>
```

Public Member Functions

- [STDAIR_Service](#) ()
Default constructor.
- [STDAIR_Service](#) (const [BasLogParams](#) &)
Constructor.
- [STDAIR_Service](#) (const [BasLogParams](#) &, const [BasDBParams](#) &)
Constructor.
- [~STDAIR_Service](#) ()
Destructor.

- void [buildSampleBom](#) ()
- void [buildDummyInventory](#) (const [CabinCapacity_T](#) &iCabinCapacity)
- void [buildSampleTravelSolutionForPricing](#) ([TravelSolutionList_T](#) &)
- void [buildSampleTravelSolutions](#) ([TravelSolutionList_T](#) &)
- [BookingRequestStruct](#) [buildSampleBookingRequest](#) (const bool isForCRS=false)
- const [Count_T](#) & [getExpectedTotalNumberOfEventsToBeGenerated](#) () const
- const [Count_T](#) & [getExpectedTotalNumberOfEventsToBeGenerated](#) (const [EventType::EN_](#) - [EventType](#) &) const
- const [Count_T](#) & [getActualTotalNumberOfEventsToBeGenerated](#) () const
- const [Count_T](#) & [getActualTotalNumberOfEventsToBeGenerated](#) (const [EventType::EN_](#) - [EventType](#) &) const
- [ProgressStatusSet](#) [popEvent](#) ([EventStruct](#) &) const
- bool [isQueueDone](#) () const
- void [reset](#) () const
- std::string [jsonExport](#) (const [AirlineCode_T](#) &, const [FlightNumber_T](#) &, const [Date_T](#) &iDepartureDate) const
- std::string [list](#) (const [AirlineCode_T](#) &iAirlineCode="all", const [FlightNumber_T](#) &iFlightNumber=0) const
- std::string [listAirportPairDateRange](#) () const
- bool [check](#) (const [AirlineCode_T](#) &, const [FlightNumber_T](#) &, const [Date_T](#) &iDepartureDate) const
- bool [check](#) (const [AirportCode_T](#) &, const [AirportCode_T](#) &, const [Date_T](#) &iDepartureDate) const
- std::string [csvDisplay](#) () const
- std::string [csvDisplay](#) (const [AirlineCode_T](#) &, const [FlightNumber_T](#) &, const [Date_T](#) &iDepartureDate) const
- std::string [csvDisplay](#) (const [TravelSolutionList_T](#) &) const
- std::string [csvDisplay](#) (const [AirportCode_T](#) &, const [AirportCode_T](#) &, const [Date_T](#) &iDepartureDate) const
- [BomRoot](#) & [getBomRoot](#) () const
 - Get a reference on the [BomRoot](#) object.*
- [EventQueue](#) & [getEventQueue](#) () const
 - Get a reference on the [EventQueue](#) object.*
- [BasLogParams](#) [getLogParams](#) () const
- const [BasDBParams](#) & [getDBParams](#) () const
- const [ServiceInitialisationType](#) & [getServiceInitialisationType](#) () const

34.130.1 Detailed Description

Interface for the STDAIR Services.

Definition at line 43 of file [STDAIR_Service.hpp](#).

34.130.2 Constructor & Destructor Documentation

34.130.2.1 stdair::STDAIR_Service::STDAIR_Service ()

Default constructor.

Definition at line 43 of file [STDAIR_Service.cpp](#).

34.130.2.2 stdair::STDAIR_Service::STDAIR_Service (const BasLogParams & iLogParams)

Constructor.

The init() method is called; see the corresponding documentation for more details.

Moreover, a reference on an output stream is given, so that log outputs can be directed onto that stream.

Parameters

in	const	BasLogParams & Parameters for the output log stream.
----	-------	--

Definition at line 59 of file [STDAIR_Service.cpp](#).

34.130.2.3 stdair::STDAIR_Service::STDAIR_Service (const BasLogParams & iLogParams, const BasDBParams & iDBParams)

Constructor.

The init() method is called; see the corresponding documentation for more details.

A reference on an output stream is given, so that log outputs can be directed onto that stream.

Moreover, database connection parameters are given, so that database events can use the corresponding access.

Parameters

in	const	BasLogParams & Parameters for the output log stream.
in	const	BasDBParams & Parameters for the database session.

Definition at line 73 of file [STDAIR_Service.cpp](#).

34.130.2.4 stdair::STDAIR_Service::~~STDAIR_Service ()

Destructor.

Definition at line 91 of file [STDAIR_Service.cpp](#).

34.130.3 Member Function Documentation

34.130.3.1 void stdair::STDAIR_Service::buildSampleBom ()

Build a sample BOM tree, and attach it to the [BomRoot](#) instance.

As for now, a single sample BOM tree is built, with objects for all the simulator-related components, i.e.:

- schedule (e.g., AirSched),
- inventory (e.g., AirInv),
- revenue management (e.g., RMOL),
- pricing (e.g., SimFQT),
- revenue accounting (e.g., AirRAC),
- demand generation (e.g., TraDemGen),
- customer choice (e.g., TravelCCM),
- event manager (e.g., SEvMgr)

Most of the inventories just contain one flight. One of those flights has two legs (and therefore three segments).

Definition at line 164 of file [STDAIR_Service.cpp](#).

34.130.3.2 void stdair::STDAIR_Service::buildDummyInventory (const CabinCapacity_T & iCabinCapacity)

Build a dummy inventory, containing a dummy flight-date with a single leg-cabin and some virtual booking classes. That structure is the bare minimum required to perform an optimisation on a leg-cabin.

As for now, that method is called only by RMOL. Indeed, the revenue management component (RMOL) needs very basic set up in order to perform optimisation at leg-level. Hence, there are:

- a dedicated inventory ('XX'),
- the corresponding flight-date (#9999, departing 01/01/1900),
- a leg-date (departing and arriving from/to 'XXX' airport),
-
- a leg-cabin ('X').
-

Most of the data is dummy because RMOL uses only the cabin capacity from that part of the BOM tree.

Parameters

<i>const</i> CabinCapacity_T& Cabin capacity for revenue management optimisation.

Definition at line 178 of file [STDAIR_Service.cpp](#).

34.130.3.3 void stdair::STDAIR_Service::buildSampleTravelSolutionForPricing (
 TravelSolutionList_T & ioTravelSolutionList)

Build a sample list of travel solutions.

As of now (March 2011), that list is made of the following travel solutions:

- BA9
- LHR-SYD
- 2011-06-10

Parameters

<i>TravelSolutionList_T</i> &	Sample list of travel solution structures. It should be given empty. It is altered with the returned sample.
-------------------------------	--

Definition at line 192 of file [STDAIR_Service.cpp](#).

34.130.3.4 void stdair::STDAIR_Service::buildSampleTravelSolutions (TravelSolutionList_T
& ioTravelSolutionList)

Build a sample list of travel solutions.

As of now (March 2011), that list is made of the following travel solutions:

- BA9
- LHR-SYD
- 2011-06-10
- Q
- WTP: 900
- Change fee: 20; Non refundable; Saturday night stay

Parameters

<i>TravelSolutionList_T</i> &	Sample list of travel solution structures. It should be given empty. It is altered with the returned sample.
-------------------------------	--

Definition at line 199 of file [STDAIR_Service.cpp](#).

34.130.3.5 BookingRequestStruct stdair::STDAIR_Service::buildSampleBookingRequest (
 const bool isForCRS = false)

Build a sample booking request structure.

As of now (March 2011), the sample booking request is made of the following parameters:

- Return trip (inbound): LHR-SYD (POS: LHR, Channel: DN),
- Departing 10-JUN-2011 around 8:00, staying 7 days
- Requested on 15-MAY-2011 at 10:00
- Economy cabin, 3 persons, FF member
- WTP: 1000.0 EUR
- Dis-utility: 100.0 EUR/hour

As of now (March 2011), the CRS-related booking request is made of the following parameters:

- Return trip (inbound): SIN-BKK (POS: SIN, Channel: IN),
- Departing 30-JAN-2010 around 10:00, staying 7 days
- Requested on 22-JAN-2010 at 10:00
- Economy cabin, 3 persons, FF member
- WTP: 1000.0 EUR
- Dis-utility: 100.0 EUR/hour

Parameters

<code>const</code> bool isForCRS Whether the sample booking request is for CRS.

Returns

[BookingRequestStruct](#)& Sample booking request structure.

Definition at line 206 of file [STDAIR_Service.cpp](#).

```
34.130.3.6  const Count_T & stdair::STDAIR_
            Service::getExpectedTotalNumberOfEventsToBeGenerated ( )
            const
```

Get the expected number of events to be generated.

The `getExpectedTotalNbOfEvents()` method is called on the underlying [EventQueue](#) object, which keeps track of that number.

Note

That number usually corresponds to an expectation (i.e., the mean value of a random distribution), and may not be accurate. The actual number will be known after calling the `generateFirstEvents()` method for each event type (e.g., booking request, optimisation notification, etc).

Returns

`const Count_T` Expected number of events to be generated.

Definition at line 442 of file [STDAIR_Service.cpp](#).

References [stdair::EventQueue::getExpectedTotalNbOfEvents\(\)](#).

```
34.130.3.7  const Count_T & stdair::STDAIR_-
            Service::getExpectedTotalNumberOfEventsToBeGenerated (
            const EventType::EN_EventType & iType ) const
```

Get the expected number of events to be generated for the given event type.

The `getExpectedTotalNbOfEvents()` method is called on the underlying [EventQueue](#) object, which keeps track of that number.

Note

That number usually corresponds to an expectation (i.e., the mean value of a random distribution), and may not be accurate. The actual number will be known after calling the `generateFirstEvents()` method for each event type (e.g., booking request, optimisation notification, etc).

Parameters

<i>const</i> EventType_T& Event type for which the number is calculated.
--

Returns

const Count_T& Expected number of events to be generated.

Definition at line 461 of file [STDAIR_Service.cpp](#).

References [stdair::EventQueue::getExpectedTotalNbOfEvents\(\)](#).

```
34.130.3.8  const Count_T & stdair::STDAIR_-
            Service::getActualTotalNumberOfEventsToBeGenerated ( )
            const
```

Get the actual number of events to be generated for all the demand streams.

The `getActualTotalNbOfEvents()` method is called on the underlying [EventQueue](#) object, which keeps track of that number.

Note

That number is being known after calling the `generateFirstEvents()` method.

Returns

const Count_T& Expected number of events to be generated.

Definition at line 480 of file [STDAIR_Service.cpp](#).

References [stdair::EventQueue::getActualTotalNbOfEvents\(\)](#).

34.130.3.9 **const Count_T & stdair::STDAIR_Service::getActualTotalNumberOfEventsToBeGenerated (const EventType::EN_EventType & *iType*) const**

Get the actual number of events to be generated for the given event type.

The getActualTotalNbOfEvents() method is called on the underlying [EventQueue](#) object, which keeps track of that number.

Note

That number is being known after calling the generateFirstEvents() method.

Parameters

<i>const</i> EventType_T& Event type for which the number is calculated.
--

Returns

const Count_T& Expected number of events to be generated.

Definition at line 499 of file [STDAIR_Service.cpp](#).

References [stdair::EventQueue::getActualTotalNbOfEvents\(\)](#).

34.130.3.10 **ProgressStatusSet stdair::STDAIR_Service::popEvent (EventStruct & ioEventStruct) const**

Pop the next coming (in time) event, and remove it from the event queue.

- The next coming (in time) event corresponds to the event having the earliest date-time stamp. In other words, it is the first/front element of the event queue.
- That (first) event/element is then removed from the event queue
- The progress status is updated for the corresponding event type.

Returns

[EventStruct](#) A copy of the event structure, which comes first in time from within the event queue.

Definition at line 517 of file [STDAIR_Service.cpp](#).

References [stdair::EventQueue::popEvent\(\)](#).

34.130.3.11 **bool stdair::STDAIR_Service::isQueueDone () const**

States whether the event queue has reached the end.

For now, that method states whether or not the event queue is empty.

Definition at line 531 of file [STDAIR_Service.cpp](#).

References [stdair::EventQueue::isQueueDone\(\)](#).

34.130.3.12 void stdair::STDAIR_Service::reset () const

Reset the context of the demand streams for another demand generation without having to reparse the demand input file.

Definition at line 548 of file [STDAIR_Service.cpp](#).

References [stdair::EventQueue::reset\(\)](#).

34.130.3.13 std::string stdair::STDAIR_Service::jsonExport (const AirlineCode_T & iAirlineCode, const FlightNumber_T & iFlightNumber, const Date_T & iDepartureDate) const

Recursively dump, in the returned string and in JSON format, the flight-date corresponding to the parameters given as input.

Parameters

<i>const</i>	AirlineCode_T& Airline code of the flight to dump.
<i>const</i>	FlightNumber_T& Flight number of the flight to dump.
<i>const</i>	Date_T& Departure date of the flight to dump.

Returns

std::string Output string in which the BOM tree is JSON-ified.

Definition at line 218 of file [STDAIR_Service.cpp](#).

References [stdair::BomRetriever::retrieveFlightDateFromKeySet\(\)](#).

34.130.3.14 std::string stdair::STDAIR_Service::list (const AirlineCode_T & iAirlineCode = "all", const FlightNumber_T & iFlightNumber = 0) const

Display the list of flight-dates (contained within the BOM tree) corresponding to the parameters given as input.

Parameters

<i>const</i>	AirlineCode& Airline for which the flight-dates should be displayed. If set to "all" (the default), all the inventories will be displayed.
<i>const</i>	FlightNumber_T& Flight number for which all the departure dates should be displayed. If set to 0 (the default), all the flight numbers will be displayed.

Returns

std::string Output string in which the BOM tree is logged/dumped.

Definition at line 265 of file [STDAIR_Service.cpp](#).

34.130.3.15 std::string stdair::STDAIR_Service::listAirportPairDateRange () const

Display the list of airports pairs and date ranges (contained within the BOM tree)

Parameters

<code>std::ostream&</code>	Output stream in which the airport pairs and date ranges are logged/dumped.
--------------------------------	---

Definition at line 283 of file [STDAIR_Service.cpp](#).

34.130.3.16 `bool stdair::STDAIR_Service::check (const AirlineCode_T & iAirlineCode, const FlightNumber_T & iFlightNumber, const Date_T & iDepartureDate) const`

Check whether the given flight-date is a valid one.

Parameters

<code>const stdair::AirlineCode_T</code>	& Airline code of the flight to check.
<code>const stdair::FlightNumber_T</code>	& Flight number of the flight to check.
<code>const stdair::Date_T</code>	& Departure date of the flight to check.

Returns

bool Whether or not the given flight date is valid.

Definition at line 300 of file [STDAIR_Service.cpp](#).

References [stdair::BomRetriever::retrieveFlightDateFromKeySet\(\)](#).

34.130.3.17 `bool stdair::STDAIR_Service::check (const AirportCode_T & ioOrigin, const AirportCode_T & ioDestination, const Date_T & iDepartureDate) const`

Check whether the given couple airportpair-date is a valid one.

Parameters

<code>const stdair::AirportCode_T</code>	& Origin airport of the fare rule to check.
<code>const stdair::AirportCode_T</code>	& Destination airport of the fare rule to check.
<code>const stdair::Date_T</code>	& Departure date of the fare rule to check.

Returns

bool Whether or not the given airportpair-date couple is a valid one.

Definition at line 322 of file [STDAIR_Service.cpp](#).

References [stdair::BomRetriever::retrieveDatePeriodListFromKeySet\(\)](#).

34.130.3.18 `std::string stdair::STDAIR_Service::csvDisplay () const`

Recursively display (dump in the returned string) the objects of the BOM tree.

Returns

std::string Output string in which the BOM tree is logged/dumped.

Definition at line 345 of file [STDAIR_Service.cpp](#).

Referenced by [csvDisplay\(\)](#).

34.130.3.19 `std::string stdair::STDAIR_Service::csvDisplay (const AirlineCode_T & iAirlineCode, const FlightNumber_T & iFlightNumber, const Date_T & iDepartureDate) const`

Recursively display (dump in the returned string) the flight-date corresponding to the parameters given as input.

Parameters

<i>const</i>	AirlineCode_T& Airline code of the flight to display.
<i>const</i>	FlightNumber_T& Flight number of the flight to display.
<i>const</i>	Date_T& Departure date of the flight to display.

Returns

std::string Output string in which the BOM tree is logged/dumped.

Definition at line 363 of file [STDAIR_Service.cpp](#).

References [csvDisplay\(\)](#), and [stdair::BomRetriever::retrieveFlightDateFromKeySet\(\)](#).

34.130.3.20 `std::string stdair::STDAIR_Service::csvDisplay (const TravelSolutionList_T & iTravelSolutionList) const`

Display (dump in the returned string) the full list of travel solution structures.

Returns

std::string Output string in which the list of travel solutions is logged/dumped.

Definition at line 394 of file [STDAIR_Service.cpp](#).

References [csvDisplay\(\)](#).

34.130.3.21 `std::string stdair::STDAIR_Service::csvDisplay (const AirportCode_T & iOrigin, const AirportCode_T & iDestination, const Date_T & iDepartureDate) const`

Recursively display (dump in the returned string) the fare-rules corresponding to the parameters given as input.

Parameters

<i>const</i>	AirportCode_T& Origin airport of the fare-rules to display
<i>const</i>	AirportCode_T& Destination airport of the fare-rules to display.
<i>const</i>	Date_T& Departure date of the fare-rules to display.

Returns

std::string Output string in which the BOM tree is logged/dumped.

Definition at line 405 of file [STDAIR_Service.cpp](#).

References [csvDisplay\(\)](#), and [stdair::BomRetriever::retrieveDatePeriodListFromKeySet\(\)](#).

34.130.3.22 BomRoot & stdair::STDAIR_Service::getBomRoot () const

Get a reference on the [BomRoot](#) object.

If the service context has not been initialised, that method throws an exception (failing assertion).

Returns

[BomRoot](#)& Reference on the [BomRoot](#).

Definition at line 126 of file [STDAIR_Service.cpp](#).

34.130.3.23 EventQueue & stdair::STDAIR_Service::getEventQueue () const

Get a reference on the [EventQueue](#) object.

If the service context has not been initialised, that method throws an exception (failing assertion).

Returns

[EventQueue](#)& Reference on the [EventQueue](#).

Definition at line 134 of file [STDAIR_Service.cpp](#).

34.130.3.24 BasLogParams stdair::STDAIR_Service::getLogParams () const

Get the log parameters.

Returns

[BasLogParams](#) Copy of the structure holding the log parameters.

Definition at line 142 of file [STDAIR_Service.cpp](#).

34.130.3.25 const BasDBParams & stdair::STDAIR_Service::getDBParams () const

Get the database parameters.

Returns

const [BasDBParams](#)& Reference on the structure holding the database parameters.

Definition at line 147 of file [STDAIR_Service.cpp](#).

34.130.3.26 const ServiceInitialisationType & stdair::STDAIR_Service::getServiceInitialisationType () const

Get the type of initialisation (e.g., not yet, file parsing, sample BOM) which the component (owner of the current [STDAIR_Service](#) instance) has gone through.

Returns

const [ServiceInitialisationType](#)& Reference on the type of initialisation (enumeration structure).

Definition at line 156 of file [STDAIR_Service.cpp](#).

The documentation for this class was generated from the following files:

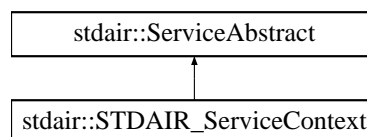
- [stdair/STDAIR_Service.hpp](#)
- [stdair/service/STDAIR_Service.cpp](#)

34.131 stdair::STDAIR_ServiceContext Class Reference

Class holding the context of the Stdair services.

```
#include <stdair/service/STDAIR_ServiceContext.hpp>
```

Inheritance diagram for stdair::STDAIR_ServiceContext:

**Public Member Functions**

- virtual void [toStream](#) (std::ostream &ioOut) const
- virtual void [fromStream](#) (std::istream &ioIn)

Friends

- class [STDAIR_Service](#)
- class [FacSTDAIRServiceContext](#)

34.131.1 Detailed Description

Class holding the context of the Stdair services.

Definition at line 25 of file [STDAIR_ServiceContext.hpp](#).

34.131.2 Member Function Documentation

34.131.2.1 virtual void stdair::ServiceAbstract::toStream (std::ostream & *ioOut*) const
[inline, virtual, inherited]

Dump a Business Object into an output stream.

Parameters

<i>ostream</i> & the output stream.

Definition at line 28 of file [ServiceAbstract.hpp](#).

Referenced by [operator<<\(\)](#).

34.131.2.2 virtual void stdair::ServiceAbstract::fromStream (std::istream & *ioIn*)
[inline, virtual, inherited]

Read a Business Object from an input stream.

Parameters

<i>istream</i> & the input stream.

Definition at line 35 of file [ServiceAbstract.hpp](#).

Referenced by [operator>>\(\)](#).

34.131.3 Friends And Related Function Documentation

34.131.3.1 friend class **STDAIR_Service** [friend]

The [STDAIR_Service](#) class should be the sole class to get access to ServiceContext content: general users do not want to bother with a context interface.

Definition at line 29 of file [STDAIR_ServiceContext.hpp](#).

34.131.3.2 friend class **FacSTDAIRServiceContext** [friend]

Definition at line 30 of file [STDAIR_ServiceContext.hpp](#).

The documentation for this class was generated from the following file:

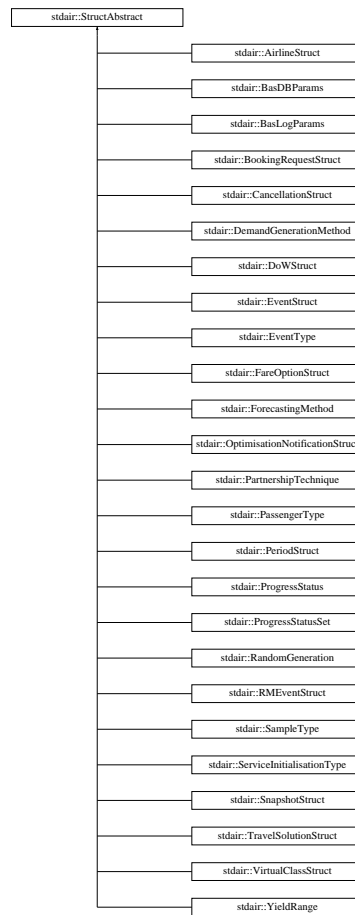
- stdair/service/[STDAIR_ServiceContext.hpp](#)

34.132 stdair::StructAbstract Struct Reference

Base class for the light structures.

```
#include <stdair/basic/StructAbstract.hpp>
```

Inheritance diagram for stdair::StructAbstract:



Public Member Functions

- virtual [~StructAbstract](#) ()
- void [toStream](#) (std::ostream &ioOut) const
- virtual void [fromStream](#) (std::istream &ioIn)
- virtual const std::string [describe](#) () const =0

Protected Member Functions

- [StructAbstract](#) ()

34.132.1 Detailed Description

Base class for the light structures.

Definition at line 16 of file [StructAbstract.hpp](#).

34.132.2 Constructor & Destructor Documentation

34.132.2.1 `virtual stdair::StructAbstract::~~StructAbstract () [inline, virtual]`

Destructor.

Definition at line 22 of file [StructAbstract.hpp](#).

34.132.2.2 `stdair::StructAbstract::StructAbstract () [inline, protected]`

Protected Default Constructor to ensure this class is abstract.

Definition at line 49 of file [StructAbstract.hpp](#).

34.132.3 Member Function Documentation

34.132.3.1 `void stdair::StructAbstract::toStream (std::ostream & ioOut) const [inline]`

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Reimplemented in [stdair::YieldRange](#), [stdair::AirlineStruct](#), [stdair::BookingRequestStruct](#), [stdair::CancellationStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::TravelSolutionStruct](#), and [stdair::VirtualClassStruct](#).

Definition at line 29 of file [StructAbstract.hpp](#).

References [describe\(\)](#).

Referenced by [operator<<\(\)](#).

34.132.3.2 `virtual void stdair::StructAbstract::fromStream (std::istream & ioln) [inline, virtual]`

Read a Business Object from an input stream.

Parameters

<i>istream&</i> the input stream.

Reimplemented in [stdair::ProgressStatusSet](#), [stdair::YieldRange](#), [stdair::AirlineStruct](#), [stdair::BookingRequestStruct](#), [stdair::CancellationStruct](#), [stdair::EventStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::TravelSolutionStruct](#), and [stdair::VirtualClassStruct](#).

Definition at line 38 of file [StructAbstract.hpp](#).

Referenced by [operator>>\(\)](#).

34.132.3.3 `virtual const std::string stdair::StructAbstract::describe () const [pure virtual]`

Display of the structure.

Implemented in [stdair::BasDBParams](#), [stdair::BasLogParams](#), [stdair::DemandGenerationMethod](#), [stdair::EventType](#), [stdair::ForecastingMethod](#), [stdair::PartnershipTechnique](#), [stdair::PassengerType](#), [stdair::ProgressStatus](#), [stdair::ProgressStatusSet](#), [stdair::RandomGeneration](#), [stdair::SampleType](#), [stdair::ServiceInitialisationType](#), [stdair::YieldRange](#), [stdair::AirlineStruct](#), [stdair::BookingRequestStruct](#), [stdair::CancellationStruct](#), [stdair::DoWStruct](#), [stdair::EventStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::PeriodStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::TravelSolutionStruct](#), and [stdair::VirtualClassStruct](#).

Referenced by [toStream\(\)](#).

The documentation for this struct was generated from the following file:

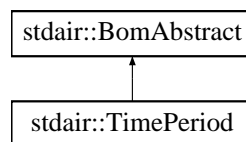
- [stdair/basic/StructAbstract.hpp](#)

34.133 stdair::TimePeriod Class Reference

Class representing the actual attributes for a fare time-period.

```
#include <stdair/bom/TimePeriod.hpp>
```

Inheritance diagram for `stdair::TimePeriod`:



Public Types

- typedef [TimePeriodKey](#) [Key_T](#)

Public Member Functions

- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- std::string [toString](#) () const
- const std::string [describeKey](#) () const
- const [Key_T](#) & [getKey](#) () const
- [BomAbstract](#) *const [getParent](#) () const
- const [HolderMap_T](#) & [getHolderMap](#) () const
- const [Time_T](#) & [getTimeRangeStart](#) () const
- const [Time_T](#) & [getTimeRangeEnd](#) () const
- bool [isDepartureTimeValid](#) (const [Time_T](#) &) const

Protected Member Functions

- `TimePeriod` (const `Key_T` &)
- virtual `~TimePeriod` ()

Protected Attributes

- `Key_T_key`
- `BomAbstract` * `_parent`
- `HolderMap_T_holderMap`

Friends

- class `FacBom`
- class `FacBomManager`

34.133.1 Detailed Description

Class representing the actual attributes for a fare time-period.

Definition at line 18 of file [TimePeriod.hpp](#).

34.133.2 Member Typedef Documentation

34.133.2.1 `typedef TimePeriodKey stdair::TimePeriod::Key_T`

Definition allowing to retrieve the associated BOM key type.

Definition at line 27 of file [TimePeriod.hpp](#).

34.133.3 Constructor & Destructor Documentation

34.133.3.1 `stdair::TimePeriod::TimePeriod (const Key_T & iKey)` [protected]

Main constructor.

Definition at line 28 of file [TimePeriod.cpp](#).

34.133.3.2 `stdair::TimePeriod::~~TimePeriod ()` [protected, virtual]

Destructor.

Definition at line 33 of file [TimePeriod.cpp](#).

34.133.4 Member Function Documentation

34.133.4.1 `void stdair::TimePeriod::toStream (std::ostream & ioOut) const` `[inline, virtual]`

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Implements [stdair::BomAbstract](#).

Definition at line 37 of file [TimePeriod.hpp](#).

References [toString\(\)](#).

34.133.4.2 `void stdair::TimePeriod::fromStream (std::istream & ioIn)` `[inline, virtual]`

Read a Business Object from an input stream.

Parameters

<i>istream&</i> the input stream.

Implements [stdair::BomAbstract](#).

Definition at line 46 of file [TimePeriod.hpp](#).

34.133.4.3 `std::string stdair::TimePeriod::toString () const` `[virtual]`

Get the serialised version of the Business Object.

Implements [stdair::BomAbstract](#).

Definition at line 37 of file [TimePeriod.cpp](#).

References [describeKey\(\)](#).

Referenced by [toStream\(\)](#).

34.133.4.4 `const std::string stdair::TimePeriod::describeKey () const` `[inline]`

Get a string describing the key.

Definition at line 57 of file [TimePeriod.hpp](#).

References [_key](#), and [stdair::TimePeriodKey::toString\(\)](#).

Referenced by [toString\(\)](#).

34.133.4.5 `const Key_T& stdair::TimePeriod::getKey () const` `[inline]`

Get the primary key (time range start, time range end).

Definition at line 66 of file [TimePeriod.hpp](#).

References [_key](#).

34.133.4.6 BomAbstract* const stdair::TimePeriod::getParent () const [inline]

Get a reference on the parent object instance.

Definition at line 73 of file [TimePeriod.hpp](#).

References [_parent](#).

34.133.4.7 const HolderMap_T& stdair::TimePeriod::getHolderMap () const [inline]

Get a reference on the children holder.

Definition at line 80 of file [TimePeriod.hpp](#).

References [_holderMap](#).

34.133.4.8 const Time_T& stdair::TimePeriod::getTimeRangeStart () const [inline]

Get the time range start.

Definition at line 87 of file [TimePeriod.hpp](#).

References [_key](#), and [stdair::TimePeriodKey::getTimeRangeStart\(\)](#).

Referenced by [isDepartureTimeValid\(\)](#).

34.133.4.9 const Time_T& stdair::TimePeriod::getTimeRangeEnd () const [inline]

Get the time range end

Definition at line 94 of file [TimePeriod.hpp](#).

References [_key](#), and [stdair::TimePeriodKey::getTimeRangeEnd\(\)](#).

Referenced by [isDepartureTimeValid\(\)](#).

34.133.4.10 bool stdair::TimePeriod::isDepartureTimeValid (const Time_T & iFlightTime) const

Check if the given departure time is included in the departure period of the segment path.

Definition at line 45 of file [TimePeriod.cpp](#).

References [getTimeRangeEnd\(\)](#), [getTimeRangeStart\(\)](#), and [STDAIR_LOG_DEBUG](#).

34.133.5 Friends And Related Function Documentation**34.133.5.1 friend class FacBom** [friend]

Definition at line 19 of file [TimePeriod.hpp](#).

34.133.5.2 friend class FacBomManager [friend]

Definition at line 20 of file [TimePeriod.hpp](#).

34.133.6 Member Data Documentation

34.133.6.1 Key_T stdair::TimePeriod::_key [protected]

Primary key (flight number and departure date).

Definition at line 132 of file [TimePeriod.hpp](#).

Referenced by [describeKey\(\)](#), [getKey\(\)](#), [getTimeRangeEnd\(\)](#), and [getTimeRangeStart\(\)](#).

34.133.6.2 BomAbstract* stdair::TimePeriod::_parent [protected]

Pointer on the parent class ([Inventory](#)).

Definition at line 137 of file [TimePeriod.hpp](#).

Referenced by [getParent\(\)](#).

34.133.6.3 HolderMap_T stdair::TimePeriod::_holderMap [protected]

Map holding the children.

Definition at line 142 of file [TimePeriod.hpp](#).

Referenced by [getHolderMap\(\)](#).

The documentation for this class was generated from the following files:

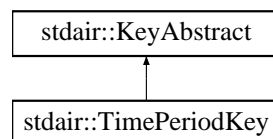
- [stdair/bom/TimePeriod.hpp](#)
- [stdair/bom/TimePeriod.cpp](#)

34.134 stdair::TimePeriodKey Struct Reference

Key of time-period.

```
#include <stdair/bom/TimePeriodKey.hpp>
```

Inheritance diagram for stdair::TimePeriodKey:



Public Member Functions

- [TimePeriodKey](#) (const [Time_T](#) &, const [Time_T](#) &)
- [TimePeriodKey](#) (const [TimePeriodKey](#) &)
- [~TimePeriodKey](#) ()
- const [Time_T](#) & [getTimeRangeStart](#) () const
- const [Time_T](#) & [getTimeRangeEnd](#) () const

- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- const std::string [toString](#) () const

34.134.1 Detailed Description

Key of time-period.

Definition at line 15 of file [TimePeriodKey.hpp](#).

34.134.2 Constructor & Destructor Documentation

34.134.2.1 **stdair::TimePeriodKey::TimePeriodKey** (const Time_T & *iTimeRangeStart*, const Time_T & *iTimeRangeEnd*)

Main constructor.

Definition at line 21 of file [TimePeriodKey.cpp](#).

34.134.2.2 **stdair::TimePeriodKey::TimePeriodKey** (const TimePeriodKey & *iKey*)

Copy constructor.

Definition at line 28 of file [TimePeriodKey.cpp](#).

34.134.2.3 **stdair::TimePeriodKey::~~TimePeriodKey** ()

Destructor.

Definition at line 34 of file [TimePeriodKey.cpp](#).

34.134.3 Member Function Documentation

34.134.3.1 **const Time_T& stdair::TimePeriodKey::getTimeRangeStart** () const
[inline]

Get the time period start.

Definition at line 35 of file [TimePeriodKey.hpp](#).

Referenced by [stdair::TimePeriod::getTimeRangeStart\(\)](#).

34.134.3.2 **const Time_T& stdair::TimePeriodKey::getTimeRangeEnd** () const [inline]

Get the time period end.

Definition at line 42 of file [TimePeriodKey.hpp](#).

Referenced by [stdair::TimePeriod::getTimeRangeEnd\(\)](#).

34.134.3.3 void stdair::TimePeriodKey::toStream (std::ostream & *ioOut*) const
[virtual]

Dump a Business Object Key into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 38 of file [TimePeriodKey.cpp](#).

References [toString\(\)](#).

34.134.3.4 void stdair::TimePeriodKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters

<i>istream&</i> the input stream.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 43 of file [TimePeriodKey.cpp](#).

34.134.3.5 const std::string stdair::TimePeriodKey::toString () const [virtual]

Get the serialised version of the Business Object Key. That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 47 of file [TimePeriodKey.cpp](#).

Referenced by [stdair::TimePeriod::describeKey\(\)](#), and [toStream\(\)](#).

The documentation for this struct was generated from the following files:

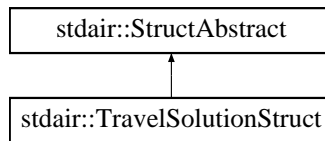
- [stdair/bom/TimePeriodKey.hpp](#)
- [stdair/bom/TimePeriodKey.cpp](#)

34.135 stdair::TravelSolutionStruct Struct Reference

Structure holding the elements of a travel solution.

```
#include <stdair/bom/TravelSolutionStruct.hpp>
```

Inheritance diagram for stdair::TravelSolutionStruct:



Public Member Functions

- const [SegmentPath_T](#) & [getSegmentPath](#) () const
- const [ClassAvailabilityMapHolder_T](#) & [getClassAvailabilityMapHolder](#) () const
- const [ClassYieldMapHolder_T](#) & [getClassYieldMapHolder](#) () const
- const [BidPriceVectorHolder_T](#) & [getBidPriceVectorHolder](#) () const
- const [ClassBpvMapHolder_T](#) & [getClassBpvMapHolder](#) () const
- const [FareOptionList_T](#) & [getFareOptionList](#) () const
- [FareOptionList_T](#) & [getFareOptionListRef](#) ()
- const [FareOptionStruct](#) & [getChosenFareOption](#) () const
- void [addSegment](#) (const std::string &)
- void [addClassAvailabilityMap](#) (const [ClassAvailabilityMap_T](#) &)
- void [addClassYieldMap](#) (const [ClassYieldMap_T](#) &)
- void [addBidPriceVector](#) (const [BidPriceVector_T](#) &)
- void [addClassBpvMap](#) (const [ClassBpvMap_T](#) &)
- void [addFareOption](#) (const [FareOptionStruct](#) &)
- void [setChosenFareOption](#) (const [FareOptionStruct](#) &iChosenFO)
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- const std::string [describe](#) () const
- const std::string [display](#) () const
- [TravelSolutionStruct](#) ()
- [~TravelSolutionStruct](#) ()

34.135.1 Detailed Description

Structure holding the elements of a travel solution.

Definition at line 24 of file [TravelSolutionStruct.hpp](#).

34.135.2 Constructor & Destructor Documentation

34.135.2.1 stdair::TravelSolutionStruct::TravelSolutionStruct ()

Default constructor.

Definition at line 15 of file [TravelSolutionStruct.cpp](#).

34.135.2.2 stdair::TravelSolutionStruct::~~TravelSolutionStruct ()

Destructor.

Definition at line 19 of file [TravelSolutionStruct.cpp](#).

34.135.3 Member Function Documentation

34.135.3.1 const SegmentPath_T& stdair::TravelSolutionStruct::getSegmentPath () const [inline]

Get the segment path.

Definition at line 28 of file [TravelSolutionStruct.hpp](#).

34.135.3.2 const ClassAvailabilityMapHolder_T& stdair::TravelSolutionStruct::getClassAvailabilityMapHolder () const [inline]

Get the holder of availabilities.

Definition at line 33 of file [TravelSolutionStruct.hpp](#).

34.135.3.3 const ClassYieldMapHolder_T& stdair::TravelSolutionStruct::getClassYieldMapHolder () const [inline]

Get the holder of yields.

Definition at line 38 of file [TravelSolutionStruct.hpp](#).

34.135.3.4 const BidPriceVectorHolder_T& stdair::TravelSolutionStruct::getBidPriceVectorHolder () const [inline]

Get the holder of bid price vectors.

Definition at line 43 of file [TravelSolutionStruct.hpp](#).

34.135.3.5 const ClassBpvMapHolder_T& stdair::TravelSolutionStruct::getClassBpvMapHolder () const [inline]

Get the holder of class - bid price reference.

Definition at line 48 of file [TravelSolutionStruct.hpp](#).

34.135.3.6 const FareOptionList_T& stdair::TravelSolutionStruct::getFareOptionList () const [inline]

Get the list of fare options.

Definition at line 53 of file [TravelSolutionStruct.hpp](#).

34.135.3.7 **FareOptionList_T**& stdair::TravelSolutionStruct::getFareOptionListRef ()
[inline]

Get the non-const list of fare options.

Definition at line 58 of file [TravelSolutionStruct.hpp](#).

34.135.3.8 **const FareOptionStruct**& stdair::TravelSolutionStruct::getChosenFareOption ()
const [inline]

Get the chosen fare option.

Definition at line 63 of file [TravelSolutionStruct.hpp](#).

34.135.3.9 **void** stdair::TravelSolutionStruct::addSegment (**const** std::string & *iKey*)

Add a segment key to the segment path.

Definition at line 133 of file [TravelSolutionStruct.cpp](#).

34.135.3.10 **void** stdair::TravelSolutionStruct::addClassAvailabilityMap (**const**
ClassAvailabilityMap_T & *iMap*)

Add a class availability map.

Definition at line 139 of file [TravelSolutionStruct.cpp](#).

34.135.3.11 **void** stdair::TravelSolutionStruct::addClassYieldMap (**const** **ClassYieldMap_T** &
iMap)

Add a class yield map.

Definition at line 145 of file [TravelSolutionStruct.cpp](#).

34.135.3.12 **void** stdair::TravelSolutionStruct::addBidPriceVector (**const** **BidPriceVector_T** &
iBpv)

Add a bid price vector.

Definition at line 151 of file [TravelSolutionStruct.cpp](#).

34.135.3.13 **void** stdair::TravelSolutionStruct::addClassBpvMap (**const** **ClassBpvMap_T** &
iMap)

Add a class bpv reference map.

Definition at line 157 of file [TravelSolutionStruct.cpp](#).

34.135.3.14 **void** stdair::TravelSolutionStruct::addFareOption (**const** **FareOptionStruct** &
iFareOption)

Add a fare option.

Definition at line 163 of file [TravelSolutionStruct.cpp](#).

34.135.3.15 void stdair::TravelSolutionStruct::setChosenFareOption (const
FareOptionStruct & iChosenFO) [inline]

Set the chosen fare option.

Definition at line 89 of file [TravelSolutionStruct.hpp](#).

34.135.3.16 void stdair::TravelSolutionStruct::toStream (std::ostream & ioOut) const

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Reimplemented from [stdair::StructAbstract](#).

Definition at line 23 of file [TravelSolutionStruct.cpp](#).

References [describe\(\)](#).

34.135.3.17 void stdair::TravelSolutionStruct::fromStream (std::istream & ioIn)
[virtual]

Read a Business Object from an input stream.

Parameters

<i>istream&</i> the input stream.

Reimplemented from [stdair::StructAbstract](#).

Definition at line 28 of file [TravelSolutionStruct.cpp](#).

34.135.3.18 const std::string stdair::TravelSolutionStruct::describe () const [virtual]

Display of the structure.

Implements [stdair::StructAbstract](#).

Definition at line 32 of file [TravelSolutionStruct.cpp](#).

References [stdair::FareOptionStruct::describe\(\)](#), [stdair::BomKeyManager::extractKeys\(\)](#),
and [stdair::ParsedKey::toString\(\)](#).

Referenced by [toStream\(\)](#).

34.135.3.19 const std::string stdair::TravelSolutionStruct::display () const

Display of the structure.

Definition at line 74 of file [TravelSolutionStruct.cpp](#).

References [stdair::FareOptionStruct::display\(\)](#), [stdair::BomKeyManager::extractKeys\(\)](#),
and [stdair::ParsedKey::toString\(\)](#).

The documentation for this struct was generated from the following files:

- [stdair/bom/TravelSolutionStruct.hpp](#)
- [stdair/bom/TravelSolutionStruct.cpp](#)

34.136 `soci::type_conversion< stdair::AirlineStruct >` Struct Template Reference

```
#include <stdair/dbadaptor/DbAirline.hpp>
```

Public Types

- typedef values [base_type](#)

Static Public Member Functions

- static void [from_base](#) (values const &iAirlineValues, indicator, [stdair::AirlineStruct](#) &ioAirline)
- static void [to_base](#) (const [stdair::AirlineStruct](#) &iAirline, values &ioAirlineValues, indicator &ioIndicator)

34.136.1 Detailed Description

```
template<>struct soci::type_conversion< stdair::AirlineStruct >
```

Specify how the AirlineStruct struct can be converted to (resp. from) values stored into (resp. retrieved from) database, using the SOCI framework.

Definition at line [25](#) of file [DbAirline.hpp](#).

34.136.2 Member Typedef Documentation

34.136.2.1 typedef values `soci::type_conversion< stdair::AirlineStruct >::base_type`

Definition at line [27](#) of file [DbAirline.hpp](#).

34.136.3 Member Function Documentation

34.136.3.1 `void soci::type_conversion< stdair::AirlineStruct >::from_base (values const & iAirlineValues, indicator , stdair::AirlineStruct & ioAirline) [static]`

Fill an Airline object from the database values.

Definition at line [17](#) of file [DbAirline.cpp](#).

References [stdair::AirlineStruct::setAirlineCode\(\)](#), and [stdair::AirlineStruct::setAirlineName\(\)](#).

34.136.3.2 `void soci::type_conversion< stdair::AirlineStruct >::to_base (const
stdair::AirlineStruct & iAirline, values & ioAirlineValues, indicator & ioIndicator)
[static]`

Fill the database values from an Airline object.

Definition at line 30 of file [Dbairline.cpp](#).

References [stdair::AirlineStruct::getAirlineCode\(\)](#), and [stdair::AirlineStruct::getAirlineName\(\)](#).

The documentation for this struct was generated from the following files:

- [stdair/dbadaptor/Dbairline.hpp](#)
- [stdair/dbadaptor/Dbairline.cpp](#)

34.137 `TypeWithSize< size >` Class Template Reference

```
#include <stdair/basic/float_utils_google.hpp>
```

Public Types

- typedef void [UInt](#)

34.137.1 Detailed Description

```
template<size_t size>class TypeWithSize< size >
```

Definition at line 54 of file [float_utils_google.hpp](#).

34.137.2 Member Typedef Documentation

34.137.2.1 `template<size_t size> typedef void TypeWithSize< size >::UInt`

Definition at line 58 of file [float_utils_google.hpp](#).

The documentation for this class was generated from the following file:

- [stdair/basic/float_utils_google.hpp](#)

34.138 `TypeWithSize< 4 >` Class Template Reference

```
#include <stdair/basic/float_utils_google.hpp>
```

Public Types

- typedef int [Int](#)
- typedef unsigned int [UInt](#)

34.138.1 Detailed Description

```
template<>class TypeWithSize< 4 >
```

Definition at line 63 of file [float_utils_google.hpp](#).

34.138.2 Member Typedef Documentation

34.138.2.1 `typedef int TypeWithSize< 4 >::Int`

Definition at line 69 of file [float_utils_google.hpp](#).

34.138.2.2 `typedef unsigned int TypeWithSize< 4 >::UInt`

Definition at line 70 of file [float_utils_google.hpp](#).

The documentation for this class was generated from the following file:

- [stdair/basic/float_utils_google.hpp](#)

34.139 `TypeWithSize< 8 >` Class Template Reference

```
#include <stdair/basic/float_utils_google.hpp>
```

Public Types

- `typedef long long` [Int](#)
- `typedef unsigned long long` [UInt](#)

34.139.1 Detailed Description

```
template<>class TypeWithSize< 8 >
```

Definition at line 75 of file [float_utils_google.hpp](#).

34.139.2 Member Typedef Documentation

34.139.2.1 `typedef long long TypeWithSize< 8 >::Int`

Definition at line 81 of file [float_utils_google.hpp](#).

34.139.2.2 `typedef unsigned long long TypeWithSize< 8 >::UInt`

Definition at line 82 of file [float_utils_google.hpp](#).

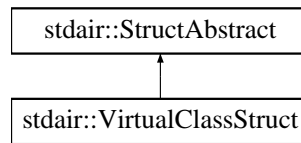
The documentation for this class was generated from the following file:

- [stdair/basic/float_utils_google.hpp](#)

34.140 stdair::VirtualClassStruct Struct Reference

```
#include <stdair/bom/VirtualClassStruct.hpp>
```

Inheritance diagram for stdair::VirtualClassStruct:



Public Member Functions

- const [Yield_T](#) & [getYield](#) () const
- const [MeanValue_T](#) & [getMean](#) () const
- const [StdDevValue_T](#) & [getStdDev](#) () const
- const [BookingLimit_T](#) & [getCumulatedBookingLimit](#) () const
- const [ProtectionLevel_T](#) & [getCumulatedProtection](#) () const
- const [GeneratedDemandVector_T](#) & [getGeneratedDemandVector](#) () const
- void [setYield](#) (const [Yield_T](#) &iYield)
- void [setMean](#) (const [MeanValue_T](#) &iMean)
- void [setStdDev](#) (const [StdDevValue_T](#) &iStdDev)
- void [setCumulatedBookingLimit](#) (const [BookingLimit_T](#) &iBL)
- void [setCumulatedProtection](#) (const [ProtectionLevel_T](#) &iP)
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- const std::string [describe](#) () const
- [VirtualClassStruct](#) (const [VirtualClassStruct](#) &)
- [VirtualClassStruct](#) ([BookingClass](#) &)
- [~VirtualClassStruct](#) ()

34.140.1 Detailed Description

Structure holding the elements of a virtual class.

Definition at line 23 of file [VirtualClassStruct.hpp](#).

34.140.2 Constructor & Destructor Documentation

34.140.2.1 stdair::VirtualClassStruct::VirtualClassStruct (const VirtualClassStruct & iVC)

Constructor.

Definition at line 19 of file [VirtualClassStruct.cpp](#).

34.140.2.2 stdair::VirtualClassStruct::VirtualClassStruct (BookingClass & ioBookingClass)

Default copy constructor.

Definition at line 25 of file [VirtualClassStruct.cpp](#).

34.140.2.3 stdair::VirtualClassStruct::~~VirtualClassStruct ()

Destructor.

Definition at line 30 of file [VirtualClassStruct.cpp](#).

34.140.3 Member Function Documentation**34.140.3.1 const Yield_T& stdair::VirtualClassStruct::getYield () const [inline]**

Get the yield (average price paid for that virtual class).

Definition at line 27 of file [VirtualClassStruct.hpp](#).

Referenced by [stdair::LegCabin::displayVirtualClassList\(\)](#).

34.140.3.2 const MeanValue_T& stdair::VirtualClassStruct::getMean () const [inline]

Get the mean value of the demand distribution.

Definition at line 32 of file [VirtualClassStruct.hpp](#).

34.140.3.3 const StdDevValue_T& stdair::VirtualClassStruct::getStdDev () const [inline]

Get the standard deviation of the demand distribution.

Definition at line 37 of file [VirtualClassStruct.hpp](#).

34.140.3.4 const BookingLimit_T& stdair::VirtualClassStruct::getCumulatedBookingLimit () const [inline]

Get the booking limit of the class.

Definition at line 42 of file [VirtualClassStruct.hpp](#).

Referenced by [stdair::LegCabin::displayVirtualClassList\(\)](#).

34.140.3.5 const ProtectionLevel_T& stdair::VirtualClassStruct::getCumulatedProtection () const [inline]

Get the protection level of the class.

Definition at line 47 of file [VirtualClassStruct.hpp](#).

Referenced by [stdair::LegCabin::displayVirtualClassList\(\)](#).

34.140.3.6 **const GeneratedDemandVector_T &**
stdair::VirtualClassStruct::getGeneratedDemandVector ()
const

Get the generated demand sample vector for Monte-Carlo method.

Definition at line 53 of file [VirtualClassStruct.cpp](#).

References [stdair::BookingClass::getGeneratedDemandVector\(\)](#).

34.140.3.7 **void stdair::VirtualClassStruct::setYield (const Yield_T & iYield)** **[inline]**

Set the yield (average price paid for that virtual class).

Definition at line 57 of file [VirtualClassStruct.hpp](#).

34.140.3.8 **void stdair::VirtualClassStruct::setMean (const MeanValue_T & iMean)**
[inline]

Set the mean value of the demand distribution.

Definition at line 62 of file [VirtualClassStruct.hpp](#).

34.140.3.9 **void stdair::VirtualClassStruct::setStdDev (const StdDevValue_T & iStdDev)**
[inline]

Set the standard deviation of the demand distribution.

Definition at line 67 of file [VirtualClassStruct.hpp](#).

34.140.3.10 **void stdair::VirtualClassStruct::setCumulatedBookingLimit (const**
BookingLimit_T & iBL) **[inline]**

Set the booking limit of the class.

Definition at line 72 of file [VirtualClassStruct.hpp](#).

34.140.3.11 **void stdair::VirtualClassStruct::setCumulatedProtection (const**
ProtectionLevel_T & iP) **[inline]**

Set the protection level of the class.

Definition at line 77 of file [VirtualClassStruct.hpp](#).

34.140.3.12 **void stdair::VirtualClassStruct::toStream (std::ostream & ioOut) const**

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Reimplemented from [stdair::StructAbstract](#).

Definition at line 35 of file [VirtualClassStruct.cpp](#).

References [describe\(\)](#).

34.140.3.13 void stdair::VirtualClassStruct::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object from an input stream.

Parameters

<i>istream</i> & the input stream.

Reimplemented from [stdair::StructAbstract](#).

Definition at line 40 of file [VirtualClassStruct.cpp](#).

34.140.3.14 const std::string stdair::VirtualClassStruct::describe () const [virtual]

Display of the structure.

Implements [stdair::StructAbstract](#).

Definition at line 44 of file [VirtualClassStruct.cpp](#).

Referenced by [toStream\(\)](#).

The documentation for this struct was generated from the following files:

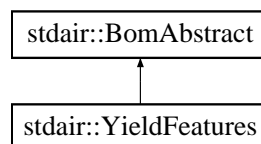
- [stdair/bom/VirtualClassStruct.hpp](#)
- [stdair/bom/VirtualClassStruct.cpp](#)

34.141 stdair::YieldFeatures Class Reference

Class representing the actual attributes for a yield date-period.

```
#include <stdair/bom/YieldFeatures.hpp>
```

Inheritance diagram for stdair::YieldFeatures:



Public Types

- typedef [YieldFeaturesKey](#) Key_T

Public Member Functions

- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- std::string [toString](#) () const
- const std::string [describeKey](#) () const

- const [Key_T](#) & [getKey](#) () const
- [BomAbstract](#) *const [getParent](#) () const
- const [HolderMap_T](#) & [getHolderMap](#) () const
- const [CabinCode_T](#) & [getCabinCode](#) () const
- const [TripType_T](#) & [getTripType](#) () const
- bool [isTripTypeValid](#) (const [TripType_T](#) &) const

Protected Member Functions

- [YieldFeatures](#) (const [Key_T](#) &)
- virtual [~YieldFeatures](#) ()

Protected Attributes

- [Key_T _key](#)
- [BomAbstract](#) * [_parent](#)
- [HolderMap_T _holderMap](#)

Friends

- class [FacBom](#)
- class [FacBomManager](#)

34.141.1 Detailed Description

Class representing the actual attributes for a yield date-period.

Definition at line 19 of file [YieldFeatures.hpp](#).

34.141.2 Member Typedef Documentation

34.141.2.1 typedef YieldFeaturesKey stdair::YieldFeatures::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 28 of file [YieldFeatures.hpp](#).

34.141.3 Constructor & Destructor Documentation

34.141.3.1 stdair::YieldFeatures::YieldFeatures (const Key_T & iKey) [protected]

Main constructor.

Definition at line 29 of file [YieldFeatures.cpp](#).

34.141.3.2 stdair::YieldFeatures::~~YieldFeatures () [protected, virtual]

Destructor.

Definition at line 34 of file [YieldFeatures.cpp](#).

34.141.4 Member Function Documentation

34.141.4.1 void stdair::YieldFeatures::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i>	the output stream.
---------------------	--------------------

Implements [stdair::BomAbstract](#).

Definition at line 37 of file [YieldFeatures.hpp](#).

References [toString\(\)](#).

34.141.4.2 void stdair::YieldFeatures::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters

<i>istream&</i>	the input stream.
---------------------	-------------------

Implements [stdair::BomAbstract](#).

Definition at line 46 of file [YieldFeatures.hpp](#).

34.141.4.3 std::string stdair::YieldFeatures::toString () const [virtual]

Get the serialised version of the Business Object.

Implements [stdair::BomAbstract](#).

Definition at line 38 of file [YieldFeatures.cpp](#).

References [describeKey\(\)](#).

Referenced by [toStream\(\)](#).

34.141.4.4 const std::string stdair::YieldFeatures::describeKey () const [inline]

Get a string describing the key.

Definition at line 57 of file [YieldFeatures.hpp](#).

References [_key](#), and [stdair::YieldFeaturesKey::toString\(\)](#).

Referenced by [toString\(\)](#).

34.141.4.5 **const Key_T& stdair::YieldFeatures::getKey () const** [\[inline\]](#)

Get the primary key (trip type, cabin code).

Definition at line 66 of file [YieldFeatures.hpp](#).

References [_key](#).

34.141.4.6 **BomAbstract* const stdair::YieldFeatures::getParent () const** [\[inline\]](#)

Get a reference on the parent object instance.

Definition at line 73 of file [YieldFeatures.hpp](#).

References [_parent](#).

34.141.4.7 **const HolderMap_T& stdair::YieldFeatures::getHolderMap () const**
[\[inline\]](#)

Get a reference on the children holder.

Definition at line 80 of file [YieldFeatures.hpp](#).

References [_holderMap](#).

34.141.4.8 **const CabinCode_T& stdair::YieldFeatures::getCabinCode () const**
[\[inline\]](#)

Get the cabin code.

Definition at line 87 of file [YieldFeatures.hpp](#).

References [_key](#), and [stdair::YieldFeaturesKey::getCabinCode\(\)](#).

34.141.4.9 **const TripType_T& stdair::YieldFeatures::getTripType () const** [\[inline\]](#)

Get the trip type.

Definition at line 94 of file [YieldFeatures.hpp](#).

References [_key](#), and [stdair::YieldFeaturesKey::getTripType\(\)](#).

Referenced by [isTripTypeValid\(\)](#).

34.141.4.10 **bool stdair::YieldFeatures::isTripTypeValid (const TripType_T &
iBookingRequestTripType) const**

Check whether the fare rule trip type corresponds to the booking request trip type.

Definition at line 46 of file [YieldFeatures.cpp](#).

References [getTripType\(\)](#), [stdair::TRIP_TYPE_INBOUND](#), [stdair::TRIP_TYPE_OUTBOUND](#), and [stdair::TRIP_TYPE_ROUND_TRIP](#).

34.141.5 Friends And Related Function Documentation

34.141.5.1 friend class FacBom [friend]

Definition at line 20 of file [YieldFeatures.hpp](#).

34.141.5.2 friend class FacBomManager [friend]

Definition at line 21 of file [YieldFeatures.hpp](#).

34.141.6 Member Data Documentation

34.141.6.1 Key_T stdair::YieldFeatures::_key [protected]

Primary key (flight number and departure date).

Definition at line 137 of file [YieldFeatures.hpp](#).

Referenced by [describeKey\(\)](#), [getCabinCode\(\)](#), [getKey\(\)](#), and [getTripType\(\)](#).

34.141.6.2 BomAbstract* stdair::YieldFeatures::_parent [protected]

Pointer on the parent class.

Definition at line 142 of file [YieldFeatures.hpp](#).

Referenced by [getParent\(\)](#).

34.141.6.3 HolderMap_T stdair::YieldFeatures::_holderMap [protected]

Map holding the children.

Definition at line 147 of file [YieldFeatures.hpp](#).

Referenced by [getHolderMap\(\)](#).

The documentation for this class was generated from the following files:

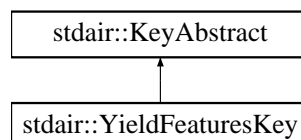
- [stdair/bom/YieldFeatures.hpp](#)
- [stdair/bom/YieldFeatures.cpp](#)

34.142 stdair::YieldFeaturesKey Struct Reference

Key of date-period.

```
#include <stdair/bom/YieldFeaturesKey.hpp>
```

Inheritance diagram for stdair::YieldFeaturesKey:



Public Member Functions

- [YieldFeaturesKey](#) (const [TripType_T](#) &, const [CabinCode_T](#) &)
- [YieldFeaturesKey](#) (const [YieldFeaturesKey](#) &)
- [~YieldFeaturesKey](#) ()
- const [TripType_T](#) & [getTripType](#) () const
- const [CabinCode_T](#) & [getCabinCode](#) () const
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- const std::string [toString](#) () const

34.142.1 Detailed Description

Key of date-period.

Definition at line 18 of file [YieldFeaturesKey.hpp](#).

34.142.2 Constructor & Destructor Documentation

34.142.2.1 stdair::YieldFeaturesKey::YieldFeaturesKey (const [TripType_T](#) & *iTripType*, const [CabinCode_T](#) & *iCabin*)

Main constructor.

Definition at line 21 of file [YieldFeaturesKey.cpp](#).

34.142.2.2 stdair::YieldFeaturesKey::YieldFeaturesKey (const [YieldFeaturesKey](#) & *iKey*)

Copy constructor.

Definition at line 27 of file [YieldFeaturesKey.cpp](#).

34.142.2.3 stdair::YieldFeaturesKey::~~YieldFeaturesKey ()

Destructor.

Definition at line 32 of file [YieldFeaturesKey.cpp](#).

34.142.3 Member Function Documentation

34.142.3.1 const [TripType_T](#)& stdair::YieldFeaturesKey::getTripType () const `[inline]`

Get the fare trip type.

Definition at line 44 of file [YieldFeaturesKey.hpp](#).

Referenced by [stdair::YieldFeatures::getTripType\(\)](#).

34.142.3.2 `const CabinCode_T& stdair::YieldFeaturesKey::getCabinCode () const`
`[inline]`

Get the cabin.

Definition at line 51 of file [YieldFeaturesKey.hpp](#).

Referenced by [stdair::YieldFeatures::getCabinCode\(\)](#).

34.142.3.3 `void stdair::YieldFeaturesKey::toStream (std::ostream & ioOut) const`
`[virtual]`

Dump a Business Object Key into an output stream.

Parameters

<code>ostream&</code> the output stream.
--

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 36 of file [YieldFeaturesKey.cpp](#).

References [toString\(\)](#).

34.142.3.4 `void stdair::YieldFeaturesKey::fromStream (std::istream & ioIn)` `[virtual]`

Read a Business Object Key from an input stream.

Parameters

<code>istream&</code> the input stream.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 41 of file [YieldFeaturesKey.cpp](#).

34.142.3.5 `const std::string stdair::YieldFeaturesKey::toString () const` `[virtual]`

Get the serialised version of the Business Object Key. That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 45 of file [YieldFeaturesKey.cpp](#).

Referenced by [stdair::YieldFeatures::describeKey\(\)](#), and [toStream\(\)](#).

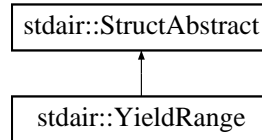
The documentation for this struct was generated from the following files:

- [stdair/bom/YieldFeaturesKey.hpp](#)
- [stdair/bom/YieldFeaturesKey.cpp](#)

34.143 stdair::YieldRange Class Reference

```
#include <stdair/basic/YieldRange.hpp>
```

Inheritance diagram for stdair::YieldRange:



Public Member Functions

- [YieldRange](#) ()
- [YieldRange](#) (const [YieldRange](#) &)
- [YieldRange](#) (const [Yield_T](#) iUpperYield)
- [YieldRange](#) (const [Yield_T](#) iUpperYield, const [Yield_T](#) iAverageYield)
- [YieldRange](#) (const [Yield_T](#) iUpperYield, const [Yield_T](#) iAverageYield, const [Yield_T](#) iLowerYield)
- virtual [~YieldRange](#) ()
- [Yield_T](#) [getUpperYield](#) () const
- [Yield_T](#) [getAverageYield](#) () const
- [Yield_T](#) [getLowerYield](#) () const
- void [setUpperYield](#) (const [Yield_T](#) iUpperYield)
- void [setAverageYield](#) (const [Yield_T](#) iAverageYield)
- void [setLowerYield](#) (const [Yield_T](#) iLowerYield)
- void [toStream](#) (std::ostream &) const
- void [fromStream](#) (std::istream &)
- const std::string [describe](#) () const

34.143.1 Detailed Description

Class representing a range of yields.

Typically, bookings are priced according to rules (e.g., fare rules), leading to slight variations of revenues for a given product. The "yield range" captures the extent of revenues earned for a given product.

When no average and lower yields are defined, they are assumed to be equal to the upper yield.

Note that the lower yield is generally not defined, as it corresponds to the upper yield of the lower yield range.

Definition at line 23 of file [YieldRange.hpp](#).

34.143.2 Constructor & Destructor Documentation

34.143.2.1 stdair::YieldRange::YieldRange ()

Constructors.

Definition at line 13 of file [YieldRange.cpp](#).

34.143.2.2 `stdair::YieldRange::YieldRange (const YieldRange & iYieldRange)`

Definition at line 20 of file [YieldRange.cpp](#).

34.143.2.3 `stdair::YieldRange::YieldRange (const Yield_T iUpperYield)`

Definition at line 27 of file [YieldRange.cpp](#).

34.143.2.4 `stdair::YieldRange::YieldRange (const Yield_T iUpperYield, const Yield_T iAverageYield)`

Definition at line 33 of file [YieldRange.cpp](#).

34.143.2.5 `stdair::YieldRange::YieldRange (const Yield_T iUpperYield, const Yield_T iAverageYield, const Yield_T iLowerYield)`

Definition at line 40 of file [YieldRange.cpp](#).

34.143.2.6 `stdair::YieldRange::~YieldRange () [virtual]`

Constructors.

Definition at line 48 of file [YieldRange.cpp](#).

34.143.3 Member Function Documentation

34.143.3.1 `Yield_T stdair::YieldRange::getUpperYield () const [inline]`

Getter for the upper yield of the range.

Definition at line 39 of file [YieldRange.hpp](#).

34.143.3.2 `Yield_T stdair::YieldRange::getAverageYield () const [inline]`

Getter for the average yield of the range.

Definition at line 43 of file [YieldRange.hpp](#).

34.143.3.3 `Yield_T stdair::YieldRange::getLowerYield () const [inline]`

Getter for the lower yield of the range.

Definition at line 47 of file [YieldRange.hpp](#).

34.143.3.4 `void stdair::YieldRange::setUpperYield (const Yield_T iUpperYield) [inline]`

Setter for the upper yield of the range.

Definition at line 53 of file [YieldRange.hpp](#).

34.143.3.5 void stdair::YieldRange::setAverageYield (const Yield_T iAverageYield)
[inline]

Setter for the average yield of the range.

Definition at line 57 of file [YieldRange.hpp](#).

34.143.3.6 void stdair::YieldRange::setLowerYield (const Yield_T iLowerYield)
[inline]

Setter for the lower yield of the range.

Definition at line 61 of file [YieldRange.hpp](#).

34.143.3.7 void stdair::YieldRange::toStream (std::ostream & ioOut) const

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Reimplemented from [stdair::StructAbstract](#).

Definition at line 52 of file [YieldRange.cpp](#).

34.143.3.8 void stdair::YieldRange::fromStream (std::istream & ioIn) [virtual]

Read a Business Object from an input stream.

Parameters

<i>istream&</i> the input stream.

Reimplemented from [stdair::StructAbstract](#).

Definition at line 58 of file [YieldRange.cpp](#).

34.143.3.9 const std::string stdair::YieldRange::describe () const [virtual]

Display of the structure.

Implements [stdair::StructAbstract](#).

Definition at line 62 of file [YieldRange.cpp](#).

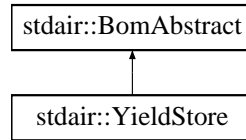
The documentation for this class was generated from the following files:

- [stdair/basic/YieldRange.hpp](#)
- [stdair/basic/YieldRange.cpp](#)

34.144 stdair::YieldStore Class Reference

```
#include <stdair/bom/YieldStore.hpp>
```

Inheritance diagram for stdair::YieldStore:



Public Types

- typedef [YieldStoreKey](#) [Key_T](#)

Public Member Functions

- void [toStream](#) (std::ostream &ioOut) const
- [BomAbstract](#) *const [getParent](#) () const
- void [fromStream](#) (std::istream &ioIn)
- std::string [toString](#) () const
- const std::string [describeKey](#) () const
- const [Key_T](#) & [getKey](#) () const
- const [AirlineCode_T](#) & [getAirlineCode](#) () const

Protected Member Functions

- [YieldStore](#) (const [Key_T](#) &)
- [YieldStore](#) (const [YieldStore](#) &)
- [~YieldStore](#) ()

Protected Attributes

- [Key_T _key](#)
- [BomAbstract](#) * [_parent](#)

Friends

- class [FacBom](#)
- class [FacBomManager](#)

34.144.1 Detailed Description

Class representing the actual attributes for an airline [YieldStore](#).

Definition at line 18 of file [YieldStore.hpp](#).

34.144.2 Member Typedef Documentation

34.144.2.1 typedef YieldStoreKey stdair::YieldStore::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 25 of file [YieldStore.hpp](#).

34.144.3 Constructor & Destructor Documentation

34.144.3.1 stdair::YieldStore::YieldStore (const Key_T & iKey) [protected]

Default constructors.

Definition at line 13 of file [YieldStore.cpp](#).

34.144.3.2 stdair::YieldStore::YieldStore (const YieldStore &) [protected]

34.144.3.3 stdair::YieldStore::~~YieldStore () [protected]

Destructor.

Definition at line 17 of file [YieldStore.cpp](#).

34.144.4 Member Function Documentation

34.144.4.1 void stdair::YieldStore::toStream (std::ostream & ioOut) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Implements [stdair::BomAbstract](#).

Definition at line 31 of file [YieldStore.hpp](#).

References [toString\(\)](#).

34.144.4.2 BomAbstract* const stdair::YieldStore::getParent () const [inline]

Get the parent object.

Definition at line 34 of file [YieldStore.hpp](#).

References [_parent](#).

34.144.4.3 void stdair::YieldStore::fromStream (std::istream & iIn) [inline, virtual]

Read a Business Object from an input stream.

Parameters

<i>istream&</i> the input stream.

Implements [stdair::BomAbstract](#).

Definition at line 38 of file [YieldStore.hpp](#).

34.144.4.4 std::string stdair::YieldStore::toString () const [virtual]

Get the serialised version of the Business Object.

Implements [stdair::BomAbstract](#).

Definition at line 21 of file [YieldStore.cpp](#).

References [_key](#), and [stdair::YieldStoreKey::toString\(\)](#).

Referenced by [toStream\(\)](#).

34.144.4.5 const std::string stdair::YieldStore::describeKey () const [inline]

Get a string describing the key.

Definition at line 44 of file [YieldStore.hpp](#).

References [_key](#), and [stdair::YieldStoreKey::toString\(\)](#).

34.144.4.6 const Key_T& stdair::YieldStore::getKey () const [inline]

Get the [YieldStore](#) key.

Definition at line 49 of file [YieldStore.hpp](#).

References [_key](#).

34.144.4.7 const AirlineCode_T& stdair::YieldStore::getAirlineCode () const [inline]

Get the airline code.

Definition at line 52 of file [YieldStore.hpp](#).

References [_key](#), and [stdair::YieldStoreKey::getAirlineCode\(\)](#).

34.144.5 Friends And Related Function Documentation**34.144.5.1 friend class FacBom [friend]**

Definition at line 19 of file [YieldStore.hpp](#).

34.144.5.2 friend class FacBomManager [friend]

Definition at line 20 of file [YieldStore.hpp](#).

34.144.6 Member Data Documentation

34.144.6.1 **Key_T** stdair::YieldStore::_key [protected]

The key of both structure and objects.

Definition at line 66 of file [YieldStore.hpp](#).

Referenced by [describeKey\(\)](#), [getAirlineCode\(\)](#), [getKey\(\)](#), and [toString\(\)](#).

34.144.6.2 **BomAbstract*** stdair::YieldStore::_parent [protected]

Definition at line 67 of file [YieldStore.hpp](#).

Referenced by [getParent\(\)](#).

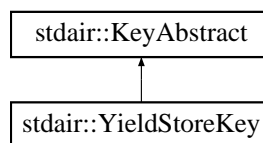
The documentation for this class was generated from the following files:

- [stdair/bom/YieldStore.hpp](#)
- [stdair/bom/YieldStore.cpp](#)

34.145 stdair::YieldStoreKey Struct Reference

```
#include <stdair/bom/YieldStoreKey.hpp>
```

Inheritance diagram for stdair::YieldStoreKey:



Public Member Functions

- [YieldStoreKey](#) (const [AirlineCode_T](#) &iAirlineCode)
- [YieldStoreKey](#) (const [YieldStoreKey](#) &)
- [~YieldStoreKey](#) ()
- const [AirlineCode_T](#) & [getAirlineCode](#) () const
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- const std::string [toString](#) () const

34.145.1 Detailed Description

Key of [YieldStore](#).

Definition at line 14 of file [YieldStoreKey.hpp](#).

34.145.2 Constructor & Destructor Documentation**34.145.2.1 stdair::YieldStoreKey::YieldStoreKey (const AirlineCode_T & iAirlineCode)**

Constructors.

Definition at line 10 of file [YieldStoreKey.cpp](#).

34.145.2.2 stdair::YieldStoreKey::YieldStoreKey (const YieldStoreKey & iKey)

Definition at line 14 of file [YieldStoreKey.cpp](#).

34.145.2.3 stdair::YieldStoreKey::~YieldStoreKey ()

Destructor.

Definition at line 19 of file [YieldStoreKey.cpp](#).

34.145.3 Member Function Documentation**34.145.3.1 const AirlineCode_T& stdair::YieldStoreKey::getAirlineCode () const**
[inline]

Get the airline code.

Definition at line 30 of file [YieldStoreKey.hpp](#).

Referenced by [stdair::YieldStore::getAirlineCode\(\)](#).

34.145.3.2 void stdair::YieldStoreKey::toStream (std::ostream & ioOut) const [virtual]

Dump a Business Object Key into an output stream.

Parameters

<i>ostream&</i> the output stream.
--

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 23 of file [YieldStoreKey.cpp](#).

References [toString\(\)](#).

34.145.3.3 void stdair::YieldStoreKey::fromStream (std::istream & ioIn) [virtual]

Read a Business Object Key from an input stream.

Parameters

<i>istream&</i> the input stream.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 28 of file [YieldStoreKey.cpp](#).

34.145.3.4 `const std::string stdair::YieldStoreKey::toString () const` `[virtual]`

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 32 of file [YieldStoreKey.cpp](#).

Referenced by [stdair::YieldStore::describeKey\(\)](#), [toStream\(\)](#), and [stdair::YieldStore::toString\(\)](#).

The documentation for this struct was generated from the following files:

- [stdair/bom/YieldStoreKey.hpp](#)
- [stdair/bom/YieldStoreKey.cpp](#)

35 File Documentation

35.1 [batches/stdair.cpp](#) File Reference

35.2 [stdair.cpp](#)

```

00001
00005 // STL
00006 #include <cassert>
00007 #include <iostream>
00008 #include <sstream>
00009 #include <fstream>
00010 #include <string>
00011 // Boost (Extended STL)
00012 #include <boost/date_time/posix_time/posix_time.hpp>
00013 #include <boost/date_time/gregorian/gregorian.hpp>
00014 #include <boost/program_options.hpp>
00015 #include <boost/tokenizer.hpp>
00016 #include <boost/lexical_cast.hpp>
00017 // StdAir
00018 #include <stdair/stdair_types.hpp>
00019 #include <stdair/bom/BomArchive.hpp>
00020 #include <stdair/bom/BookingRequestStruct.hpp>
00021 #include <stdair/bom/TravelSolutionStruct.hpp>
00022 #include <stdair/service/Logger.hpp>
00023 #include <stdair/STDAIR_Service.hpp>
00024 #include <stdair/config/stdair-paths.hpp>
00025
00026 // ////////// Constants //////////
00030 const std::string K_STDAIR_DEFAULT_LOG_FILENAME ("stdair.log");
00031
00035 const std::string K_STDAIR_DEFAULT_INPUT_FILENAME (STDAIR_SAMPLE_DIR
00036                                                    "/schedule01.csv");
00037
00042 const bool K_STDAIR_DEFAULT_BUILT_IN_INPUT = false;
00043

```

```

00049 const bool K_STDAIR_DEFAULT_BUILT_FOR_RMOl = false;
00050
00056 const bool K_STDAIR_DEFAULT_BUILT_FOR_CRS = false;
00057
00062 const int K_STDAIR_EARLY_RETURN_STATUS = 99;
00063
00064 // ////////// Parsing of Options & Configuration //////////
00065 // A helper function to simplify the main part.
00066 template<class T> std::ostream& operator<< (std::ostream& os,
00067                                           const std::vector<T>& v) {
00068     std::copy (v.begin(), v.end(), std::ostream_iterator<T> (std::cout, " "));
00069     return os;
00070 }
00071
00073 int readConfiguration (int argc, char* argv[], bool& ioIsBuiltin,
00074                       bool& ioIsForRMOl, bool& ioIsForCRS,
00075                       stdair::Filename_T& ioInputFilename,
00076                       std::string& ioLogFilename) {
00077     // Default for the built-in input
00078     ioIsBuiltin = K_STDAIR_DEFAULT_BUILT_IN_INPUT;
00079
00080     // Default for the RMOl input
00081     ioIsForRMOl = K_STDAIR_DEFAULT_BUILT_FOR_RMOl;
00082
00083     // Default for the CRS input
00084     ioIsForCRS = K_STDAIR_DEFAULT_BUILT_FOR_CRS;
00085
00086     // Declare a group of options that will be allowed only on command line
00087     boost::program_options::options_description generic ("Generic options");
00088     generic.add_options()
00089         ("prefix", "print installation prefix")
00090         ("version,v", "print version string")
00091         ("help,h", "produce help message");
00092
00093     // Declare a group of options that will be allowed both on command
00094     // line and in config file
00095
00096     boost::program_options::options_description config ("Configuration");
00097     config.add_options()
00098         ("builtin,b",
00099          "The sample BOM tree can be either built-in or parsed from an input file. Th
00100          at latter must then be given with the -i/--input option")
00101         ("rmol,r",
00102          "Build a sample BOM tree for RMOl (i.e., a dummy flight-date with a single l
00103          eg-cabin)")
00104         ("crs,c",
00105          "Build a sample BOM tree for CRS")
00106         ("input,i",
00107          boost::program_options::value< std::string >(&ioInputFilename)->default_valu
00108          e(K_STDAIR_DEFAULT_INPUT_FILENAME),
00109          "(CVS) input file for the demand distributions")
00110         ("log,l",
00111          boost::program_options::value< std::string >(&ioLogFilename)->default_value(
00112          K_STDAIR_DEFAULT_LOG_FILENAME),
00113          "Filename for the logs")
00114         ;
00115
00116     // Hidden options, will be allowed both on command line and
00117     // in config file, but will not be shown to the user.
00118     boost::program_options::options_description hidden ("Hidden options");
00119     hidden.add_options()
00120         ("copyright",

```

```

00117     boost::program_options::value< std::vector<std::string> >(),
00118     "Show the copyright (license)");
00119
00120     boost::program_options::options_description cmdline_options;
00121     cmdline_options.add(generic).add(config).add(hidden);
00122
00123     boost::program_options::options_description config_file_options;
00124     config_file_options.add(config).add(hidden);
00125     boost::program_options::options_description visible ("Allowed options");
00126     visible.add(generic).add(config);
00127
00128     boost::program_options::positional_options_description p;
00129     p.add ("copyright", -1);
00130
00131     boost::program_options::variables_map vm;
00132     boost::program_options::
00133     store (boost::program_options::command_line_parser (argc, argv).
00134     options (cmdline_options).positional(p).run(), vm);
00135
00136     std::ifstream ifs ("stdair.cfg");
00137     boost::program_options::store (parse_config_file (ifs, config_file_options),
00138     vm);
00139     boost::program_options::notify (vm);
00140
00141     if (vm.count ("help")) {
00142         std::cout << visible << std::endl;
00143         return K_STDAIR_EARLY_RETURN_STATUS;
00144     }
00145
00146     if (vm.count ("version")) {
00147         std::cout << PACKAGE_NAME << ", version " << PACKAGE_VERSION << std::endl;
00148         return K_STDAIR_EARLY_RETURN_STATUS;
00149     }
00150
00151     if (vm.count ("prefix")) {
00152         std::cout << "Installation prefix: " << PREFIXDIR << std::endl;
00153         return K_STDAIR_EARLY_RETURN_STATUS;
00154     }
00155
00156     if (vm.count ("builtin")) {
00157         ioIsBuiltin = true;
00158     }
00159
00160     if (vm.count ("rmol")) {
00161         ioIsForRMOL = true;
00162
00163         // The RMOL sample tree takes precedence over the default built-in BOM tree
00164         ioIsBuiltin = false;
00165     }
00166
00167     if (vm.count ("crs")) {
00168         ioIsForCRS = true;
00169
00170         // The RMOL sample tree takes precedence over the default built-in BOM tree
00171         ioIsBuiltin = false;
00172     }
00173
00174     const std::string isBuiltinStr = (ioIsBuiltin == true)?"yes":"no";
00175     std::cout << "The BOM should be built-in? " << isBuiltinStr << std::endl;
00176
00177     const std::string isForRMOLStr = (ioIsForRMOL == true)?"yes":"no";
00178     std::cout << "The BOM should be built-in for RMOL? " << isForRMOLStr

```

```

00179         << std::endl;
00180
00181     const std::string isForCRSStr = (ioIsForCRS == true)?"yes":"no";
00182     std::cout << "The BOM should be built-in for CRS? " << isForCRSStr
00183         << std::endl;
00184
00185     if (ioIsBuiltin == false && ioIsForRMOL == false && ioIsForCRS == false) {
00186         if (vm.count ("input")) {
00187             ioInputFilename = vm["input"].as< std::string >();
00188             std::cout << "Input filename is: " << ioInputFilename << std::endl;
00189         } else {
00190             std::cerr << "Either one among the -b/--builtin, -r/--rmol, -c/--crs "
00191                 << "or -i/--input options must be specified" << std::endl;
00192         }
00193     }
00194 }
00195
00196 if (vm.count ("log")) {
00197     ioLogFilename = vm["log"].as< std::string >();
00198     std::cout << "Log filename is: " << ioLogFilename << std::endl;
00199 }
00200
00201 return 0;
00202 }
00203
00204
00205 // ////////////////////////////////// M A I N //////////////////////////////////
00206 int main (int argc, char* argv[]) {
00207
00208     // State whether the BOM tree should be built-in or parsed from an
00209     // input file
00210     bool isBuiltin;
00211
00212     // State whether a sample BOM tree should be built for RMOL.
00213     bool isForRMOL;
00214
00215     // State whether a sample BOM tree should be built for the CRS.
00216     bool isForCRS;
00217
00218     // Input file name
00219     stdair::Filename_T lInputFilename;
00220
00221     // Output log File
00222     std::string lLogFilename;
00223
00224     // Call the command-line option parser
00225     const int lOptionParserStatus =
00226         readConfiguration (argc, argv, isBuiltin, isForRMOL, isForCRS,
00227             lInputFilename, lLogFilename);
00228
00229     if (lOptionParserStatus == K_STDAIR_EARLY_RETURN_STATUS) {
00230         return 0;
00231     }
00232
00233     // Set the log parameters
00234     std::ofstream logOutputFile;
00235     // Open and clean the log outputfile
00236     logOutputFile.open (lLogFilename.c_str());
00237     logOutputFile.clear();
00238
00239     const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
00240     stdair::STDAIR_Service stdairService (lLogParams);

```

```

00241
00242 // DEBUG
00243 STDAIR_LOG_DEBUG ("Welcome to stdair");
00244
00245 // Check whether or not a (CSV) input file should be read
00246 if (isBuiltin == true || isForRMOL == true || isForCRS == true) {
00247
00248     if (isForRMOL == true) {
00249         // Build the sample BOM tree for RMOL
00250         stdairService.buildDummyInventory (300);
00251
00252     } else if (isForCRS == true) {
00253         //
00254         stdair::TravelSolutionList_T lTravelSolutionList;
00255         stdairService.buildSampleTravelSolutions (lTravelSolutionList);
00256
00257         // Build the sample BOM tree for CRS
00258         const stdair::BookingRequestStruct& lBookingRequest =
00259             stdairService.buildSampleBookingRequest();
00260
00261         // DEBUG: Display the travel solution and booking request
00262         STDAIR_LOG_DEBUG ("Booking request: " << lBookingRequest.display());
00263
00264         const std::string& lCSVDump =
00265             stdairService.csvDisplay (lTravelSolutionList);
00266         STDAIR_LOG_DEBUG (lCSVDump);
00267
00268     } else {
00269         assert (isBuiltin == true);
00270
00271         // Build a sample BOM tree
00272         stdairService.buildSampleBom();
00273     }
00274
00275 } else {
00276     // Read the input file
00277     //stdairService.readFromInputFile (lInputFilename);
00278
00279     // DEBUG
00280     STDAIR_LOG_DEBUG ("StdAir will parse " << lInputFilename
00281         << " and build the corresponding BOM tree.");
00282 }
00283
00284 // DEBUG: Display the whole BOM tree
00285 const std::string& lCSVDump = stdairService.csvDisplay();
00286 STDAIR_LOG_DEBUG (lCSVDump);
00287
00288 // Close the Log outputFile
00289 logOutputFile.close();
00290
00291 /*
00292 Note: as that program is not intended to be run on a server in
00293 production, it is better not to catch the exceptions. When it
00294 happens (that an exception is throwned), that way we get the
00295 call stack.
00296 */
00297
00298 return 0;
00299 }
00300

```

35.3 doc/local/authors.doc File Reference

35.4 doc/local/codingrules.doc File Reference

35.5 doc/local/copyright.doc File Reference

35.6 doc/local/documentation.doc File Reference

35.7 doc/local/features.doc File Reference

35.8 doc/local/help_wanted.doc File Reference

35.9 doc/local/howto_release.doc File Reference

35.10 doc/local/index.doc File Reference

35.11 doc/local/installation.doc File Reference

35.12 doc/local/linking.doc File Reference

35.13 doc/local/test.doc File Reference

35.14 doc/local/users_guide.doc File Reference

35.15 doc/local/verification.doc File Reference

35.16 doc/tutorial/tutorial.doc File Reference

35.17 stdair/basic/BasChronometer.cpp File Reference

```
#include <cassert>
#include <stdair/basic/BasChronometer.hpp>
```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

35.18 BasChronometer.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
```

```

00006 // Stdair
00007 #include <stdair/basic/BasChronometer.hpp>
00008
00009 namespace stdair {
00010
00011 // //////////////////////////////////////
00012 BasChronometer::BasChronometer () : _startTimeLaunched (false) {
00013 }
00014
00015 // //////////////////////////////////////
00016 void BasChronometer::start () {
00017     // Get the time-stamp of now, and store it for later use
00018     _startTime = boost::posix_time::microsec_clock::local_time();
00019
00020     // Update the boolean which states whether the chronometer
00021     // is launched
00022     _startTimeLaunched = true;
00023 }
00024
00025 // //////////////////////////////////////
00026 double BasChronometer::elapsed () const {
00027     assert (_startTimeLaunched == true);
00028
00029     // Get the time-stamp of now
00030     const boost::posix_time::ptime lStopTime =
00031         boost::posix_time::microsec_clock::local_time();
00032
00033     // Calculate the time elapsed since the last time-stamp
00034     const boost::posix_time::time_duration lElapsedTime =
00035         lStopTime - _startTime;
00036
00037     // Derived the corresponding number of milliseconds
00038     const double lElapsedTimeInMicroSeconds =
00039         static_cast<const double> (lElapsedTime.total_microseconds());
00040
00041     // The elapsed time given in return is expressed in seconds
00042     return (lElapsedTimeInMicroSeconds / 1e6);
00043 }
00044
00045 }

```

35.19 stdair/basic/BasChronometer.hpp File Reference

```
#include <boost/date_time/posix_time/posix_time.hpp>
```

Classes

- struct [stdair::BasChronometer](#)

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.20 BasChronometer.hpp

```

00001 #ifndef __STDAIR_BAS_BASCHRONOMETER_HPP
00002 #define __STDAIR_BAS_BASCHRONOMETER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // Boost (STL Extension)
00008 // Boost Date-Time (http://boost.org/doc/html/date_time/posix_time.html)
00009 #include <boost/date_time/posix_time/posix_time.hpp>
00010
00011 namespace stdair {
00012
00014     struct BasChronometer {
00016         BasChronometer();
00017
00021         void start ();
00022
00024         std::string getStart () const {
00025             return boost::posix_time::to_simple_string (_startTime);
00026         }
00027
00030         double elapsed () const;
00031
00032     private:
00034         boost::posix_time::ptime _startTime;
00035
00037         bool _startTimeLaunched;
00038     };
00039
00040 }
00041 #endif // __STDAIR_BAS_BASCHRONOMETER_HPP

```

35.21 stdair/basic/BasConst.cpp File Reference

```

#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/basic/BasConst_Event.hpp>
#include <stdair/basic/BasConst_Request.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_BookingClass.hpp>
#include <stdair/basic/BasConst_Yield.hpp>
#include <stdair/basic/BasConst_DefaultObject.hpp>
#include <stdair/basic/BasConst_Period_BOM.hpp>
#include <stdair/basic/BasConst_TravelSolution.hpp>

```

Namespaces

- namespace `stdair`

Handle on the StdAir library context.

Functions

- const std::string [stdair::DEFAULT_BOM_ROOT_KEY](#) (" -- ROOT -- ")
- const double [stdair::DEFAULT_EPSILON_VALUE](#) (0.0001)
- const unsigned int [stdair::DEFAULT_FLIGHT_SPEED](#) (900)
- const NbOfFlightDates_T [stdair::DEFAULT_NB_OF_FLIGHTDATES](#) (0.0)
- const Duration_T [stdair::NULL_BOOST_TIME_DURATION](#) (-1,-1,-1)
- const unsigned int [stdair::DEFAULT_NB_OF_DAYS_IN_A_YEAR](#) (365)
- const unsigned int [stdair::DEFAULT_NUMBER_OF_SUBDIVISIONS](#) (1000)
- const DayDuration_T [stdair::DEFAULT_DAY_DURATION](#) (0)
- const DatePeriod_T [stdair::BOOST_DEFAULT_DATE_PERIOD](#) (Date_T(2007, 1, 1), Date_T(2007, 1, 1))
- const DOW_String_T [stdair::DEFAULT_DOW_STRING](#) ("00000000")
- const DateOffset_T [stdair::DEFAULT_DATE_OFFSET](#) (0)
- const Date_T [stdair::DEFAULT_DATE](#) (2010, boost::gregorian::Jan, 1)
- const DateTime_T [stdair::DEFAULT_DATETIME](#) (DEFAULT_DATE, NULL_BOOST_TIME_DURATION)
- const Duration_T [stdair::DEFAULT_EPSILON_DURATION](#) (0, 0, 0, 1)
- const Count_T [stdair::SECONDS_IN_ONE_DAY](#) (86400)
- const Count_T [stdair::MILLISECONDS_IN_ONE_SECOND](#) (1000)
- const RandomSeed_T [stdair::DEFAULT_RANDOM_SEED](#) (120765987)
- const AirportCode_T [stdair::AIRPORT_LHR](#) ("LHR")
- const AirportCode_T [stdair::AIRPORT_SYD](#) ("SYD")
- const CityCode_T [stdair::POS_LHR](#) ("LHR")
- const Date_T [stdair::DATE_20110115](#) (2011, boost::gregorian::Jan, 15)
- const Date_T [stdair::DATE_20111231](#) (2011, boost::gregorian::Dec, 31)
- const DayDuration_T [stdair::NO_ADVANCE_PURCHASE](#) (0)
- const SaturdayStay_T [stdair::SATURDAY_STAY](#) (true)
- const SaturdayStay_T [stdair::NO_SATURDAY_STAY](#) (false)
- const ChangeFees_T [stdair::CHANGE_FEES](#) (true)
- const ChangeFees_T [stdair::NO_CHANGE_FEES](#) (false)
- const NonRefundable_T [stdair::NON_REFUNDABLE](#) (true)
- const NonRefundable_T [stdair::No_NON_REFUNDABLE](#) (false)
- const SaturdayStay_T [stdair::DEFAULT_BOM_TREE_SATURDAY_STAY](#) (true)
- const ChangeFees_T [stdair::DEFAULT_BOM_TREE_CHANGE_FEES](#) (true)
- const NonRefundable_T [stdair::DEFAULT_BOM_TREE_NON_REFUNDABLE](#) (true)
- const DayDuration_T [stdair::NO_STAY_DURATION](#) (0)
- const AirlineCode_T [stdair::AIRLINE_CODE_BA](#) ("BA")
- const CabinCode_T [stdair::CABIN_Y](#) ("Y")
- const ClassCode_T [stdair::CLASS_CODE_Y](#) ("Y")
- const ClassCode_T [stdair::CLASS_CODE_Q](#) ("Q")
- const AirportCode_T [stdair::AIRPORT_SIN](#) ("SIN")
- const AirportCode_T [stdair::AIRPORT_BKK](#) ("BKK")
- const CityCode_T [stdair::POS_SIN](#) ("SIN")

- const CabinCode_T [stdair::CABIN_ECO](#) ("Eco")
- const FrequentFlyer_T [stdair::FREQUENT_FLYER_MEMBER](#) ("M")
- const ClassCode_T [stdair::DEFAULT_FAMILY_CODE](#) ("0")
- const NbOfAirlines_T [stdair::DEFAULT_NBOFAIRLINES](#) (0)
- const FlightPathCode_T [stdair::DEFAULT_FLIGHTPATH_CODE](#) ("")
- const Distance_T [stdair::DEFAULT_DISTANCE_VALUE](#) (0)
- const ClassCode_T [stdair::DEFAULT_CLOSED_CLASS_CODE](#) ("CC")
- const NbOfBookings_T [stdair::DEFAULT_CLASS_NB_OF_BOOKINGS](#) (0)
- const NbOfBookings_T [stdair::DEFAULT_CLASS_TOTAL_NB_OF_BOOKINGS](#) (0)
- const NbOfBookings_T [stdair::DEFAULT_CLASS_UNCONSTRAINED_DEMAND](#) (0)
- const NbOfBookings_T [stdair::DEFAULT_CLASS_REMAINING_DEMAND_MEAN](#) (0)
- const NbOfBookings_T [stdair::DEFAULT_CLASS_REMAINING_DEMAND_STANDARD_DEVIATION](#) (0)
- const NbOfCancellations_T [stdair::DEFAULT_CLASS_NB_OF_CANCELLATIONS](#) (0)
- const NbOfNoShows_T [stdair::DEFAULT_CLASS_NB_OF_NOSHOWS](#) (0)
- const CabinCapacity_T [stdair::DEFAULT_CABIN_CAPACITY](#) (100.0)
- const CommittedSpace_T [stdair::DEFAULT_COMMITTED_SPACE](#) (0.0)
- const BlockSpace_T [stdair::DEFAULT_BLOCK_SPACE](#) (0.0)
- const Availability_T [stdair::DEFAULT_NULL_AVAILABILITY](#) (0.0)
- const Availability_T [stdair::DEFAULT_AVAILABILITY](#) (9.0)
- const Availability_T [stdair::MAXIMAL_AVAILABILITY](#) (9999.0)
- const CensorshipFlag_T [stdair::DEFAULT_CLASS_CENSORSHIPFLAG](#) (false)
- const BookingLimit_T [stdair::DEFAULT_CLASS_BOOKING_LIMIT](#) (9999.0)
- const AuthorizationLevel_T [stdair::DEFAULT_CLASS_AUTHORIZATION_LEVEL](#) (9999.0)
- const AuthorizationLevel_T [stdair::DEFAULT_CLASS_MAX_AUTHORIZATION_LEVEL](#) (9999.0)
- const AuthorizationLevel_T [stdair::DEFAULT_CLASS_MIN_AUTHORIZATION_LEVEL](#) (0.0)
- const OverbookingRate_T [stdair::DEFAULT_CLASS_OVERBOOKING_RATE](#) (0.0)
- const BookingRatio_T [stdair::DEFAULT_OND_BOOKING_RATE](#) (0.0)
- const Fare_T [stdair::DEFAULT_FARE_VALUE](#) (0.0)
- const Yield_T [stdair::DEFAULT_CLASS_YIELD_VALUE](#) (0.0)
- const Revenue_T [stdair::DEFAULT_REVENUE_VALUE](#) (0.0)
- const Percentage_T [stdair::DEFAULT_LOAD_FACTOR_VALUE](#) (100.0)
- const Yield_T [stdair::DEFAULT_YIELD_VALUE](#) (0.0)
- const Yield_T [stdair::DEFAULT_YIELD_MAX_VALUE](#) (std::numeric_limits< double >::max())
- const NbOfBookings_T [stdair::DEFAULT_YIELD_NB_OF_BOOKINGS](#) (0.0)
- const Identity_T [stdair::DEFAULT_BOOKING_NUMBER](#) (0)
- const NbOfCancellations_T [stdair::DEFAULT_YIELD_NB_OF_CANCELLATIONS](#) (0.0)
- const NbOfNoShows_T [stdair::DEFAULT_YIELD_NB_OF_NOSHOWS](#) (0.0)
- const Availability_T [stdair::DEFAULT_YIELD_AVAILABILITY](#) (0.0)

- const CensorshipFlag_T [stdair::DEFAULT_YIELD_CENSORSHIPFLAG](#) (false)
- const BookingLimit_T [stdair::DEFAULT_YIELD_BOOKING_LIMIT](#) (0.0)
- const OverbookingRate_T [stdair::DEFAULT_YIELD_OVERBOOKING_RATE](#) (0.0)
- const Fare_T [stdair::DEFAULT_OND_FARE_VALUE](#) (0.0)
- const EventQueueID_T [stdair::DEFAULT_EVENT_QUEUE_ID](#) ("EQ01")
- const Count_T [stdair::DEFAULT_PROGRESS_STATUS](#) (0)
- const Date_T [stdair::DEFAULT_EVENT_OLDEST_DATE](#) (2008, boost::gregorian::Jan, 1)
- const DateTime_T [stdair::DEFAULT_EVENT_OLDEST_DATETIME](#) (DEFAULT_EVENT_OLDEST_DATE, NULL_BOOST_TIME_DURATION)
- const PartySize_T [stdair::DEFAULT_PARTY_SIZE](#) (1)
- const DayDuration_T [stdair::DEFAULT_STAY_DURATION](#) (7)
- const WTP_T [stdair::DEFAULT_WTP](#) (1000.0)
- const Date_T [stdair::DEFAULT_PREFERRED_DEPARTURE_DATE](#) (DEFAULT_DEPARTURE_DATE)
- const Duration_T [stdair::DEFAULT_PREFERRED_DEPARTURE_TIME](#) (8, 0, 0)
- const DateOffset_T [stdair::DEFAULT_ADVANCE_PURCHASE](#) (22)
- const Date_T [stdair::DEFAULT_REQUEST_DATE](#) (DEFAULT_PREFERRED_DEPARTURE_DATE-DEFAULT_ADVANCE_PURCHASE)
- const Duration_T [stdair::DEFAULT_REQUEST_TIME](#) (8, 0, 0)
- const DateTime_T [stdair::DEFAULT_REQUEST_DATE_TIME](#) (DEFAULT_REQUEST_DATE, DEFAULT_REQUEST_TIME)
- const CabinCode_T [stdair::DEFAULT_PREFERRED_CABIN](#) ("M")
- const CityCode_T [stdair::DEFAULT_POS](#) ("ALL")
- const ChannelLabel_T [stdair::DEFAULT_CHANNEL](#) ("DC")
- const ChannelLabel_T [stdair::CHANNEL_DN](#) ("DN")
- const ChannelLabel_T [stdair::CHANNEL_IN](#) ("IN")
- const TripType_T [stdair::TRIP_TYPE_ONE_WAY](#) ("OW")
- const TripType_T [stdair::TRIP_TYPE_ROUND_TRIP](#) ("RT")
- const TripType_T [stdair::TRIP_TYPE_INBOUND](#) ("RI")
- const TripType_T [stdair::TRIP_TYPE_OUTBOUND](#) ("RO")
- const FrequentFlyer_T [stdair::DEFAULT_FF_TIER](#) ("N")
- const PriceValue_T [stdair::DEFAULT_VALUE_OF_TIME](#) (100.0)
- const Duration_T [stdair::DEFAULT_MINIMAL_CONNECTION_TIME](#) (0, 30, 0)
- const Duration_T [stdair::DEFAULT_MAXIMAL_CONNECTION_TIME](#) (24, 0, 0)
- const MatchingIndicator_T [stdair::DEFAULT_MATCHING_INDICATOR](#) (0.0)
- const PriceCurrency_T [stdair::DEFAULT_CURRENCY](#) ("EUR")
- const AvailabilityStatus_T [stdair::DEFAULT_AVAILABILITY_STATUS](#) (false)
- const AirlineCode_T [stdair::DEFAULT_AIRLINE_CODE](#) ("XX")
- const AirlineCode_T [stdair::DEFAULT_NULL_AIRLINE_CODE](#) ("")
- const FlightNumber_T [stdair::DEFAULT_FLIGHT_NUMBER](#) (9999)
- const GuillotineNumber_T [stdair::DEFAULT_GUILLOTINE_NUMBER](#) (9999)
- const Date_T [stdair::DEFAULT_DEPARTURE_DATE](#) (1900, boost::gregorian::Jan, 1)
- const AirportCode_T [stdair::DEFAULT_AIRPORT_CODE](#) ("XXX")
- const AirportCode_T [stdair::DEFAULT_NULL_AIRPORT_CODE](#) ("")
- const AirportCode_T [stdair::DEFAULT_ORIGIN](#) ("XXX")

- const AirportCode_T stdair::DEFAULT_DESTINATION ("XXX")
- const CabinCode_T stdair::DEFAULT_CABIN_CODE ("X")
- const FamilyCode_T stdair::DEFAULT_FARE_FAMILY_CODE ("EcoSaver")
- const FamilyCode_T stdair::DEFAULT_NULL_FARE_FAMILY_CODE ("NoFF")
- const ClassCode_T stdair::DEFAULT_CLASS_CODE ("X")
- const ClassCode_T stdair::DEFAULT_NULL_CLASS_CODE ("")
- const BidPrice_T stdair::DEFAULT_BID_PRICE (0.0)
- const unsigned short stdair::MAXIMAL_NUMBER_OF_LEGS_IN_FLIGHT (7)
- const unsigned short stdair::MAXIMAL_NUMBER_OF_SEGMENTS_IN_OND (3)
- const SeatIndex_T stdair::DEFAULT_SEAT_INDEX (1)
- const std::string stdair::DEFAULT_FARE_FAMILY_VALUE_TYPE ("FF")
- const std::string stdair::DEFAULT_SEGMENT_CABIN_VALUE_TYPE ("SC")
- const std::string stdair::DEFAULT_KEY_FLD_DELIMITER (";")
- const std::string stdair::DEFAULT_KEY_SUB_FLD_DELIMITER (",")
- const boost::char_separator< char > stdair::DEFAULT_KEY_TOKEN_DELIMITER (";", ",")

Variables

- const std::string stdair::DOW_STR []
- const CensorshipFlagList_T stdair::DEFAULT_CLASS_CENSORSHIPFLAG_LIST
- const Date_T stdair::DEFAULT_DICO_STUDIED_DATE
- const AirlineCodeList_T stdair::DEFAULT_AIRLINE_CODE_LIST
- const ClassList_StringList_T stdair::DEFAULT_CLASS_CODE_LIST
- const BidPriceVector_T stdair::DEFAULT_BID_PRICE_VECTOR = std::vector<BidPrice_T>()
- const int stdair::DEFAULT_MAX_DTD = 365
- const DCPList_T stdair::DEFAULT_DCP_LIST = DefaultDCPList::init()
- const DTDFractMap_T stdair::DEFAULT_DTD_FRAT5COEF_MAP
- const DTDProbMap_T stdair::DEFAULT_DTD_PROB_MAP
- const OnDStringList_T stdair::DEFAULT_OND_STRING_LIST
- const std::string stdair::DISPLAY_LEVEL_STRING_ARRAY [51]

35.22 BasConst.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // StdAir
00005 #include <stdair/basic/BasConst_General.hpp>
00006 #include <stdair/basic/BasConst_BomDisplay.hpp>
00007 #include <stdair/basic/BasConst_Event.hpp>
00008 #include <stdair/basic/BasConst_Request.hpp>
00009 #include <stdair/basic/BasConst_Inventory.hpp>
00010 #include <stdair/basic/BasConst_BookingClass.hpp>
00011 #include <stdair/basic/BasConst_Yield.hpp>
00012 #include <stdair/basic/BasConst_DefaultObject.hpp>
00013 #include <stdair/basic/BasConst_Period_BOM.hpp>
00014 #include <stdair/basic/BasConst_TravelSolution.hpp>

```

```

00015
00016 namespace stdair {
00017
00018     // ////////// General //////////
00020     const std::string DEFAULT_BOM_ROOT_KEY (" -- ROOT -- ");
00021
00023     const double DEFAULT_EPSILON_VALUE (0.0001);
00024
00026     const unsigned int DEFAULT_FLIGHT_SPEED (900);
00027
00029     const NbOfFlightDates_T DEFAULT_NB_OF_FLIGHTDATES (0.0);
00030
00032     const Duration_T NULL_BOOST_TIME_DURATION (-1, -1, -1);
00033
00035     const unsigned int DEFAULT_NB_OF_DAYS_IN_A_YEAR (365);
00036
00038     const unsigned int DEFAULT_NUMBER_OF_SUBDIVISIONS (1000);
00039
00040     // ////////// (Flight-)Period-related BOM //////////
00042     const DayDuration_T DEFAULT_DAY_DURATION (0);
00043
00045     const DatePeriod_T BOOST_DEFAULT_DATE_PERIOD (Date_T (2007, 1, 1),
00046                                                    Date_T (2007, 1, 1));
00047
00049     const std::string DOW_STR[] =
00050     { "Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun" };
00051
00053     const DOW_String_T DEFAULT_DOW_STRING ("0000000");
00054
00056     const DateOffset_T DEFAULT_DATE_OFFSET (0);
00057
00058
00059     // // ////////// General //////////
00061     const Date_T DEFAULT_DATE (2010, boost::gregorian::Jan, 1);
00062
00064     const DateTime_T DEFAULT_DATETIME (DEFAULT_DATE, NULL_BOOST_TIME_DURATION);
00065
00067     const Duration_T DEFAULT_EPSILON_DURATION (0, 0, 0, 1);
00068
00070     const Count_T SECONDS_IN_ONE_DAY (86400);
00071
00073     const Count_T MILLISECONDS_IN_ONE_SECOND (1000);
00074
00076     const RandomSeed_T DEFAULT_RANDOM_SEED (120765987);
00077
00078     // ////////// Default BOM tree objects ////////// //////////
00080     const AirportCode_T AIRPORT_LHR ("LHR");
00081
00083     const AirportCode_T AIRPORT_SYD ("SYD");
00084
00086     const CityCode_T POS_LHR ("LHR");
00087
00089     const Date_T DATE_20110115 (2011, boost::gregorian::Jan, 15);
00090     const Date_T DATE_20111231 (2011, boost::gregorian::Dec, 31);
00091
00093     const DayDuration_T NO_ADVANCE_PURCHASE (0);
00094
00096     const SaturdayStay_T SATURDAY_STAY (true);
00097
00099     const SaturdayStay_T NO_SATURDAY_STAY (false);
00100
00102     const ChangeFees_T CHANGE_FEES (true);

```

```

00103
00105     const ChangeFees_T NO_CHANGE_FEES (false);
00106
00108     const NonRefundable_T NON_REFUNDABLE (true);
00109
00111     const NonRefundable_T No_NON_REFUNDABLE (false);
00112
00114     const SaturdayStay_T DEFAULT_BOM_TREE_SATURDAY_STAY (true);
00115
00117     const ChangeFees_T DEFAULT_BOM_TREE_CHANGE_FEES (true);
00118
00120     const NonRefundable_T DEFAULT_BOM_TREE_NON_REFUNDABLE (true);
00121
00123     const DayDuration_T NO_STAY_DURATION (0);
00124
00126     const AirlineCode_T AIRLINE_CODE_BA ("BA");
00127
00129     const CabinCode_T CABIN_Y ("Y");
00130
00132     const ClassCode_T CLASS_CODE_Y ("Y");
00133
00134     // ////////// Travel solutions related objects////////
00136     const ClassCode_T CLASS_CODE_Q ("Q");
00137
00138     // ////////// Booking request related objects////////
00140     const AirportCode_T AIRPORT_SIN ("SIN");
00141
00143     const AirportCode_T AIRPORT_BKK ("BKK");
00144
00146     const CityCode_T POS_SIN ("SIN");
00147
00149     const CabinCode_T CABIN_ECO ("Eco");
00150
00152     const FrequentFlyer_T FREQUENT_FLYER_MEMBER ("M");
00153
00154     // ////////// Default //////////
00156     const ClassCode_T DEFAULT_FAMILY_CODE ("0");
00157
00159     const NbofAirlines_T DEFAULT_NBOFAIRLINES (0);
00160
00162     const FlightPathCode_T DEFAULT_FLIGHTPATH_CODE ("");
00163
00164     // ////////// Booking-class-related BOM //////////
00166     const Distance_T DEFAULT_DISTANCE_VALUE (0);
00167
00169     const ClassCode_T DEFAULT_CLOSED_CLASS_CODE ("CC");
00170
00173     const NbofBookings_T DEFAULT_CLASS_NB_OF_BOOKINGS (0);
00174
00177     const NbofBookings_T DEFAULT_CLASS_TOTAL_NB_OF_BOOKINGS (0);
00178
00180     const NbofBookings_T DEFAULT_CLASS_UNCONSTRAINED_DEMAND (0);
00181
00183     const NbofBookings_T DEFAULT_CLASS_REMAINING_DEMAND_MEAN (0);
00184
00186     const NbofBookings_T DEFAULT_CLASS_REMAINING_DEMAND_STANDARD_DEVIATION (0);
00187
00189     const NbofCancellations_T DEFAULT_CLASS_NB_OF_CANCELLATIONS (0);
00190
00192     const NbofNoShows_T DEFAULT_CLASS_NB_OF_NOSHOWS (0);
00193
00195     const CabinCapacity_T DEFAULT_CABIN_CAPACITY (100.0);

```

```

00196
00198     const CommittedSpace_T DEFAULT_COMMITTED_SPACE (0.0);
00199
00201     const BlockSpace_T DEFAULT_BLOCK_SPACE (0.0);
00202
00204     const Availability_T DEFAULT_NULL_AVAILABILITY (0.0);
00205
00207     const Availability_T DEFAULT_AVAILABILITY (9.0);
00208
00210     const Availability_T MAXIMAL_AVAILABILITY (9999.0);
00211
00212
00213     // ////////// (Segment-)Class-related BOM //////////
00216     const CensorshipFlag_T DEFAULT_CLASS_CENSORSHIPFLAG (false);
00217
00220     const CensorshipFlagList_T DEFAULT_CLASS_CENSORSHIPFLAG_LIST =
00221         std::vector<CensorshipFlag_T>();
00222
00224     const BookingLimit_T DEFAULT_CLASS_BOOKING_LIMIT (9999.0);
00225
00227     const AuthorizationLevel_T DEFAULT_CLASS_AUTHORIZATION_LEVEL (9999.0);
00228
00230     const AuthorizationLevel_T DEFAULT_CLASS_MAX_AUTHORIZATION_LEVEL (9999.0);
00231
00233     const AuthorizationLevel_T DEFAULT_CLASS_MIN_AUTHORIZATION_LEVEL (0.0);
00234
00236     const OverbookingRate_T DEFAULT_CLASS_OVERBOOKING_RATE (0.0);
00237
00239     const BookingRatio_T DEFAULT_OND_BOOKING_RATE (0.0);
00240
00242     const Fare_T DEFAULT_FARE_VALUE (0.0);
00243
00245     const Yield_T DEFAULT_CLASS_YIELD_VALUE (0.0);
00246
00248     const Revenue_T DEFAULT_REVENUE_VALUE (0.0);
00249
00251     const Percentage_T DEFAULT_LOAD_FACTOR_VALUE (100.0);
00252
00253
00254     // ////////// (Leg-)YieldRange-related BOM //////////
00256     const Yield_T DEFAULT_YIELD_VALUE (0.0);
00257
00259     const Yield_T DEFAULT_YIELD_MAX_VALUE (std::numeric_limits<double>::max());
00260
00262     const NbOfBookings_T DEFAULT_YIELD_NB_OF_BOOKINGS (0.0);
00263
00265     const Identity_T DEFAULT_BOOKING_NUMBER (0);
00266
00268     const NbOfCancellations_T DEFAULT_YIELD_NB_OF_CANCELLATIONS (0.0);
00269
00271     const NbOfNoShows_T DEFAULT_YIELD_NB_OF_NOSHOWS (0.0);
00272
00274     const Availability_T DEFAULT_YIELD_AVAILABILITY (0.0);
00275
00278     const CensorshipFlag_T DEFAULT_YIELD_CENSORSHIPFLAG (false);
00279
00281     const BookingLimit_T DEFAULT_YIELD_BOOKING_LIMIT (0.0);
00282
00284     const OverbookingRate_T DEFAULT_YIELD_OVERBOOKING_RATE (0.0);
00285
00286
00287     // ////////// OnD-related BOM //////////

```

```
00289     const Fare_T DEFAULT_OND_FARE_VALUE (0.0);
00290
00291
00292     // ////////// Event Generation //////////
00294     const EventQueueID_T DEFAULT_EVENT_QUEUE_ID ("EQ01");
00295
00297     const Count_T DEFAULT_PROGRESS_STATUS (0);
00298
00301     const Date_T DEFAULT_EVENT_OLDEST_DATE (2008, boost::gregorian::Jan, 1);
00302
00305     const DateTime_T DEFAULT_EVENT_OLDEST_DATETIME (DEFAULT_EVENT_OLDEST_DATE,
00306                                                     NULL_BOOST_TIME_DURATION);
00307
00308
00309     // ////////// Booking Request //////////
00311     const PartySize_T DEFAULT_PARTY_SIZE (1);
00312
00314     const DayDuration_T DEFAULT_STAY_DURATION (7);
00315
00317     const WTP_T DEFAULT_WTP (1000.0);
00318
00320     const Date_T DEFAULT_PREFERRED_DEPARTURE_DATE (DEFAULT_DEPARTURE_DATE);
00321
00323     const Duration_T DEFAULT_PREFERRED_DEPARTURE_TIME (8, 0, 0);
00324
00326     const DateOffset_T DEFAULT_ADVANCE_PURCHASE (22);
00327
00329     const Date_T DEFAULT_REQUEST_DATE (DEFAULT_PREFERRED_DEPARTURE_DATE
00330                                       - DEFAULT_ADVANCE_PURCHASE);
00331
00333     const Duration_T DEFAULT_REQUEST_TIME (8, 0, 0);
00334
00336     const DateTime_T DEFAULT_REQUEST_DATE_TIME (DEFAULT_REQUEST_DATE,
00337                                                  DEFAULT_REQUEST_TIME);
00338
00340     const CabinCode_T DEFAULT_PREFERRED_CABIN ("M");
00341
00343     const CityCode_T DEFAULT_POS ("ALL");
00344
00346     const ChannelLabel_T DEFAULT_CHANNEL ("DC");
00347
00349     const ChannelLabel_T CHANNEL_DN ("DN");
00350
00352     const ChannelLabel_T CHANNEL_IN ("IN");
00353
00355     const TripType_T TRIP_TYPE_ONE_WAY ("OW");
00356
00358     const TripType_T TRIP_TYPE_ROUND_TRIP ("RT");
00359
00361     const TripType_T TRIP_TYPE_INBOUND ("RI");
00362
00364     const TripType_T TRIP_TYPE_OUTBOUND ("RO");
00365
00367     const FrequentFlyer_T DEFAULT_FF_TIER ("N");
00368
00370     const PriceValue_T DEFAULT_VALUE_OF_TIME (100.0);
00371
00372
00373     // ////////// Travel Solutions //////////
00375     const Duration_T DEFAULT_MINIMAL_CONNECTION_TIME (0, 30, 0);
00376
00378     const Duration_T DEFAULT_MAXIMAL_CONNECTION_TIME (24, 0, 0);
```

```
00379
00381     const MatchingIndicator_T DEFAULT_MATCHING_INDICATOR (0.0);
00382
00384     const PriceCurrency_T DEFAULT_CURRENCY ("EUR");
00385
00387     const AvailabilityStatus_T DEFAULT_AVAILABILITY_STATUS (false);
00388
00390     const Date_T DEFAULT_DICO_STUDIED_DATE;
00391
00392     // ////////// Inventory-related BOM //////////
00394     const AirlineCode_T DEFAULT_AIRLINE_CODE ("XX");
00395
00397     const AirlineCode_T DEFAULT_NULL_AIRLINE_CODE ("");
00398
00400     const AirlineCodeList_T DEFAULT_AIRLINE_CODE_LIST;
00401
00403     const FlightNumber_T DEFAULT_FLIGHT_NUMBER (9999);
00404
00406     const GuillotineNumber_T DEFAULT_GUILLOTINE_NUMBER (9999);
00407
00409     const Date_T DEFAULT_DEPARTURE_DATE (1900, boost::gregorian::Jan, 1);
00410
00412     const AirportCode_T DEFAULT_AIRPORT_CODE ("XXX");
00413
00415     const AirportCode_T DEFAULT_NULL_AIRPORT_CODE ("");
00416
00418     const AirportCode_T DEFAULT_ORIGIN ("XXX");
00419
00421     const AirportCode_T DEFAULT_DESTINATION ("XXX");
00422
00424     const CabinCode_T DEFAULT_CABIN_CODE ("X");
00425
00427     const FamilyCode_T DEFAULT_FARE_FAMILY_CODE ("EcoSaver");
00428
00430     const FamilyCode_T DEFAULT_NULL_FARE_FAMILY_CODE ("NoFF");
00431
00433     const ClassCode_T DEFAULT_CLASS_CODE ("X");
00434
00436     const ClassCode_T DEFAULT_NULL_CLASS_CODE ("");
00437
00439     const ClassList_StringList_T DEFAULT_CLASS_CODE_LIST;
00440
00442     const BidPrice_T DEFAULT_BID_PRICE (0.0);
00443
00445     const BidPriceVector_T DEFAULT_BID_PRICE_VECTOR = std::vector<BidPrice_T>();
00446
00450     const unsigned short MAXIMAL_NUMBER_OF_LEGS_IN_FLIGHT (7);
00451
00454     const unsigned short MAXIMAL_NUMBER_OF_SEGMENTS_IN_OND (3);
00455
00457     const SeatIndex_T DEFAULT_SEAT_INDEX (1);
00458
00460     const std::string DEFAULT_FARE_FAMILY_VALUE_TYPE ("FF");
00461
00463     const std::string DEFAULT_SEGMENT_CABIN_VALUE_TYPE ("SC");
00464
00466     const int DEFAULT_MAX_DTD = 365;
00467
00469     const DCPList_T DEFAULT_DCP_LIST = DefaultDCPList::init();
00470     DCPList_T DefaultDCPList::init() {
00471         DCPList_T oDCPList;
00472         //oDCPList.push_back (72);
```

Generated on Wed Nov 2 2011 22:19:08 for StdAir by Doxygen

```

00545     "
00546     "
00547     "
00548     "
00549     "
00550     "
00551     "
00552     "
00553     "
00554     "
00555     "
00556     "
00557     "
00558     "
00559     "
00560     "
00561     "

00562     "
00563     "
00564     "
00565     "
00566     "
00567     "
00568     "
00569     "
00570     "
00571     "
00572     "
00573     "
00574     "
00575     "

    " };

00576
00577
00578 }
```

35.23 stdair/basic/BasConst_BomDisplay.hpp File Reference

```

#include <string>
#include <boost/tokenizer.hpp>
```

Namespaces

- namespace [stdair](#)

Handle on the StdAir library context.

Variables

- const std::string [stdair::DEFAULT_KEY_FLD_DELIMITER](#)
- const std::string [stdair::DEFAULT_KEY_SUB_FLD_DELIMITER](#)
- const boost::char_separator< char > [stdair::DEFAULT_KEY_TOKEN_DELIMITER](#)

35.24 BasConst_BomDisplay.hpp

```

00001 #ifndef __STDAIR_BAS_BASCONST_BOMMANAGER_HPP
00002 #define __STDAIR_BAS_BASCONST_BOMMANAGER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // Boost
00010 #include <boost/tokenizer.hpp>
00011
00012 namespace stdair {
00013
00016     extern const std::string DISPLAY_LEVEL_STRING_ARRAY[51];
00017
00020     extern const std::string DEFAULT_KEY_FLD_DELIMITER;
00021
00024     extern const std::string DEFAULT_KEY_SUB_FLD_DELIMITER;
00025
00027     extern const boost::char_separator<char> DEFAULT_KEY_TOKEN_DELIMITER;
00028
00029 }
00030 #endif // __STDAIR_BAS_BASCONST_BOMMANAGER_HPP

```

35.25 stdair/basic/BasConst_BookingClass.hpp File Reference

```

#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/stdair_fare_types.hpp>

```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

Variables

- const Distance_T [stdair::DEFAULT_DISTANCE_VALUE](#)

- const ClassCode_T stdair::DEFAULT_CLOSED_CLASS_CODE
- const NbOfBookings_T stdair::DEFAULT_CLASS_NB_OF_BOOKINGS
- const NbOfBookings_T stdair::DEFAULT_CLASS_TOTAL_NB_OF_BOOKINGS
- const NbOfBookings_T stdair::DEFAULT_CLASS_UNCONSTRAINED_DEMAND
- const NbOfBookings_T stdair::DEFAULT_CLASS_REMAINING_DEMAND_MEAN
- const NbOfBookings_T stdair::DEFAULT_CLASS_REMAINING_DEMAND_STANDARD_DEVIATION
- const NbOfCancellations_T stdair::DEFAULT_CLASS_NB_OF_CANCELLATIONS
- const NbOfNoShows_T stdair::DEFAULT_CLASS_NB_OF_NOSHOWS
- const CabinCapacity_T stdair::DEFAULT_CABIN_CAPACITY
- const CommittedSpace_T stdair::DEFAULT_COMMITTED_SPACE
- const BlockSpace_T stdair::DEFAULT_BLOCK_SPACE
- const Availability_T stdair::DEFAULT_NULL_AVAILABILITY
- const Availability_T stdair::DEFAULT_AVAILABILITY
- const CensorshipFlag_T stdair::DEFAULT_CLASS_CENSORSHIPFLAG
- const BookingLimit_T stdair::DEFAULT_CLASS_BOOKING_LIMIT
- const AuthorizationLevel_T stdair::DEFAULT_CLASS_AUTHORIZATION_LEVEL
- const AuthorizationLevel_T stdair::DEFAULT_CLASS_MAX_AUTHORIZATION_LEVEL
- const AuthorizationLevel_T stdair::DEFAULT_CLASS_MIN_AUTHORIZATION_LEVEL
- const OverbookingRate_T stdair::DEFAULT_CLASS_OVERBOOKING_RATE
- const Fare_T stdair::DEFAULT_FARE_VALUE
- const Revenue_T stdair::DEFAULT_REVENUE_VALUE
- const PriceCurrency_T stdair::DEFAULT_CURRENCY
- const Percentage_T stdair::DEFAULT_LOAD_FACTOR_VALUE
- const DayDuration_T stdair::DEFAULT_DAY_DURATION
- const double stdair::DEFAULT_EPSILON_VALUE

35.26 BasConst_BookingClass.hpp

```

00001 #ifndef __STDAIR_BAS_BASCONST_BOOKINGCLASS_HPP
00002 #define __STDAIR_BAS_BASCONST_BOOKINGCLASS_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009 #include <stdair/stdair_inventory_types.hpp>
00010 #include <stdair/stdair_demand_types.hpp>
00011 #include <stdair/stdair_fare_types.hpp>
00012
00013 namespace stdair {
00014
00015     // ////////// (Segment-)Class-related BOM //////////
00017     extern const Distance_T DEFAULT_DISTANCE_VALUE;
00018
00020     extern const ClassCode_T DEFAULT_CLOSED_CLASS_CODE;
00021
00024     extern const NbOfBookings_T DEFAULT_CLASS_NB_OF_BOOKINGS;

```

```

00025
00028     extern const NbOfBookings_T DEFAULT_CLASS_TOTAL_NB_OF_BOOKINGS;
00029
00031     extern const NbOfBookings_T DEFAULT_CLASS_UNCONSTRAINED_DEMAND;
00032
00034     extern const NbOfBookings_T DEFAULT_CLASS_REMAINING_DEMAND_MEAN;
00035
00038     extern const NbOfBookings_T DEFAULT_CLASS_REMAINING_DEMAND_STANDARD_DEVIATION;
00039
00041     extern const NbOfCancellations_T DEFAULT_CLASS_NB_OF_CANCELLATIONS;
00042
00044     extern const NbOfNoShows_T DEFAULT_CLASS_NB_OF_NOSHOWS;
00045
00047     extern const CabinCapacity_T DEFAULT_CABIN_CAPACITY;
00048
00050     extern const CommittedSpace_T DEFAULT_COMMITTED_SPACE;
00051
00053     extern const BlockSpace_T DEFAULT_BLOCK_SPACE;
00054
00056     extern const Availability_T DEFAULT_NULL_AVAILABILITY;
00057
00059     extern const Availability_T DEFAULT_AVAILABILITY;
00060
00063     extern const CensorshipFlag_T DEFAULT_CLASS_CENSORSHIPFLAG;
00064
00067     extern const CensorshipFlagList_T DEFAULT_CLASS_CENSORSHIPFLAG_LIST;
00068
00070     extern const BookingLimit_T DEFAULT_CLASS_BOOKING_LIMIT;
00071
00073     extern const AuthorizationLevel_T DEFAULT_CLASS_AUTHORIZATION_LEVEL;
00074
00076     extern const AuthorizationLevel_T DEFAULT_CLASS_MAX_AUTHORIZATION_LEVEL;
00077
00079     extern const AuthorizationLevel_T DEFAULT_CLASS_MIN_AUTHORIZATION_LEVEL;
00080
00082     extern const OverbookingRate_T DEFAULT_CLASS_OVERBOOKING_RATE;
00083
00085     extern const Fare_T DEFAULT_FARE_VALUE;
00086
00088     extern const Revenue_T DEFAULT_REVENUE_VALUE;
00089
00091     extern const PriceCurrency_T DEFAULT_CURRENCY;
00092
00094     extern const Percentage_T DEFAULT_LOAD_FACTOR_VALUE;
00095
00097     extern const DayDuration_T DEFAULT_DAY_DURATION;
00098
00101     extern const double DEFAULT_EPSILON_VALUE;
00102
00103 }
00104 #endif // __STDAIR_BAS_BASCONST_BOOKINGCLASS_HPP

```

35.27 stdair/basic/BasConst_DefaultObject.hpp File Reference

```
#include <stdair/stdair_types.hpp>
```

Namespaces

- namespace `stdair`

Handle on the StdAir library context.

Variables

- const AirportCode_T [stdair::AIRPORT_LHR](#)
- const AirportCode_T [stdair::AIRPORT_SYD](#)
- const CityCode_T [stdair::POS_LHR](#)
- const DayDuration_T [stdair::NO_ADVANCE_PURCHASE](#)
- const SaturdayStay_T [stdair::SATURDAY_STAY](#)
- const SaturdayStay_T [stdair::NO_SATURDAY_STAY](#)
- const ChangeFees_T [stdair::CHANGE_FEES](#)
- const ChangeFees_T [stdair::NO_CHANGE_FEES](#)
- const NonRefundable_T [stdair::NON_REFUNDABLE](#)
- const NonRefundable_T [stdair::NO_NON_REFUNDABLE](#)
- const DayDuration_T [stdair::NO_STAY_DURATION](#)
- const CabinCode_T [stdair::CABIN_Y](#)
- const AirlineCode_T [stdair::AIRLINE_CODE_BA](#)
- const ClassCode_T [stdair::CLASS_CODE_Y](#)
- const ClassCode_T [stdair::CLASS_CODE_Q](#)
- const AirportCode_T [stdair::AIRPORT_SIN](#)
- const AirportCode_T [stdair::AIRPORT_BKK](#)
- const CityCode_T [stdair::POS_SIN](#)
- const CabinCode_T [stdair::CABIN_ECO](#)
- const FrequentFlyer_T [stdair::FREQUENT_FLYER_MEMBER](#)

35.28 BasConst_DefaultObject.hpp

```

00001 #ifndef __STDAIR_BAS_BASCONST_DEFAULTOBJECT_HPP
00002 #define __STDAIR_BAS_BASCONST_DEFAULTOBJECT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_types.hpp>
00009
00010 namespace stdair {
00011
00012 // /////////// Fare and Yield related BOM Tree/////////
00014 extern const AirportCode_T AIRPORT_LHR;
00015
00017 extern const AirportCode_T AIRPORT_SYD;
00018
00020 extern const CityCode_T POS_LHR;
00021
00023 extern const DayDuration_T NO_ADVANCE_PURCHASE;
00024
00026 extern const SaturdayStay_T SATURDAY_STAY;
00027
00029 extern const SaturdayStay_T NO_SATURDAY_STAY;
00030
00032 extern const ChangeFees_T CHANGE_FEES;

```

```

00033
00035  extern const ChangeFees_T NO_CHANGE_FEES;
00036
00038  extern const NonRefundable_T NON_REFUNDABLE;
00039
00041  extern const NonRefundable_T NO_NON_REFUNDABLE;
00042
00044  extern const DayDuration_T NO_STAY_DURATION;
00045
00047  extern const CabinCode_T CABIN_Y;
00048
00050  extern const AirlineCode_T AIRLINE_CODE_BA;
00051
00053  extern const ClassCode_T CLASS_CODE_Y;
00054
00055  // ////////// Travel Solution related objects////////
00057  extern const ClassCode_T CLASS_CODE_Q;
00058
00059  // ////////// Booking request related objects////////
00061  extern const AirportCode_T AIRPORT_SIN;
00062
00064  extern const AirportCode_T AIRPORT_BKK;
00065
00067  extern const CityCode_T POS_SIN;
00068
00070  extern const CabinCode_T CABIN_ECO;
00071
00073  extern const FrequentFlyer_T FREQUENT_FLYER_MEMBER;
00074
00075 }
00076 #endif // __STDAIR_BAS_BASCONST_DEFAULTOBJECT_HPP

```

35.29 stdair/basic/BasConst_Event.hpp File Reference

```

#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/stdair_event_types.hpp>

```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

Variables

- const EventQueueID_T [stdair::DEFAULT_EVENT_QUEUE_ID](#)
- const Count_T [stdair::DEFAULT_PROGRESS_STATUS](#)
- const Date_T [stdair::DEFAULT_EVENT_OLDEST_DATE](#)
- const DateTime_T [stdair::DEFAULT_EVENT_OLDEST_DATETIME](#)

35.30 BasConst_Event.hpp

```

00001 #ifndef __STDAIR_BAS_BASCONST_EVENT_HPP
00002 #define __STDAIR_BAS_BASCONST_EVENT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009 #include <stdair/stdair_date_time_types.hpp>
00010 #include <stdair/stdair_event_types.hpp>
00011
00012 namespace stdair {
00013
00015     extern const EventQueueID_T DEFAULT_EVENT_QUEUE_ID;
00016
00018     extern const Count_T DEFAULT_PROGRESS_STATUS;
00019
00022     extern const Date_T DEFAULT_EVENT_OLDEST_DATE;
00023
00026     extern const DateTime_T DEFAULT_EVENT_OLDEST_DATETIME;
00027
00028 }
00029 #endif // __STDAIR_BAS_BASCONST_EVENT_HPP

```

35.31 stdair/basic/BasConst_General.hpp File Reference

```

#include <string>
#include <stdair/stdair_types.hpp>

```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

Variables

- const std::string `stdair::DEFAULT_BOM_ROOT_KEY`
- const NbOfFlightDates_T `stdair::DEFAULT_NB_OF_FLIGHTDATES`
- const unsigned int `stdair::DEFAULT_FLIGHT_SPEED`
- const BookingRatio_T `stdair::DEFAULT_OND_BOOKING_RATE`
- const Count_T `stdair::SECONDS_IN_ONE_DAY`
- const Count_T `stdair::MILLISECONDS_IN_ONE_SECOND`
- const Date_T `stdair::DEFAULT_DATE`
- const DateTime_T `stdair::DEFAULT_DATETIME`
- const Duration_T `stdair::DEFAULT_EPSILON_DURATION`
- const RandomSeed_T `stdair::DEFAULT_RANDOM_SEED`
- const Duration_T `stdair::NULL_BOOST_TIME_DURATION`
- const Fare_T `stdair::DEFAULT_CLASS_FARE_VALUE`
- const NbOfAirlines_T `stdair::DEFAULT_NBOFAIRLINES`

- const unsigned int `stdair::DEFAULT_NB_OF_DAYS_IN_A_YEAR`
- const `Channellabel_T` `stdair::DEFAULT_CHANNEL`
- const unsigned int `stdair::DEFAULT_NUMBER_OF_SUBDIVISIONS`

35.32 BasConst_General.hpp

```

00001 #ifndef __STDAIR_BAS_BASCONST_GENERAL_HPP
00002 #define __STDAIR_BAS_BASCONST_GENERAL_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/stdair_types.hpp>
00011
00012 namespace stdair {
00013
00015     extern const std::string DEFAULT_BOM_ROOT_KEY;
00016
00018     extern const double DEFAULT_EPSILON_VALUE;
00019
00021     extern const CabinCapacity_T DEFAULT_CABIN_CAPACITY;
00022
00024     extern const NbOfFlightDates_T DEFAULT_NB_OF_FLIGHTDATES;
00025
00027     extern const NbOfBookings_T DEFAULT_CLASS_NB_OF_BOOKINGS;
00028
00030     extern const Distance_T DEFAULT_DISTANCE_VALUE;
00031
00033     extern const unsigned int DEFAULT_FLIGHT_SPEED;
00034
00036     extern const Fare_T DEFAULT_FARE_VALUE;
00037
00039     extern const PriceCurrency_T DEFAULT_CURRENCY;
00040
00042     extern const Revenue_T DEFAULT_REVENUE_VALUE;
00043
00045     extern const BookingRatio_T DEFAULT_OND_BOOKING_RATE;
00046
00048     extern const Count_T SECONDS_IN_ONE_DAY;
00049
00051     extern const Count_T MILLISECONDS_IN_ONE_SECOND;
00052
00054     extern const Date_T DEFAULT_DATE;
00055
00057     extern const DateTime_T DEFAULT_DATETIME;
00058
00060     extern const Duration_T DEFAULT_EPSILON_DURATION;
00061
00063     extern const RandomSeed_T DEFAULT_RANDOM_SEED;
00064
00066     extern const Duration_T NULL_BOOST_TIME_DURATION;
00067
00069     extern const Fare_T DEFAULT_CLASS_FARE_VALUE;
00070
00072     extern const NbOfAirlines_T DEFAULT_NBOFAIRLINES;
00073
00075     extern const unsigned int DEFAULT_NB_OF_DAYS_IN_A_YEAR;

```

```

00076
00078     extern const NbOfBookings_T DEFAULT_CLASS_NB_OF_BOOKINGS;
00079
00081     extern const ChannelLabel_T DEFAULT_CHANNEL;
00082
00084     extern const OnDStringList_T DEFAULT_OND_STRING_LIST;
00085
00087     extern const unsigned int DEFAULT_NUMBER_OF_SUBDIVISIONS;
00088
00089 }
00090 #endif // __STDAIR_BAS_BASCONST_GENERAL_HPP

```

35.33 stdair/basic/BasConst_Inventory.hpp File Reference

```
#include <stdair/stdair_inventory_types.hpp>
```

```
#include <stdair/stdair_date_time_types.hpp>
```

Classes

- struct [stdair::DefaultDCPList](#)
- struct [stdair::DefaultDtdFratMap](#)
- struct [stdair::DefaultDtdProbMap](#)

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

Variables

- const AirlineCode_T [stdair::DEFAULT_AIRLINE_CODE](#)
- const AirlineCode_T [stdair::DEFAULT_NULL_AIRLINE_CODE](#)
- const FlightNumber_T [stdair::DEFAULT_FLIGHT_NUMBER](#)
- const GuillotineNumber_T [stdair::DEFAULT_GUILLOTINE_NUMBER](#)
- const Date_T [stdair::DEFAULT_DEPARTURE_DATE](#)
- const AirportCode_T [stdair::DEFAULT_AIRPORT_CODE](#)
- const AirportCode_T [stdair::DEFAULT_NULL_AIRPORT_CODE](#)
- const AirportCode_T [stdair::DEFAULT_ORIGIN](#)
- const AirportCode_T [stdair::DEFAULT_DESTINATION](#)
- const CabinCode_T [stdair::DEFAULT_CABIN_CODE](#)
- const FamilyCode_T [stdair::DEFAULT_FARE_FAMILY_CODE](#)
- const FamilyCode_T [stdair::DEFAULT_NULL_FARE_FAMILY_CODE](#)
- const ClassCode_T [stdair::DEFAULT_CLASS_CODE](#)
- const ClassCode_T [stdair::DEFAULT_NULL_CLASS_CODE](#)
- const BidPrice_T [stdair::DEFAULT_BID_PRICE](#)
- const unsigned short [stdair::MAXIMAL_NUMBER_OF_LEGS_IN_FLIGHT](#)
- const unsigned short [stdair::MAXIMAL_NUMBER_OF_SEGMENTS_IN_OND](#)

- const Availability_T stdair::MAXIMAL_AVAILABILITY
- const SeatIndex_T stdair::DEFAULT_SEAT_INDEX
- const std::string stdair::DEFAULT_FARE_FAMILY_VALUE_TYPE
- const std::string stdair::DEFAULT_SEGMENT_CABIN_VALUE_TYPE

35.34 BasConst_Inventory.hpp

```

00001 #ifndef __STDAIR_BAS_BASCONST_INVENTORY_HPP
00002 #define __STDAIR_BAS_BASCONST_INVENTORY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_inventory_types.hpp>
00009 #include <stdair/stdair_date_time_types.hpp>
00010
00011 namespace stdair {
00012
00013     // ////////// Inventory-related BOM //////////
00015     extern const AirlineCode_T DEFAULT_AIRLINE_CODE;
00016
00018     extern const AirlineCode_T DEFAULT_NULL_AIRLINE_CODE;
00019
00021     extern const AirlineCodeList_T DEFAULT_AIRLINE_CODE_LIST;
00022
00024     extern const FlightNumber_T DEFAULT_FLIGHT_NUMBER;
00025
00027     extern const GuillotineNumber_T DEFAULT_GUILLOTINE_NUMBER;
00028
00030     extern const Date_T DEFAULT_DEPARTURE_DATE;
00031
00033     extern const AirportCode_T DEFAULT_AIRPORT_CODE;
00034
00036     extern const AirportCode_T DEFAULT_NULL_AIRPORT_CODE;
00037
00039     extern const AirportCode_T DEFAULT_ORIGIN;
00040
00042     extern const AirportCode_T DEFAULT_DESTINATION;
00043
00045     extern const CabinCode_T DEFAULT_CABIN_CODE;
00046
00048     extern const FamilyCode_T DEFAULT_FARE_FAMILY_CODE;
00049
00051     extern const FamilyCode_T DEFAULT_NULL_FARE_FAMILY_CODE;
00052
00054     extern const ClassCode_T DEFAULT_CLASS_CODE;
00055
00057     extern const ClassCode_T DEFAULT_NULL_CLASS_CODE;
00058
00060     extern const ClassList_StringList_T DEFAULT_CLASS_CODE_LIST;
00061
00063     extern const BidPrice_T DEFAULT_BID_PRICE;
00064
00066     extern const BidPriceVector_T DEFAULT_BID_PRICE_VECTOR;
00067
00071     extern const unsigned short MAXIMAL_NUMBER_OF_LEGS_IN_FLIGHT;
00072
00075     extern const unsigned short MAXIMAL_NUMBER_OF_SEGMENTS_IN_OND;
00076

```

```

00078  extern const Availability_T MAXIMAL_AVAILABILITY;
00079
00081  extern const SeatIndex_T DEFAULT_SEAT_INDEX;
00082
00084  extern const std::string DEFAULT_FARE_FAMILY_VALUE_TYPE;
00085
00087  extern const std::string DEFAULT_SEGMENT_CABIN_VALUE_TYPE;
00088
00090  extern const int DEFAULT_MAX_DTD;
00091
00093  extern const DCPList_T DEFAULT_DCP_LIST;
00094  struct DefaultDCPList { static DCPList_T init(); };
00095
00097  extern const DTDFratMap_T DEFAULT_DTD_FRAT5COEF_MAP;
00098  struct DefaultDtdFratMap { static DTDFratMap_T init(); };
00099
00101  extern const DTDProbMap_T DEFAULT_DTD_PROB_MAP;
00102  struct DefaultDtdProbMap { static DTDProbMap_T init(); };
00103
00104
00105 }
00106 #endif // __STDAIR_BAS_BASCONST_INVENTORY_HPP

```

35.35 stdair/basic/BasConst_Period_BOM.hpp File Reference

```
#include <stdair/stdair_types.hpp>
```

Namespaces

- namespace `stdair`
Handle on the `StdAir` library context.

Variables

- const `DatePeriod_T` `stdair::BOOST_DEFAULT_DATE_PERIOD`
- const `DOW_String_T` `stdair::DEFAULT_DOW_STRING`
- const `DateOffset_T` `stdair::DEFAULT_DATE_OFFSET`

35.36 BasConst_Period_BOM.hpp

```

00001 #ifndef __STDAIR_BAS_BASCONST_PERIOD_BOM_HPP
00002 #define __STDAIR_BAS_BASCONST_PERIOD_BOM_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_types.hpp>
00009
00010 namespace stdair {
00011
00012 // ////////// (Flight-)Period-related BOM //////////
00014  extern const DatePeriod_T BOOST_DEFAULT_DATE_PERIOD;

```

```

00015
00017     extern const std::string DOW_STR[];
00018
00020     extern const DOW_String_T DEFAULT_DOW_STRING;
00021
00023     extern const DateOffset_T DEFAULT_DATE_OFFSET;
00024
00026     extern const DayDuration_T DEFAULT_DAY_DURATION;
00027
00028 }
00029 #endif // __STDAIR_BAS_BASCONST_PERIOD_BOM_HPP

```

35.37 stdair/basic/BasConst_Request.hpp File Reference

```

#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/stdair_date_time_types.hpp>

```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

Variables

- const PartySize_T `stdair::DEFAULT_PARTY_SIZE`
- const DayDuration_T `stdair::DEFAULT_STAY_DURATION`
- const WTP_T `stdair::DEFAULT_WTP`
- const CityCode_T `stdair::DEFAULT_POS`
- const Date_T `stdair::DEFAULT_PREFERRED_DEPARTURE_DATE`
- const Duration_T `stdair::DEFAULT_PREFERRED_DEPARTURE_TIME`
- const DateOffset_T `stdair::DEFAULT_ADVANCE_PURCHASE`
- const Date_T `stdair::DEFAULT_REQUEST_DATE`
- const Duration_T `stdair::DEFAULT_REQUEST_TIME`
- const DateTime_T `stdair::DEFAULT_REQUEST_DATE_TIME`
- const CabinCode_T `stdair::DEFAULT_PREFERRED_CABIN`
- const ChannelLabel_T `stdair::CHANNEL_DN`
- const ChannelLabel_T `stdair::CHANNEL_IN`
- const TripType_T `stdair::TRIP_TYPE_ONE_WAY`
- const TripType_T `stdair::TRIP_TYPE_ROUND_TRIP`
- const TripType_T `stdair::TRIP_TYPE_INBOUND`
- const TripType_T `stdair::TRIP_TYPE_OUTBOUND`
- const FrequentFlyer_T `stdair::DEFAULT_FF_TIER`
- const PriceValue_T `stdair::DEFAULT_VALUE_OF_TIME`

35.38 BasConst_Request.hpp

```

00001 #ifndef __STDAIR_BAS_BASCONST_REQUEST_HPP
00002 #define __STDAIR_BAS_BASCONST_REQUEST_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009 #include <stdair/stdair_demand_types.hpp>
00010 #include <stdair/stdair_date_time_types.hpp>
00011
00012 namespace stdair {
00013
00015     extern const PartySize_T DEFAULT_PARTY_SIZE;
00016
00018     extern const DayDuration_T DEFAULT_STAY_DURATION;
00019
00021     extern const WTP_T DEFAULT_WTP;
00022
00024     extern const CityCode_T DEFAULT_POS;
00025
00027     extern const Date_T DEFAULT_PREFERRED_DEPARTURE_DATE;
00028
00030     extern const Duration_T DEFAULT_PREFERRED_DEPARTURE_TIME;
00031
00033     extern const DateOffset_T DEFAULT_ADVANCE_PURCHASE;
00034
00036     extern const Date_T DEFAULT_REQUEST_DATE;
00037
00039     extern const Duration_T DEFAULT_REQUEST_TIME;
00040
00042     extern const DateTime_T DEFAULT_REQUEST_DATE_TIME;
00043
00045     extern const CabinCode_T DEFAULT_PREFERRED_CABIN;
00046
00048     extern const ChannelLabel_T DEFAULT_CHANNEL;
00049
00051     extern const ChannelLabel_T CHANNEL_DN;
00052
00054     extern const ChannelLabel_T CHANNEL_IN;
00055
00057     extern const TripType_T TRIP_TYPE_ONE_WAY;
00058
00060     extern const TripType_T TRIP_TYPE_ROUND_TRIP;
00061
00063     extern const TripType_T TRIP_TYPE_INBOUND;
00064
00066     extern const TripType_T TRIP_TYPE_OUTBOUND;
00067
00069     extern const FrequentFlyer_T DEFAULT_FF_TIER;
00070
00072     extern const PriceValue_T DEFAULT_VALUE_OF_TIME;
00073
00074 }
00075 #endif // __STDAIR_BAS_BASCONST_REQUEST_HPP

```

35.39 stdair/basic/BasConst_TravelSolution.hpp File Reference

```
#include <stdair/stdair_types.hpp>
```

Namespaces

- namespace `stdair`
Handle on the `StdAir` library context.

Variables

- const `Duration_T` `stdair::DEFAULT_MINIMAL_CONNECTION_TIME`
- const `Duration_T` `stdair::DEFAULT_MAXIMAL_CONNECTION_TIME`
- const `FlightPathCode_T` `stdair::DEFAULT_FLIGHTPATH_CODE`
- const `Availability_T` `stdair::DEFAULT_CLASS_AVAILABILITY`
- const `AvailabilityStatus_T` `stdair::DEFAULT_AVAILABILITY_STATUS`
- const unsigned short `stdair::DEFAULT_NUMBER_OF_REQUIRED_SEATS`
- const `MatchingIndicator_T` `stdair::DEFAULT_MATCHING_INDICATOR`
- const `AirlineCode_T` `stdair::DEFAULT_DICO_STUDIED_AIRLINE`

35.40 BasConst_TravelSolution.hpp

```
00001 #ifndef __STDAIR_BAS_BASCONST_TRAVELSOLUTION_HPP
00002 #define __STDAIR_BAS_BASCONST_TRAVELSOLUTION_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_types.hpp>
00009
00010 namespace stdair {
00011
00012 // ////////// Travel Solutions //////////
00014 extern const Distance_T DEFAULT_DISTANCE_VALUE;
00015
00017 extern const Duration_T DEFAULT_MINIMAL_CONNECTION_TIME;
00018
00020 extern const Duration_T DEFAULT_MAXIMAL_CONNECTION_TIME;
00021
00023 extern const Duration_T NULL_BOOST_TIME_DURATION;
00024
00026 extern const FlightPathCode_T DEFAULT_FLIGHTPATH_CODE;
00027
00029 extern const Availability_T DEFAULT_CLASS_AVAILABILITY;
00030
00032 extern const AvailabilityStatus_T DEFAULT_AVAILABILITY_STATUS;
00033
00035 extern const unsigned short DEFAULT_NUMBER_OF_REQUIRED_SEATS;
00036
00039 extern const MatchingIndicator_T DEFAULT_MATCHING_INDICATOR;
00040
00042 extern const Revenue_T DEFAULT_REVENUE_VALUE;
```

```

00043
00045     extern const AirlineCode_T DEFAULT_DICO_STUDIED_AIRLINE;
00046
00048     extern const Date_T DEFAULT_DICO_STUDIED_DATE;
00049
00050 }
00051 #endif // __STDAIR_BAS_BASCONST_TRAVELSOLUTION_HPP

```

35.41 stdair/basic/BasConst_Yield.hpp File Reference

```
#include <stdair/stdair_types.hpp>
```

Namespaces

- namespace `stdair`
Handle on the *StdAir* library context.

Variables

- const Yield_T `stdair::DEFAULT_YIELD_VALUE`
- const Yield_T `stdair::DEFAULT_YIELD_MAX_VALUE`

35.42 BasConst_Yield.hpp

```

00001 #ifndef __STDAIR_BAS_BASCONST_YIELD_HPP
00002 #define __STDAIR_BAS_BASCONST_YIELD_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_types.hpp>
00009
00010 namespace stdair {
00011
00012     // ////////// (Leg-)Yield-related BOM //////////
00014     extern const Yield_T DEFAULT_YIELD_VALUE;
00015
00017     extern const Yield_T DEFAULT_YIELD_MAX_VALUE;
00018
00019 }
00020 #endif // __STDAIR_BAS_BASCONST_YIELD_HPP

```

35.43 stdair/basic/BasDBParams.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/basic/BasDBParams.hpp>

```

Namespaces

- namespace `stdair`

Handle on the StdAir library context.

35.44 BasDBParams.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasDBParams.hpp>
00009
00010 namespace stdair {
00011
00012 // //////////////////////////////////////
00013 BasDBParams::BasDBParams() {
00014 }
00015
00016 // //////////////////////////////////////
00017 BasDBParams::BasDBParams(const BasDBParams& iDBParams)
00018 : _user (iDBParams._user), _passwd (iDBParams._passwd),
00019   _host (iDBParams._host), _port (iDBParams._port),
00020   _dbname (iDBParams._dbname) {
00021 }
00022
00023 // //////////////////////////////////////
00024 BasDBParams::BasDBParams(const std::string& iDBUser,
00025                          const std::string& iDBPasswd,
00026                          const std::string& iDBHost,
00027                          const std::string& iDBPort,
00028                          const std::string& iDBName)
00029 : _user (iDBUser), _passwd (iDBPasswd), _host (iDBHost), _port (iDBPort),
00030   _dbname (iDBName) {
00031 }
00032
00033 // //////////////////////////////////////
00034 BasDBParams::~BasDBParams() {
00035 }
00036
00037 // //////////////////////////////////////
00038 const std::string BasDBParams::describe() const {
00039     return toString();
00040 }
00041
00042 // //////////////////////////////////////
00043 std::string BasDBParams::toShortString() const {
00044     std::ostringstream oStr;
00045     oStr << _dbname << "." << _user << "@" << _host << ":" << _port;
00046     return oStr.str();
00047 }
00048
00049 // //////////////////////////////////////
00050 std::string BasDBParams::toString() const {
00051     std::ostringstream oStr;
00052     oStr << _dbname << "." << _user << "@" << _host << ":" << _port;
00053     return oStr.str();

```

```

00054     }
00055
00056     // //////////////////////////////////////
00057     bool BasDBParams::check() const {
00058         if (_user.empty() == true || _passwd.empty() == true
00059             || _host.empty() == true || _port.empty()
00060             || _dbname.empty() == true) {
00061             return false;
00062         }
00063         return true;
00064     }
00065
00066 }

```

35.45 stdair/basic/BasDBParams.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/stdair_db.hpp>
#include <stdair/basic/StructAbstract.hpp>

```

Classes

- struct [stdair::BasDBParams](#)
Structure holding the parameters for connection to a database.

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.46 BasDBParams.hpp

```

00001 #ifndef __STDAIR_BAS_BASDBPARAMS_HPP
00002 #define __STDAIR_BAS_BASDBPARAMS_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // Stdair
00011 #include <stdair/stdair_db.hpp>
00012 #include <stdair/basic/StructAbstract.hpp>
00013
00014 namespace stdair {
00015
00019     struct BasDBParams : public StructAbstract {
00020     public:
00021         // //////////// Getters ////////////

```

```
00023     const std::string& getUser() const {
00024         return _user;
00025     }
00026
00028     const std::string& getPassword() const {
00029         return _passwd;
00030     }
00031
00033     const std::string& getHost() const {
00034         return _host;
00035     }
00036
00038     const std::string& getPort() const {
00039         return _port;
00040     }
00041
00043     const std::string& getDBName() const {
00044         return _dbname;
00045     }
00046
00047
00048     // ////////// Setters //////////
00050     void setUser (const std::string& iUser) {
00051         _user = iUser;
00052     }
00053
00055     void setPassword (const std::string& iPasswd) {
00056         _passwd = iPasswd;
00057     }
00058
00060     void setHost (const std::string& iHost) {
00061         _host = iHost;
00062     }
00063
00065     void setPort (const std::string& iPort) {
00066         _port = iPort;
00067     }
00068
00070     void setDBName (const std::string& iDBName) {
00071         _dbname = iDBName;
00072     }
00073
00074
00075     public:
00076         // ////////// Busines methods //////////
00080         bool check() const;
00081
00082
00083     public:
00084         // ////////// Display methods //////////
00088         const std::string describe() const;
00089
00093         std::string toShortString() const;
00094
00098         std::string toString() const;
00099
00100
00101     public:
00105         BasDBParams (const std::string& iDBUser, const std::string& iDBPasswd,
00106                     const std::string& iDBHost, const std::string& iDBPort,
00107                     const std::string& iDBName);
00108
```

```

00112     BasDBParams();
00113
00117     BasDBParams (const BasDBParams&);
00118
00122     ~BasDBParams();
00123
00124
00125 private:
00126     // ////////// Attributes //////////
00128     std::string _user;
00130     std::string _passwd;
00132     std::string _host;
00134     std::string _port;
00136     std::string _dbname;
00137 };
00138
00139 }
00140 #endif // __STDAIR_BASIC_BASDBPARAMS_HPP

```

35.47 stdair/basic/BasFileMgr.cpp File Reference

```

#include <cassert>
#include <boost/version.hpp>
#include <boost/filesystem/path.hpp>
#include <boost/filesystem/operations.hpp>
#include <stdair/basic/BasFileMgr.hpp>

```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

35.48 BasFileMgr.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // Boost (STL Extension)
00007 // Boost Filesystem (http://www.boost.org/doc/libs/1_41_0/libs/filesystem/doc/ind
ex.htm)
00008 #include <boost/version.hpp>
00009 #if BOOST_VERSION >= 103500
00010 #include <boost/filesystem.hpp>
00011 #else // BOOST_VERSION >= 103500
00012 #include <boost/filesystem/path.hpp>
00013 #include <boost/filesystem/operations.hpp>
00014 #endif // BOOST_VERSION >= 103500
00015 // StdAir
00016 #include <stdair/basic/BasFileMgr.hpp>
00017

```

```

00018 namespace boostfs = boost::filesystem;
00019
00020 namespace stdair {
00021
00022     // //////////////////////////////////////
00023     bool BasFileMgr::doesExistAndIsReadable (const std::string& iFilepath) {
00024         bool oFine = false;
00025
00026         boostfs::path lPath (iFilepath);
00027
00028         if (boostfs::exists (lPath) == false) {
00029             return oFine;
00030         }
00031
00032         #if BOOST_VERSION >= 103500
00033             if (boostfs::is_regular (lPath) == true) {
00034                 oFine = true;
00035             }
00036         #endif // BOOST_VERSION >= 103500
00037
00038         return oFine;
00039     }
00040
00041 }

```

35.49 stdair/basic/BasFileMgr.hpp File Reference

```
#include <string>
```

Classes

- struct [stdair::BasFileMgr](#)

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.50 BasFileMgr.hpp

```

00001 #ifndef __STDAIR_BAS_BASFILEMGR_HPP
00002 #define __STDAIR_BAS_BASFILEMGR_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009
00010 namespace stdair {
00011
00012     struct BasFileMgr {
00013     public:
00014
00015     }

```

```

00016 // ////////////////// Functional Support Methods //////////////////
00017 static bool doesExistAndIsReadable (const std::string& iFilepath);
00018 };
00019 };
00020 };
00021 }
00022 #endif // __STDAIR_BAS_BASFILEMGR_HPP

```

35.51 stdair/basic/BasLogParams.cpp File Reference

```

#include <cassert>
#include <iostream>
#include <sstream>
#include <stdair/basic/BasLogParams.hpp>

```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

35.52 BasLogParams.cpp

```

00001 // ////////////////// //////////////////////////////////////////
00002 // Import section
00003 // ////////////////// //////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <iostream>
00007 #include <sstream>
00008 // StdAir
00009 #include <stdair/basic/BasLogParams.hpp>
00010
00011 namespace stdair {
00012
00013 // ////////////////// //////////////////////////////////////////
00014 BasLogParams::BasLogParams()
00015 : _logLevel (LOG::DEBUG), _logStream (std::cout),
00016   _forceMultipleInit (false) {
00017     assert (false);
00018 }
00019
00020 // ////////////////// //////////////////////////////////////////
00021 BasLogParams::BasLogParams (const BasLogParams& iLogParams)
00022 : _logLevel (iLogParams._logLevel), _logStream (iLogParams._logStream),
00023   _forceMultipleInit (iLogParams._forceMultipleInit) {
00024 }
00025
00026 // ////////////////// //////////////////////////////////////////
00027 BasLogParams::BasLogParams (const LOG::EN_LogLevel iLogLevel,
00028                             std::ostream& ioLogOutputStream,
00029                             const bool iForceMultipleInstance)
00030 : _logLevel (iLogLevel), _logStream (ioLogOutputStream),
00031   _forceMultipleInit (iForceMultipleInstance) {

```

```

00032     }
00033
00034     // //////////////////////////////////////
00035     BasLogParams::~BasLogParams() {
00036     }
00037
00038     // //////////////////////////////////////
00039     const std::string BasLogParams::describe() const {
00040         return toString();
00041     }
00042
00043     // //////////////////////////////////////
00044     std::string BasLogParams::toShortString() const {
00045         const std::string isForcedStr = (_forceMultipleInit == true)? " (forced)":"";
00046         std::ostringstream ostr;
00047         ostr << LOG::_logLevels[_logLevel] << isForcedStr;
00048         return ostr.str();
00049     }
00050
00051     // //////////////////////////////////////
00052     std::string BasLogParams::toString() const {
00053         std::ostringstream ostr;
00054         ostr << LOG::_logLevels[_logLevel];
00055         return ostr.str();
00056     }
00057
00058 }

```

35.53 stdair/basic/BasLogParams.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/stdair_log.hpp>
#include <stdair/basic/StructAbstract.hpp>

```

Classes

- struct [stdair::BasLogParams](#)
Structure holding parameters for logging.

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.54 BasLogParams.hpp

```

00001 #ifndef __STDAIR_BAS_BASLOGPARAMS_HPP
00002 #define __STDAIR_BAS_BASLOGPARAMS_HPP
00003
00004 // //////////////////////////////////////

```

```

00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // Stdair
00011 #include <stdair/stdair_log.hpp>
00012 #include <stdair/basic/StructAbstract.hpp>
00013
00014 namespace stdair {
00015
00019     struct BasLogParams : public StructAbstract {
00020         friend class Logger;
00021     public:
00022         // ////////// Getters //////////
00026         const LOG::EN_LogLevel& getLogLevel() const {
00027             return _logLevel;
00028         }
00029
00033         std::ostream& getLogStream() const {
00034             return _logStream;
00035         }
00036
00040         const bool getForcedInitialisationFlag() const {
00041             return _forceMultipleInit;
00042         }
00043
00044
00045         // ////////// Setters //////////
00049         void setForcedInitialisationFlag (const bool iForceMultipleInstance) {
00050             _forceMultipleInit = iForceMultipleInstance;
00051         }
00052
00053
00054     public:
00055         // ////////// Busines methods //////////
00059         bool check() const;
00060
00061
00062     public:
00063         // ////////// Display methods //////////
00067         const std::string describe() const;
00068
00072         std::string toShortString() const;
00073
00077         std::string toString() const;
00078
00079
00080     public:
00089         BasLogParams (const LOG::EN_LogLevel iLogLevel,
00090                     std::ostream& ioLogOutputStream,
00091                     const bool iForceMultipleInstance = false);
00092
00096         BasLogParams (const BasLogParams&);
00097
00101         ~BasLogParams();
00102
00103     private:
00107         BasLogParams();
00108
00109
00110     private:

```

```

00111      // ////////// Attributes //////////
00115      const LOG::EN_LogLevel _logLevel;
00116
00120      std::ostream& _logStream;
00121
00135      bool _forceMultipleInit;
00136  };
00137
00138  }
00139  #endif // __STDAIR_BAS_BASLOGPARAMS_HPP

```

35.55 stdair/basic/BasParserHelperTypes.hpp File Reference

```

#include <string>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/service/Logger.hpp>

```

Classes

- struct [stdair::date_time_element< MIN, MAX >](#)

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

Typedefs

- typedef date_time_element< 0, 23 > [stdair::hour_t](#)
- typedef date_time_element< 0, 59 > [stdair::minute_t](#)
- typedef date_time_element< 0, 59 > [stdair::second_t](#)
- typedef date_time_element< 1900, 2100 > [stdair::year_t](#)
- typedef date_time_element< 1, 12 > [stdair::month_t](#)
- typedef date_time_element< 1, 31 > [stdair::day_t](#)

Functions

- template<int MIN, int MAX>
date_time_element< MIN, MAX > [stdair::operator*](#) (const date_time_element< MIN, MAX > &o1, const date_time_element< MIN, MAX > &o2)
- template<int MIN, int MAX>
date_time_element< MIN, MAX > [stdair::operator+](#) (const date_time_element< MIN, MAX > &o1, const date_time_element< MIN, MAX > &o2)

35.56 BasParserHelperTypes.hpp

```

00001 #ifndef __STDAIR_BAS_BASCOMPARSERHELPERTYPES_HPP
00002 #define __STDAIR_BAS_BASCOMPARSERHELPERTYPES_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <sstream>
00010 // StdAir
00011 #include <stdair/stdair_exceptions.hpp>
00012 #include <stdair/service/Logger.hpp>
00013
00014 namespace stdair {
00015
00016 // //////////////////////////////////////
00017 //
00018 // Parser structure helper
00019 //
00020 // //////////////////////////////////////
00022 template <int MIN = 0, int MAX = 0>
00023 struct date_time_element {
00024     unsigned int _value;
00025
00026     // ////////////////////////////////// Constructors //////////////////////////////////
00028     date_time_element () { }
00030     date_time_element (const date_time_element& t) : _value (t._value) { }
00032     date_time_element (int i) : _value (i) { }
00034     void check () const {
00035         if (_value < MIN || _value > MAX) {
00036             std::ostringstream oMessage;
00037             oMessage << "The value: " << _value << " is out of range ("
00038                 << MIN << ", " << MAX << ")";
00039             throw stdair::ParserException (oMessage.str());
00040         }
00041     }
00042 };
00043
00045 template <int MIN, int MAX>
00046 inline date_time_element<MIN,
00047     MAX> operator*(const date_time_element<MIN, MAX>& o1,
00048     const date_time_element<MIN, MAX>& o2) {
00049     return date_time_element<MIN, MAX> (o1._value * o2._value);
00050 }
00051
00053 template <int MIN, int MAX>
00054 inline date_time_element<MIN,
00055     MAX> operator+(const date_time_element<MIN, MAX>& o1,
00056     const date_time_element<MIN, MAX>& o2) {
00057     return date_time_element<MIN, MAX> (o1._value + o2._value);
00058 }
00059
00061 typedef date_time_element<0, 23> hour_t;
00062 typedef date_time_element<0, 59> minute_t;
00063 typedef date_time_element<0, 59> second_t;
00064 typedef date_time_element<1900, 2100> year_t;
00065 typedef date_time_element<1, 12> month_t;
00066 typedef date_time_element<1, 31> day_t;

```

```

00067
00068 }
00069 #endif // __STDAIR_BAS_BASCOMPASERHELPERTYPES_HPP

```

35.57 stdair/basic/BasParserTypes.hpp File Reference

```

#include <string>
#include <boost/spirit/include/qi.hpp>
#include <boost/spirit/include/phoenix_core.hpp>
#include <boost/spirit/include/phoenix_operator.hpp>
#include <boost/spirit/include/support_multi_pass.hpp>
#include <stdair/basic/BasParserHelperTypes.hpp>

```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

Typedefs

- typedef std::istreambuf_iterator< char > [stdair::base_iterator_t](#)
- typedef boost::spirit::multi_pass< base_iterator_t > [stdair::iterator_t](#)
- typedef boost::spirit::qi::int_parser< unsigned int, 10, 1, 1 > [stdair::int1_p_t](#)
- typedef boost::spirit::qi::uint_parser< int, 10, 2, 2 > [stdair::uint2_p_t](#)
- typedef boost::spirit::qi::uint_parser< int, 10, 4, 4 > [stdair::uint4_p_t](#)
- typedef boost::spirit::qi::uint_parser< int, 10, 1, 4 > [stdair::uint1_4_p_t](#)
- typedef boost::spirit::qi::uint_parser< hour_t, 10, 2, 2 > [stdair::hour_p_t](#)
- typedef boost::spirit::qi::uint_parser< minute_t, 10, 2, 2 > [stdair::minute_p_t](#)
- typedef boost::spirit::qi::uint_parser< second_t, 10, 2, 2 > [stdair::second_p_t](#)
- typedef boost::spirit::qi::uint_parser< year_t, 10, 4, 4 > [stdair::year_p_t](#)
- typedef boost::spirit::qi::uint_parser< month_t, 10, 2, 2 > [stdair::month_p_t](#)
- typedef boost::spirit::qi::uint_parser< day_t, 10, 2, 2 > [stdair::day_p_t](#)

35.58 BasParserTypes.hpp

```

00001 #ifndef __STDAIR_BAS_BASCOMPASERTYPES_HPP
00002 #define __STDAIR_BAS_BASCOMPASERTYPES_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // Boost Spirit (Parsing)
00010 #include <boost/spirit/include/qi.hpp>
00011 #include <boost/spirit/include/phoenix_core.hpp>

```

```

00012 #include <boost/spirit/include/phoenix_operator.hpp>
00013 #include <boost/spirit/include/support_multi_pass.hpp>
00014 // STDAIR
00015 #include <stdair/basic/BasParserHelperTypes.hpp>
00016
00017 namespace stdair {
00018
00019 // //////////////////////////////////////
00020 //
00021 // Definition of Basic Types
00022 //
00023 // //////////////////////////////////////
00024 // The types of iterator, scanner and rule are then derived from
00025 // the parsing unit.
00026 typedef std::istreambuf_iterator<char> base_iterator_t;
00027 typedef boost::spirit::multi_pass<base_iterator_t> iterator_t;
00028
00029 // //////////////////////////////////////
00030 //
00031 // Parser related types
00032 //
00033 // //////////////////////////////////////
00035 typedef boost::spirit::qi::int_parser<unsigned int, 10, 1, 1> int1_p_t;
00036
00038 typedef boost::spirit::qi::uint_parser<int, 10, 2, 2> uint2_p_t;
00039
00041 typedef boost::spirit::qi::uint_parser<int, 10, 4, 4> uint4_p_t;
00042
00044 typedef boost::spirit::qi::uint_parser<int, 10, 1, 4> uint1_4_p_t;
00045
00047 typedef boost::spirit::qi::uint_parser<hour_t, 10, 2, 2> hour_p_t;
00048 typedef boost::spirit::qi::uint_parser<minute_t, 10, 2, 2> minute_p_t;
00049 typedef boost::spirit::qi::uint_parser<second_t, 10, 2, 2> second_p_t;
00050 typedef boost::spirit::qi::uint_parser<year_t, 10, 4, 4> year_p_t;
00051 typedef boost::spirit::qi::uint_parser<month_t, 10, 2, 2> month_p_t;
00052 typedef boost::spirit::qi::uint_parser<day_t, 10, 2, 2> day_p_t;
00053 }
00054 #endif // __STDAIR_BAS_BASCOMPARSERTYPES_HPP

```

35.59 stdair/basic/BasTypes.hpp File Reference

```
#include <string>
```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

35.60 BasTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BAS_BASTYPES_HPP
00003 #define __STDAIR_BAS_BASTYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section

```

```

00007 // //////////////////////////////////////
00008 // STL
00009 #include <string>
00010
00011 namespace stdair {
00012
00013 }
00014 #endif // __STDAIR_BAS_BASTYPES_HPP

```

35.61 stdair/basic/ContinuousAttributeLite.hpp File Reference

```

#include <cassert>
#include <iosfwd>
#include <string>
#include <vector>
#include <map>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/DictionaryManager.hpp>

```

Classes

- struct [stdair::ContinuousAttributeLite< T >](#)
Class modeling the distribution of values that can be taken by a continuous attribute.

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.62 ContinuousAttributeLite.hpp

```

00001 #ifndef __STDAIR_BAS_CONTINUOUSATTRIBUTE_LITE_HPP
00002 #define __STDAIR_BAS_CONTINUOUSATTRIBUTE_LITE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <cassert>
00009 #include <iosfwd>
00010 #include <string>
00011 #include <vector>
00012 #include <map>
00013 // StdAir
00014 #include <stdair/stdair_basic_types.hpp>
00015 // TraDemGen

```

```

00016 #include <stdair/stdair_exceptions.hpp>
00017 #include <stdair/basic/DictionaryManager.hpp>
00018
00019 namespace stdair {
00020
00021     template <typename T>
00022     struct ContinuousAttributeLite {
00023     public:
00024         // /////////////////////////////////////////////////////////////////// Type definitions ///////////////////////////////////////////////////////////////////
00025         typedef std::map<T, stdair::Probability_T> ContinuousDistribution_T;
00026
00027     public:
00028         // /////////////////////////////////////////////////////////////////// Business Methods ///////////////////////////////////////////////////////////////////
00029         const T getValue(const stdair::Probability_T& iCumulativeProbability) const {
00030             const DictionaryKey_T& lKey =
00031                 DictionaryManager::valueToKey (iCumulativeProbability);
00032
00033             // Find the first cumulative probability value greater or equal to lKey.
00034             unsigned int idx = 0;
00035             for (; idx < _size; ++idx) {
00036                 if (_cumulativeDistribution.at(idx) > lKey) {
00037                     break;
00038                 }
00039             }
00040
00041             if (idx == 0) {
00042                 return _valueArray.at(idx);
00043             }
00044             if (idx == _size) {
00045                 return _valueArray.at(idx-1);
00046             }
00047
00048             //
00049             const stdair::Probability_T& lCumulativeCurrentPoint =
00050                 DictionaryManager::keyToValue (_cumulativeDistribution.at(idx));
00051             const T& lValueCurrentPoint = _valueArray.at(idx);
00052
00053             //
00054             const stdair::Probability_T& lCumulativePreviousPoint =
00055                 DictionaryManager::keyToValue (_cumulativeDistribution.at(idx-1));
00056             const T& lValuePreviousPoint = _valueArray.at(idx-1);
00057
00058             if (lCumulativePreviousPoint == lCumulativeCurrentPoint) {
00059                 return lValuePreviousPoint;
00060             }
00061
00062             T oValue= lValuePreviousPoint + (lValueCurrentPoint - lValuePreviousPoint)
00063                 * (iCumulativeProbability - lCumulativePreviousPoint)
00064                 / (lCumulativeCurrentPoint - lCumulativePreviousPoint);
00065
00066             return oValue;
00067         }
00068     public:
00069         // /////////////////////////////////////////////////////////////////// Business Methods ///////////////////////////////////////////////////////////////////
00070         const stdair::Probability_T getRemainingProportion(const T& iValue) const {
00071
00072             // Find the first value greater than iValue.
00073             unsigned int idx = 0;
00074             for (; idx < _size; ++idx) {
00075                 if (_valueArray.at(idx) > iValue) {

```

```

00090         break;
00091     }
00092 }
00093 if (idx == 0) {
00094     const stdair::Probability_T& oCumulativeProbability =
00095         DictionaryManager::keyToValue (_cumulativeDistribution.at(idx));
00096     return 1 - oCumulativeProbability;
00097 }
00098 if (idx == _size) {
00099     const stdair::Probability_T& oCumulativeProbability =
00100         DictionaryManager::keyToValue (_cumulativeDistribution.at(idx-1));
00101     return 1 - oCumulativeProbability;
00102 }
00103
00104 //
00105 const stdair::Probability_T& lCumulativeCurrentPoint =
00106     DictionaryManager::keyToValue (_cumulativeDistribution.at(idx));
00107 const T& lValueCurrentPoint = _valueArray.at(idx);
00108
00109 //
00110 const stdair::Probability_T& lCumulativePreviousPoint =
00111     DictionaryManager::keyToValue (_cumulativeDistribution.at(idx-1));
00112 const T& lValuePreviousPoint = _valueArray.at(idx-1);
00113
00114 if (lValuePreviousPoint == lValueCurrentPoint) {
00115     return 1 - lCumulativePreviousPoint;
00116 }
00117
00118 const stdair::Probability_T& oCumulativeProbability =
00119     lCumulativePreviousPoint + (lCumulativeCurrentPoint - lCumulativePrevious
Point)
00120     * (iValue - lValuePreviousPoint)
00121     / (lValueCurrentPoint - lValuePreviousPoint);
00122
00123     return 1 - oCumulativeProbability;
00124 }
00125
00126 public:
00127     // ////////////////////////////////// Business Methods //////////////////////////////////
00131     const double getDerivativeValue(const T iKey) const{
00132
00133         // Find the first key value greater or equal to iKey.
00134         unsigned int idx = 0;
00135         for (; idx < _size; ++idx) {
00136             if (_valueArray.at(idx) > iKey) {
00137                 break;
00138             }
00139         }
00140         assert (idx != 0);
00141         assert (idx != _size);
00142
00143         //
00144         const stdair::Probability_T& lCumulativeCurrentPoint =
00145             DictionaryManager::keyToValue (_cumulativeDistribution.at(idx));
00146         const T& lValueCurrentPoint = _valueArray.at(idx);
00147
00148         //
00149         const stdair::Probability_T& lCumulativePreviousPoint =
00150             DictionaryManager::keyToValue (_cumulativeDistribution.at(idx-1));
00151         const T& lValuePreviousPoint = _valueArray.at(idx-1);
00152         assert (lValueCurrentPoint != lValuePreviousPoint);
00153

```

```

00154         const double oValue= (lCumulativeCurrentPoint - lCumulativePreviousPoint)
00155         / (lValueCurrentPoint - lValuePreviousPoint);
00156
00157         return oValue;
00158     }
00159
00163     const T getUpperBound (const T iKey) const {
00164         // Find the first key value greater or equal to iKey.
00165         unsigned int idx = 0;
00166         for (; idx < _size; ++idx) {
00167             if (_valueArray.at(idx) > iKey) {
00168                 break;
00169             }
00170         }
00171         assert (idx != 0);
00172         assert (idx != _size);
00173
00174         return _valueArray.at (idx);
00175     }
00176
00177 public:
00178     // //////////// Display Support Methods ////////////
00182     const std::string displayCumulativeDistribution() const {
00183         std::ostringstream oStr;
00184
00185         for (unsigned int idx = 0; idx < _size; ++idx) {
00186             if (idx != 0) {
00187                 oStr << ", ";
00188             }
00189
00190             const stdair::Probability_T lProbability =
00191                 DictionaryManager::keyToValue (_cumulativeDistribution.at(idx));
00192
00193             oStr << _valueArray.at(idx) << ":" << lProbability;
00194         }
00195         return oStr.str();
00196     }
00197
00198
00199 public:
00200     // //////////// Constructors and destructors ////////////
00204     ContinuousAttributeLite (const ContinuousDistribution_T& iValueMap)
00205         : _size (iValueMap.size()) {
00206         init (iValueMap);
00207     }
00208
00212     ContinuousAttributeLite (const ContinuousAttributeLite& iCAL)
00213         : _size (iCAL._size),
00214         _cumulativeDistribution (iCAL._cumulativeDistribution),
00215         _valueArray (iCAL._valueArray) {
00216     }
00217
00221     ContinuousAttributeLite& operator= (const ContinuousAttributeLite& iCAL) {
00222         _size = iCAL._size;
00223         _cumulativeDistribution = iCAL._cumulativeDistribution;
00224         _valueArray = iCAL._valueArray;
00225         return *this;
00226     }
00227
00231     virtual ~ContinuousAttributeLite() {
00232     }
00233

```

```

00234 private:
00238 ContinuousAttributeLite() : _size(1) {
00239 }
00240
00245 void init (const ContinuousDistribution_T& iValueMap) {
00246     //
00247     const unsigned int lSize = iValueMap.size();
00248     _cumulativeDistribution.reserve (lSize);
00249     _valueArray.reserve (lSize);
00250
00251     // Browse the map to retrieve the values and cumulative probabilities.
00252     for (typename ContinuousDistribution_T::const_iterator it =
00253         iValueMap.begin(); it != iValueMap.end(); ++it) {
00254
00255         const T& attributeValue = it->first;
00256         const DictionaryKey_T& lKey = DictionaryManager::valueToKey (it->second);
00257
00258         // Build the two arrays.
00259         _cumulativeDistribution.push_back (lKey);
00260         _valueArray.push_back (attributeValue);
00261     }
00262 }
00263
00264 private:
00265 // ////////// Attributes //////////
00266 unsigned int _size;
00271
00275 std::vector<DictionaryKey_T> _cumulativeDistribution;
00276
00280 std::vector<T> _valueArray;
00281 };
00282
00283 }
00284 #endif // __STDAIR_BAS_CONTINUOUSATTRIBUTE_LITE_HPP

```

35.63 stdair/basic/DemandGenerationMethod.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/DemandGenerationMethod.hpp>

```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

35.64 DemandGenerationMethod.cpp

```

00001 // //////////////////////////////////////
00002 // Import section

```

```

00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/stdair_exceptions.hpp>
00009 #include <stdair/basic/DemandGenerationMethod.hpp>
00010
00011 namespace stdair {
00012
00013 // //////////////////////////////////////
00014 const std::string DemandGenerationMethod::_labels[LAST_VALUE] =
00015     { "PoissonProcess", "StatisticsOrder" };
00016
00017 // //////////////////////////////////////
00018 const char DemandGenerationMethod::_methodLabels[LAST_VALUE] = { 'P', 'S' };
00019
00020
00021 // //////////////////////////////////////
00022 DemandGenerationMethod::DemandGenerationMethod() : _method (LAST_VALUE) {
00023     assert (false);
00024 }
00025
00026 // //////////////////////////////////////
00027 DemandGenerationMethod::
00028 DemandGenerationMethod (const DemandGenerationMethod& iDemandGenerationMethod)
00029     : _method (iDemandGenerationMethod._method) {
00030 }
00031
00032 // //////////////////////////////////////
00033 DemandGenerationMethod::
00034 DemandGenerationMethod (const EN_DemandGenerationMethod& iDemandGenerationMetho
00035 d)
00036     : _method (iDemandGenerationMethod) {
00037 }
00038
00039 // //////////////////////////////////////
00040 DemandGenerationMethod::EN_DemandGenerationMethod
00041 DemandGenerationMethod::getMethod (const char iMethodChar) {
00042     EN_DemandGenerationMethod oMethod;
00043     switch (iMethodChar) {
00044     case 'P': oMethod = POI_PRO; break;
00045     case 'S': oMethod = STA_ORD; break;
00046     default: oMethod = LAST_VALUE; break;
00047     }
00048
00049     if (oMethod == LAST_VALUE) {
00050         const std::string& lLabels = describeLabels();
00051         std::ostringstream oMessage;
00052         oMessage << "The demand (booking request) generation method '"
00053             << iMethodChar
00054             << "' is not known. Known demand (booking request) generation "
00055             << "methods: " << lLabels;
00056         throw CodeConversionException (oMessage.str());
00057     }
00058
00059     return oMethod;
00060 }
00061
00062 // //////////////////////////////////////
00063 DemandGenerationMethod::DemandGenerationMethod (const char iMethodChar)
00064     : _method (getMethod (iMethodChar)) {

```

```

00064     }
00065
00066     // //////////////////////////////////////
00067     DemandGenerationMethod::
00068     DemandGenerationMethod (const std::string& iMethodStr) {
00069         //
00070         const size_t lSize = iMethodStr.size();
00071         assert (lSize == 1);
00072         const char lMethodChar = iMethodStr[0];
00073         _method = getMethod (lMethodChar);
00074     }
00075
00076     // //////////////////////////////////////
00077     const std::string& DemandGenerationMethod::
00078     getLabel (const EN_DemandGenerationMethod& iMethod) {
00079         return _labels[iMethod];
00080     }
00081
00082     // //////////////////////////////////////
00083     char DemandGenerationMethod::
00084     getMethodLabel (const EN_DemandGenerationMethod& iMethod) {
00085         return _methodLabels[iMethod];
00086     }
00087
00088     // //////////////////////////////////////
00089     std::string DemandGenerationMethod::
00090     getMethodLabelAsString (const EN_DemandGenerationMethod& iMethod) {
00091         std::ostringstream ostr;
00092         ostr << _methodLabels[iMethod];
00093         return ostr.str();
00094     }
00095
00096     // //////////////////////////////////////
00097     std::string DemandGenerationMethod::describeLabels() {
00098         std::ostringstream ostr;
00099         for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
00100             if (idx != 0) {
00101                 ostr << ", ";
00102             }
00103             ostr << _labels[idx];
00104         }
00105         return ostr.str();
00106     }
00107
00108     // //////////////////////////////////////
00109     DemandGenerationMethod::EN_DemandGenerationMethod
00110     DemandGenerationMethod::getMethod() const {
00111         return _method;
00112     }
00113
00114     // //////////////////////////////////////
00115     char DemandGenerationMethod::getMethodAsChar() const {
00116         const char oMethodChar = _methodLabels[_method];
00117         return oMethodChar;
00118     }
00119
00120     // //////////////////////////////////////
00121     std::string DemandGenerationMethod::getMethodAsString() const {
00122         std::ostringstream ostr;
00123         ostr << _methodLabels[_method];
00124         return ostr.str();
00125     }

```

```

00126
00127 // //////////////////////////////////////
00128 const std::string DemandGenerationMethod::describe() const {
00129     std::ostringstream ostr;
00130     ostr << _labels[_method];
00131     return ostr.str();
00132 }
00133
00134 // //////////////////////////////////////
00135 bool DemandGenerationMethod::
00136 operator== (const EN_DemandGenerationMethod& iMethod) const {
00137     return (_method == iMethod);
00138 }
00139
00140 }

```

35.65 stdair/basic/DemandGenerationMethod.hpp File Reference

```

#include <string>
#include <stdair/basic/StructAbstract.hpp>

```

Classes

- struct [stdair::DemandGenerationMethod](#)
Enumeration of demand (booking request) generation methods.

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.66 DemandGenerationMethod.hpp

```

00001 #ifndef __STDAIR_BAS_DEMANDGENERATIONMETHOD_HPP
00002 #define __STDAIR_BAS_DEMANDGENERATIONMETHOD_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/basic/StructAbstract.hpp>
00011
00012 namespace stdair {
00013
00017     struct DemandGenerationMethod : public StructAbstract {
00018     public:
00019         typedef enum {
00020             POI_PRO = 0,
00021             STA_ORD,
00022             LAST_VALUE

```

```

00023     } EN_DemandGenerationMethod;
00024
00028     static const std::string& getLabel (const EN_DemandGenerationMethod&);
00029
00033     static EN_DemandGenerationMethod getMethod (const char);
00034
00038     static char getMethodLabel (const EN_DemandGenerationMethod&);
00039
00043     static std::string getMethodLabelAsString (const EN_DemandGenerationMethod&);
00044
00048     static std::string describeLabels();
00049
00053     EN_DemandGenerationMethod getMethod() const;
00054
00058     char getMethodAsChar() const;
00059
00063     std::string getMethodAsString() const;
00064
00069     const std::string describe() const;
00070
00071 public:
00075     bool operator== (const EN_DemandGenerationMethod&) const;
00076
00077 public:
00081     DemandGenerationMethod (const EN_DemandGenerationMethod&);
00085     DemandGenerationMethod (const char iMethod);
00089     DemandGenerationMethod (const std::string& iMethod);
00093     DemandGenerationMethod (const DemandGenerationMethod&);
00094
00095 private:
00099     DemandGenerationMethod();
00100
00101
00102 private:
00106     static const std::string _labels[LAST_VALUE];
00110     static const char _methodLabels[LAST_VALUE];
00111
00112 private:
00113     // ////////// Attributes //////////
00117     EN_DemandGenerationMethod _method;
00118 };
00119
00120 }
00121 #endif // __STDAIR_BAS_DEMANDGENERATIONMETHOD_HPP

```

35.67 stdair/basic/DictionaryManager.cpp File Reference

```

#include <stdair/basic/DictionaryManager.hpp>
#include <stdair/basic/BasConst_General.hpp>

```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

35.68 DictionaryManager.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // StdAir
00005 #include <stdair/basic/DictionaryManager.hpp>
00006 #include <stdair/basic/BasConst_General.hpp>
00007
00008 namespace stdair {
00009
00010 // //////////////////////////////////////
00011 const stdair::Probability_T DictionaryManager::
00012 keyToValue (const DictionaryKey_T iKey) {
00013     const float lValue =
00014         static_cast<float> (iKey) / DEFAULT_NUMBER_OF_SUBDIVISIONS;
00015     const stdair::Probability_T lProbability (lValue);
00016     return lProbability;
00017 }
00018
00019 // //////////////////////////////////////
00020 const DictionaryKey_T DictionaryManager::
00021 valueToKey (const stdair::Probability_T iValue) {
00022     const unsigned short lValueMultipliedByThousand =
00023         static_cast<unsigned short> (iValue) * DEFAULT_NUMBER_OF_SUBDIVISIONS;
00024     const DictionaryKey_T lDictionaryKey (lValueMultipliedByThousand);
00025     return lDictionaryKey;
00026 }
00027
00028 }

```

35.69 stdair/basic/DictionaryManager.hpp File Reference

```
#include <stdair/stdair_maths_types.hpp>
```

Classes

- class [stdair::DictionaryManager](#)
Class wrapper of dictionary business methods.

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

Typedefs

- typedef unsigned short [stdair::DictionaryKey_T](#)

35.70 DictionaryManager.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BASIC_DICTIONARYMANAGER_HPP
00003 #define __STDAIR_BASIC_DICTIONARYMANAGER_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // StdAir
00009 #include <stdair/stdair_maths_types.hpp>
00010
00011 namespace stdair {
00012
00013     // ////////////////////////////////// Type definitions //////////////////////////////////
00017     typedef unsigned short DictionaryKey_T;
00018
00022     class DictionaryManager {
00023     public:
00024         // ////////////////////////////////// Business methods //////////////////////////////////
00028         static const stdair::Probability_T keyToValue (const DictionaryKey_T);
00029
00033         static const DictionaryKey_T valueToKey (const stdair::Probability_T);
00034     };
00035 }
00036 #endif // __STDAIR_BASIC_DICTIONARYMANAGER_HPP

```

35.71 stdair/basic/EventType.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/EventType.hpp>

```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

35.72 EventType.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/stdair_exceptions.hpp>
00009 #include <stdair/basic/EventType.hpp>
00010
00011 namespace stdair {
00012

```

```

00013 // //////////////////////////////////////
00014 const std::string EventType::_labels[LAST_VALUE] =
00015 { "BookingRequest", "Cancellation", "OptimisationNotificationForFlightDate",
00016   "OptimisationNotificationForNetwork", "ScheduleChange", "Snapshot",
00017   "RevenueMangement", "BreakPoint" };
00018
00019 // //////////////////////////////////////
00020 const char EventType::_typeLabels[LAST_VALUE] = { 'B', 'X', 'F', 'N', 'C', 'S', 'R', 'P' };
00021
00022
00023
00024 // //////////////////////////////////////
00025 EventType::EventType()
00026 : _type (LAST_VALUE) {
00027     assert (false);
00028 }
00029
00030 // //////////////////////////////////////
00031 EventType::EventType (const EventType& iEventType)
00032 : _type (iEventType._type) {
00033 }
00034
00035 // //////////////////////////////////////
00036 EventType::EventType (const EN_EventType& iEventType)
00037 : _type (iEventType) {
00038 }
00039
00040 // //////////////////////////////////////
00041 EventType::EventType (const char iType) {
00042     switch (iType) {
00043     case 'B': _type = BKG_REQ; break;
00044     case 'X': _type = CX; break;
00045     case 'F': _type = OPT_NOT_4_FD; break;
00046     case 'N': _type = OPT_NOT_4_NET; break;
00047     case 'C': _type = SKD_CHG; break;
00048     case 'S': _type = SNAPSHOT; break;
00049     case 'R': _type = RM; break;
00050     case 'P': _type = BRK_PT; break;
00051     default: _type = LAST_VALUE; break;
00052     }
00053
00054     if (_type == LAST_VALUE) {
00055         const std::string& lLabels = describeLabels();
00056         std::ostringstream oMessage;
00057         oMessage << "The event type '" << iType
00058             << "' is not known. Known event types: " << lLabels;
00059         throw CodeConversionException (oMessage.str());
00060     }
00061 }
00062
00063 // //////////////////////////////////////
00064 const std::string& EventType::getLabel (const EN_EventType& iType) {
00065     return _labels[iType];
00066 }
00067
00068 // //////////////////////////////////////
00069 char EventType::getTypeLabel (const EN_EventType& iType) {
00070     return _typeLabels[iType];
00071 }
00072
00073 // //////////////////////////////////////
00074 std::string EventType::getTypeLabelAsString (const EN_EventType& iType) {

```

```

00075     std::ostringstream ostr;
00076     ostr << _typeLabels[iType];
00077     return ostr.str();
00078 }
00079
00080 // //////////////////////////////////////
00081 std::string EventType::describeLabels() {
00082     std::ostringstream ostr;
00083     for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
00084         if (idx != 0) {
00085             ostr << ", ";
00086         }
00087         ostr << _labels[idx];
00088     }
00089     return ostr.str();
00090 }
00091
00092 // //////////////////////////////////////
00093 EventType::EN_EventType EventType::getType() const {
00094     return _type;
00095 }
00096
00097 // //////////////////////////////////////
00098 std::string EventType::getTypeAsString() const {
00099     std::ostringstream ostr;
00100     ostr << _typeLabels[_type];
00101     return ostr.str();
00102 }
00103
00104 // //////////////////////////////////////
00105 const std::string EventType::describe() const {
00106     std::ostringstream ostr;
00107     ostr << _labels[_type];
00108     return ostr.str();
00109 }
00110
00111 // //////////////////////////////////////
00112 bool EventType::operator==(const EN_EventType& iType) const {
00113     return (_type == iType);
00114 }
00115
00116 }

```

35.73 stdair/basic/EventType.hpp File Reference

```
#include <string>
```

```
#include <stdair/basic/StructAbstract.hpp>
```

Classes

- struct [stdair::EventType](#)

Namespaces

- namespace [stdair](#)

Handle on the StdAir library context.

35.74 EventType.hpp

```

00001 #ifndef __STDAIR_BAS_EVENTTYPE_HPP
00002 #define __STDAIR_BAS_EVENTTYPE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/basic/StructAbstract.hpp>
00011
00012 namespace stdair {
00013
00014     struct EventType : public StructAbstract {
00015     public:
00016         typedef enum {
00017             BKG_REQ = 0,
00018             CX,
00019             OPT_NOT_4_FD,
00020             OPT_NOT_4_NET,
00021             SKD_CHG,
00022             SNAPSHOT,
00023             RM,
00024             BRK_PT,
00025             LAST_VALUE
00026         } EN_EventType;
00027
00028         static const std::string& getLabel (const EN_EventType&);
00029
00030         static char getTypeLabel (const EN_EventType&);
00031
00032         static std::string getTypeLabelAsString (const EN_EventType&);
00033
00034         static std::string describeLabels();
00035
00036         EN_EventType getType() const;
00037
00038         std::string getTypeAsString() const;
00039
00040         const std::string describe() const;
00041
00042     public:
00043         bool operator== (const EN_EventType&) const;
00044
00045     public:
00046         EventType (const EN_EventType&);
00047         EventType (const char iType);
00048         EventType (const EventType&);
00049
00050     private:
00051         EventType();
00052
00053     private:
00054         static const std::string _labels[LAST_VALUE];
00055         static const char _typeLabels[LAST_VALUE];

```

```

00074
00075
00076     private:
00077         // ////////// Attributes //////////
00079         EN_EventType _type;
00080     };
00081
00082 }
00083 #endif // __STDAIR_BAS_EVENTTYPE_HPP

```

35.75 stdair/basic/float_utils.hpp File Reference

```
#include <stdair/basic/float_utils_google.hpp>
```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.76 float_utils.hpp

```

00001 #ifndef __STDAIR_BAS_FLOAT_UTILS_HPP
00002 #define __STDAIR_BAS_FLOAT_UTILS_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/basic/float_utils_google.hpp>
00009
00010 namespace stdair {
00011
00023 }
00024 #endif // __STDAIR_BAS_FLOAT_UTILS_HPP

```

35.77 stdair/basic/float_utils_google.hpp File Reference

Classes

- class [TypeWithSize< size >](#)
- class [TypeWithSize< 4 >](#)
- class [TypeWithSize< 8 >](#)
- class [FloatingPoint< RawType >](#)

35.78 float_utils_google.hpp

```

00001 #ifndef __STDAIR_BAS_FLOAT_UTILS_GOOGLE_HPP
00002 #define __STDAIR_BAS_FLOAT_UTILS_GOOGLE_HPP
00003
00004 // Redistribution and use in source and binary forms, with or without

```

```
00005 // modification, are permitted provided that the following conditions are
00006 // met:
00007 //
00008 //      * Redistributions of source code must retain the above copyright
00009 // notice, this list of conditions and the following disclaimer.
00010 //      * Redistributions in binary form must reproduce the above
00011 // copyright notice, this list of conditions and the following disclaimer
00012 // in the documentation and/or other materials provided with the
00013 // distribution.
00014 //      * Neither the name of Google Inc. nor the names of its
00015 // contributors may be used to endorse or promote products derived from
00016 // this software without specific prior written permission.
00017 //
00018 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00019 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00020 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00021 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
00022 // OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00023 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00024 // LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00025 // DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00026 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00027 // (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00028 // OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00029 //
00030 // Authors: wan@google.com (Zhanyong Wan), eefacm@gmail.com (Sean McAfee)
00031 //
00032 // The Google C++ Testing Framework (Google Test)
00033
00034
00035 // This template class serves as a compile-time function from size to
00036 // type. It maps a size in bytes to a primitive type with that
00037 // size. e.g.
00038 //
00039 //     TypeWithSize<4>::UInt
00040 //
00041 // is typedef-ed to be unsigned int (unsigned integer made up of 4
00042 // bytes).
00043 //
00044 // Such functionality should belong to STL, but I cannot find it
00045 // there.
00046 //
00047 // Google Test uses this class in the implementation of floating-point
00048 // comparison.
00049 //
00050 // For now it only handles UInt (unsigned int) as that's all Google Test
00051 // needs. Other types can be easily added in the future if need
00052 // arises.
00053 template <size_t size>
00054 class TypeWithSize {
00055 public:
00056     // This prevents the user from using TypeWithSize<N> with incorrect
00057     // values of N.
00058     typedef void UInt;
00059 };
00060
00061 // The specialization for size 4.
00062 template <>
00063 class TypeWithSize<4> {
00064 public:
00065     // unsigned int has size 4 in both gcc and MSVC.
00066     //
```

```

00067 // As base/basicTypes.h doesn't compile on Windows, we cannot use
00068 // uint32, uint64, and etc here.
00069 typedef int Int;
00070 typedef unsigned int UInt;
00071 };
00072
00073 // The specialization for size 8.
00074 template <>
00075 class TypeWithSize<8> {
00076 public:
00077 #if GTEST_OS_WINDOWS
00078     typedef __int64 Int;
00079     typedef unsigned __int64 UInt;
00080 #else
00081     typedef long long Int; // NOLINT
00082     typedef unsigned long long UInt; // NOLINT
00083 #endif // GTEST_OS_WINDOWS
00084 };
00085
00086
00087 // This template class represents an IEEE floating-point number
00088 // (either single-precision or double-precision, depending on the
00089 // template parameters).
00090 //
00091 // The purpose of this class is to do more sophisticated number
00092 // comparison. (Due to round-off error, etc, it's very unlikely that
00093 // two floating-points will be equal exactly. Hence a naive
00094 // comparison by the == operation often doesn't work.)
00095 //
00096 // Format of IEEE floating-point:
00097 //
00098 //   The most-significant bit being the leftmost, an IEEE
00099 //   floating-point looks like
00100 //
00101 //       sign_bit exponent_bits fraction_bits
00102 //
00103 //   Here, sign_bit is a single bit that designates the sign of the
00104 //   number.
00105 //
00106 //   For float, there are 8 exponent bits and 23 fraction bits.
00107 //
00108 //   For double, there are 11 exponent bits and 52 fraction bits.
00109 //
00110 //   More details can be found at
00111 //   http://en.wikipedia.org/wiki/IEEE\_floating-point\_standard.
00112 //
00113 // Template parameter:
00114 //
00115 //   RawType: the raw floating-point type (either float or double)
00116 template <typename RawType>
00117 class FloatingPoint {
00118 public:
00119     // Defines the unsigned integer type that has the same size as the
00120     // floating point number.
00121     typedef typename TypeWithSize<sizeof(RawType)>::UInt Bits;
00122
00123     // Constants.
00124
00125     // # of bits in a number.
00126     static const size_t kBitCount = 8*sizeof(RawType);
00127
00128     // # of fraction bits in a number.

```

```

00129 static const size_t kFractionBitCount =
00130     std::numeric_limits<RawType>::digits - 1;
00131
00132 // # of exponent bits in a number.
00133 static const size_t kExponentBitCount = kBitCount - 1 - kFractionBitCount;
00134
00135 // The mask for the sign bit.
00136 static const Bits kSignBitMask = static_cast<Bits>(1) << (kBitCount - 1);
00137
00138 // The mask for the fraction bits.
00139 static const Bits kFractionBitMask =
00140     ~static_cast<Bits>(0) >> (kExponentBitCount + 1);
00141
00142 // The mask for the exponent bits.
00143 static const Bits kExponentBitMask = ~(kSignBitMask | kFractionBitMask);
00144
00145 // How many ULP's (Units in the Last Place) we want to tolerate when
00146 // comparing two numbers. The larger the value, the more error we
00147 // allow. A 0 value means that two numbers must be exactly the same
00148 // to be considered equal.
00149 //
00150 // The maximum error of a single floating-point operation is 0.5
00151 // units in the last place. On Intel CPU's, all floating-point
00152 // calculations are done with 80-bit precision, while double has 64
00153 // bits. Therefore, 4 should be enough for ordinary use.
00154 //
00155 // See the following article for more details on ULP:
00156 // http://www.cygnum-software.com/papers/comparingfloats/comparingfloats.htm.
00157 static const size_t kMaxUlp = 4;
00158
00159 // Constructs a FloatingPoint from a raw floating-point number.
00160 //
00161 // On an Intel CPU, passing a non-normalized NAN (Not a Number)
00162 // around may change its bits, although the new value is guaranteed
00163 // to be also a NAN. Therefore, don't expect this constructor to
00164 // preserve the bits in x when x is a NAN.
00165 explicit FloatingPoint(const RawType& x) { u_.value_ = x; }
00166
00167 // Static methods
00168
00169 // Reinterprets a bit pattern as a floating-point number.
00170 //
00171 // This function is needed to test the AlmostEquals() method.
00172 static RawType ReinterpretBits(const Bits bits) {
00173     FloatingPoint fp(0);
00174     fp.u_.bits_ = bits;
00175     return fp.u_.value_;
00176 }
00177
00178 // Returns the floating-point number that represent positive infinity.
00179 static RawType Infinity() {
00180     return ReinterpretBits(kExponentBitMask);
00181 }
00182
00183 // Non-static methods
00184
00185 // Returns the bits that represents this number.
00186 const Bits &bits() const { return u_.bits_; }
00187
00188 // Returns the exponent bits of this number.
00189 Bits exponent_bits() const { return kExponentBitMask & u_.bits_; }
00190

```

```

00191 // Returns the fraction bits of this number.
00192 Bits fraction_bits() const { return kFractionBitMask & u_.bits_; }
00193
00194 // Returns the sign bit of this number.
00195 Bits sign_bit() const { return kSignBitMask & u_.bits_; }
00196
00197 // Returns true iff this is NAN (not a number).
00198 bool is_nan() const {
00199     // It's a NAN if the exponent bits are all ones and the fraction
00200     // bits are not entirely zeros.
00201     return (exponent_bits() == kExponentBitMask) && (fraction_bits() != 0);
00202 }
00203
00204 // Returns true iff this number is at most kMaxUlp's ULP's away from
00205 // rhs. In particular, this function:
00206 //
00207 // - returns false if either number is (or both are) NAN.
00208 // - treats really large numbers as almost equal to infinity.
00209 // - thinks +0.0 and -0.0 are 0 DLP's apart.
00210 bool AlmostEquals(const FloatingPoint& rhs) const {
00211     // The IEEE standard says that any comparison operation involving
00212     // a NAN must return false.
00213     if (is_nan() || rhs.is_nan()) return false;
00214
00215     return DistanceBetweenSignAndMagnitudeNumbers(u_.bits_, rhs.u_.bits_)
00216         <= kMaxUlp's;
00217 }
00218
00219 private:
00220 // The data type used to store the actual floating-point number.
00221 union FloatingPointUnion {
00222     RawType value_; // The raw floating-point number.
00223     Bits bits_; // The bits that represent the number.
00224 };
00225
00226 // Converts an integer from the sign-and-magnitude representation to
00227 // the biased representation. More precisely, let N be 2 to the
00228 // power of (kBitCount - 1), an integer x is represented by the
00229 // unsigned number x + N.
00230 //
00231 // For instance,
00232 //
00233 // -N + 1 (the most negative number representable using
00234 // sign-and-magnitude) is represented by 1;
00235 // 0 is represented by N; and
00236 // N - 1 (the biggest number representable using
00237 // sign-and-magnitude) is represented by 2N - 1.
00238 //
00239 // Read http://en.wikipedia.org/wiki/Signed\_number\_representations
00240 // for more details on signed number representations.
00241 static Bits SignAndMagnitudeToBiased(const Bits &sam) {
00242     if (kSignBitMask & sam) {
00243         // sam represents a negative number.
00244         return ~sam + 1;
00245     } else {
00246         // sam represents a positive number.
00247         return kSignBitMask | sam;
00248     }
00249 }
00250
00251 // Given two numbers in the sign-and-magnitude representation,
00252 // returns the distance between them as an unsigned number.

```

```

00253     static Bits DistanceBetweenSignAndMagnitudeNumbers(const Bits &sam1,
00254                                                         const Bits &sam2) {
00255         const Bits biased1 = SignAndMagnitudeToBiased(sam1);
00256         const Bits biased2 = SignAndMagnitudeToBiased(sam2);
00257         return (biased1 >= biased2) ? (biased1 - biased2) : (biased2 - biased1);
00258     }
00259
00260     FloatingPointUnion u_;
00261 };
00262
00263 #endif // __STDAIR_BAS_FLOAT_UTILS_GOOGLE_HPP

```

35.79 stdair/basic/ForecastingMethod.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/ForecastingMethod.hpp>

```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

35.80 ForecastingMethod.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/stdair_exceptions.hpp>
00009 #include <stdair/basic/ForecastingMethod.hpp>
00010
00011 namespace stdair {
00012
00013     // //////////////////////////////////////
00014     const std::string ForecastingMethod::_labels[LAST_VALUE] =
00015         { "AdditivePickUp", "MultiplicativePickUp" };
00016
00017     // //////////////////////////////////////
00018     const char ForecastingMethod::_methodLabels[LAST_VALUE] = { 'A', 'M' };
00019
00020
00021
00022     // //////////////////////////////////////
00023     ForecastingMethod::ForecastingMethod()
00024         : _method (LAST_VALUE) {
00025         assert (false);
00026     }
00027

```

```

00028 // //////////////////////////////////////
00029 ForecastingMethod::
00030 ForecastingMethod (const ForecastingMethod& iForecastingMethod)
00031 : _method (iForecastingMethod._method) {
00032 }
00033
00034 // //////////////////////////////////////
00035 ForecastingMethod::
00036 ForecastingMethod (const EN_ForecastingMethod& iForecastingMethod)
00037 : _method (iForecastingMethod) {
00038 }
00039
00040 // //////////////////////////////////////
00041 ForecastingMethod::ForecastingMethod (const char iMethod) {
00042     switch (iMethod) {
00043         case 'A': _method = ADD_PK; break;
00044         case 'M': _method = MUL_PK; break;
00045         default: _method = LAST_VALUE; break;
00046     }
00047
00048     if (_method == LAST_VALUE) {
00049         const std::string& lLabels = describeLabels();
00050         std::ostringstream oMessage;
00051         oMessage << "The forecasting method '" << iMethod
00052             << "' is not known. Known forecasting methods: " << lLabels;
00053         throw CodeConversionException (oMessage.str());
00054     }
00055 }
00056
00057 // //////////////////////////////////////
00058 const std::string& ForecastingMethod::
00059 getLabel (const EN_ForecastingMethod& iMethod) {
00060     return _labels[iMethod];
00061 }
00062
00063 // //////////////////////////////////////
00064 char ForecastingMethod::getMethodLabel (const EN_ForecastingMethod& iMethod) {
00065     return _methodLabels[iMethod];
00066 }
00067
00068 // //////////////////////////////////////
00069 std::string ForecastingMethod::
00070 getMethodLabelAsString (const EN_ForecastingMethod& iMethod) {
00071     std::ostringstream oStr;
00072     oStr << _methodLabels[iMethod];
00073     return oStr.str();
00074 }
00075
00076 // //////////////////////////////////////
00077 std::string ForecastingMethod::describeLabels() {
00078     std::ostringstream ostr;
00079     for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
00080         if (idx != 0) {
00081             ostr << ", ";
00082         }
00083         ostr << _labels[idx];
00084     }
00085     return ostr.str();
00086 }
00087
00088 // //////////////////////////////////////
00089 ForecastingMethod::EN_ForecastingMethod ForecastingMethod::getMethod() const {

```

```

00090     return _method;
00091 }
00092
00093 // //////////////////////////////////////
00094 std::string ForecastingMethod::getMethodAsString() const {
00095     std::ostringstream ostr;
00096     ostr << _methodLabels[_method];
00097     return ostr.str();
00098 }
00099
00100 // //////////////////////////////////////
00101 const std::string ForecastingMethod::describe() const {
00102     std::ostringstream ostr;
00103     ostr << _labels[_method];
00104     return ostr.str();
00105 }
00106
00107 // //////////////////////////////////////
00108 bool ForecastingMethod::
00109 operator== (const EN_ForecastingMethod& iMethod) const {
00110     return (_method == iMethod);
00111 }
00112
00113 }

```

35.81 stdair/basic/ForecastingMethod.hpp File Reference

```

#include <string>
#include <stdair/basic/StructAbstract.hpp>

```

Classes

- struct [stdair::ForecastingMethod](#)

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.82 ForecastingMethod.hpp

```

00001 #ifndef __STDAIR_BAS_FORECASTINGMETHOD_HPP
00002 #define __STDAIR_BAS_FORECASTINGMETHOD_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/basic/StructAbstract.hpp>
00011
00012 namespace stdair {

```

```

00013
00015 struct ForecastingMethod : public StructAbstract {
00016 public:
00017     typedef enum {
00018         ADD_PK = 0,
00019         MUL_PK,
00020         LAST_VALUE
00021     } EN_ForecastingMethod;
00022
00025     static const std::string& getLabel (const EN_ForecastingMethod&);
00026
00028     static char getMethodLabel (const EN_ForecastingMethod&);
00029
00031     static std::string getMethodLabelAsString (const EN_ForecastingMethod&);
00032
00034     static std::string describeLabels();
00035
00037     EN_ForecastingMethod getMethod() const;
00038
00040     std::string getMethodAsString() const;
00041
00044     const std::string describe() const;
00045
00046 public:
00048     bool operator== (const EN_ForecastingMethod&) const;
00049
00050 public:
00052     ForecastingMethod (const EN_ForecastingMethod&);
00054     ForecastingMethod (const char iMethod);
00056     ForecastingMethod (const ForecastingMethod&);
00057
00058 private:
00060     ForecastingMethod();
00061
00062
00063 private:
00065     static const std::string _labels[LAST_VALUE];
00067     static const char _methodLabels[LAST_VALUE];
00068
00069
00070 private:
00071     // ////////// Attributes //////////
00073     EN_ForecastingMethod _method;
00074 };
00075
00076 }
00077 #endif // __STDAIR_BAS_FORECASTINGMETHOD_HPP

```

35.83 stdair/basic/PartnershipTechnique.cpp File Reference

```

#include <cassert>

#include <sstream>

#include <stdair/stdair_exceptions.hpp>

#include <stdair/basic/PartnershipTechnique.hpp>

```

Namespaces

- namespace `stdair`

Handle on the StdAir library context.

35.84 PartnershipTechnique.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/stdair_exceptions.hpp>
00009 #include <stdair/basic/PartnershipTechnique.hpp>
00010
00011 namespace stdair {
00012
00013 // //////////////////////////////////////
00014 const std::string PartnershipTechnique::_labels[LAST_VALUE] =
00015     { "None",
00016       "RevenueAvailabilityExchangeDemandAggregation",
00017       "RevenueAvailabilityExchangeYieldProration",
00018       "InterlineBidPriceDemandAggregation",
00019       "InterlineBidPriceYieldProration",
00020       "NonProtectionistInterlineBidPriceYieldProration",
00021       "RevenueManagementCooperation",
00022       "AdvancedRevenueManagementCooperation"};
00023
00024 // //////////////////////////////////////
00025 const char PartnershipTechnique::_techniqueLabels[LAST_VALUE] = { 'N',
00026                                                                      'r',
00027                                                                      'R',
00028                                                                      'i',
00029                                                                      'I',
00030                                                                      'U',
00031                                                                      'C',
00032                                                                      'A'};
00033
00034 // //////////////////////////////////////
00035 PartnershipTechnique::PartnershipTechnique() : _technique (LAST_VALUE) {
00036     assert (false);
00037 }
00038
00039 // //////////////////////////////////////
00040 PartnershipTechnique::
00041 PartnershipTechnique (const PartnershipTechnique& iPartnershipTechnique)
00042     : _technique (iPartnershipTechnique._technique) {
00043 }
00044
00045 // //////////////////////////////////////
00046 PartnershipTechnique::
00047 PartnershipTechnique (const EN_PartnershipTechnique& iPartnershipTechnique)
00048     : _technique (iPartnershipTechnique) {
00049 }
00050
00051 // //////////////////////////////////////
00052 PartnershipTechnique::EN_PartnershipTechnique

```

```

00054 PartnershipTechnique::getTechnique (const char iTechniqueChar) {
00055     EN_PartnershipTechnique oTechnique;
00056     switch (iTechniqueChar) {
00057         case 'N': oTechnique = NONE; break;
00058         case 'r': oTechnique = RAE_DA; break;
00059         case 'R': oTechnique = RAE_YP; break;
00060         case 'i': oTechnique = IBP_DA; break;
00061         case 'I': oTechnique = IBP_YP; break;
00062         case 'U': oTechnique = IBP_YP_U; break;
00063         case 'C': oTechnique = RMC; break;
00064         case 'A': oTechnique = A_RMC; break;
00065         default: oTechnique = LAST_VALUE; break;
00066     }
00067
00068     if (oTechnique == LAST_VALUE) {
00069         const std::string& lLabels = describeLabels();
00070         std::ostringstream oMessage;
00071         oMessage << "The partnership technique '"
00072                 << iTechniqueChar
00073                 << "' is not known. Known partnership techniques: "
00074                 << lLabels;
00075         throw CodeConversionException (oMessage.str());
00076     }
00077
00078     return oTechnique;
00079 }
00080
00081 // //////////////////////////////////////
00082 PartnershipTechnique::PartnershipTechnique (const char iTechniqueChar)
00083 : _technique (getTechnique (iTechniqueChar)) {
00084 }
00085
00086 // //////////////////////////////////////
00087 PartnershipTechnique::
00088 PartnershipTechnique (const std::string& iTechniqueStr) {
00089     //
00090     const size_t lSize = iTechniqueStr.size();
00091     assert (lSize == 1);
00092     const char lTechniqueChar = iTechniqueStr[0];
00093     _technique = getTechnique (lTechniqueChar);
00094 }
00095
00096 // //////////////////////////////////////
00097 const std::string& PartnershipTechnique::
00098 getLabel (const EN_PartnershipTechnique& iTechnique) {
00099     return _labels[iTechnique];
00100 }
00101
00102 // //////////////////////////////////////
00103 char PartnershipTechnique::
00104 getTechniqueLabel (const EN_PartnershipTechnique& iTechnique) {
00105     return _techniqueLabels[iTechnique];
00106 }
00107
00108 // //////////////////////////////////////
00109 std::string PartnershipTechnique::
00110 getTechniqueLabelAsString (const EN_PartnershipTechnique& iTechnique) {
00111     std::ostringstream oStr;
00112     oStr << _techniqueLabels[iTechnique];
00113     return oStr.str();
00114 }
00115

```

```

00116 ///////////////////////////////////////////////////////////////////
00117 std::string PartnershipTechnique::describeLabels() {
00118     std::ostringstream ostr;
00119     for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
00120         if (idx != 0) {
00121             ostr << ", ";
00122         }
00123         ostr << _labels[idx];
00124     }
00125     return ostr.str();
00126 }
00127
00128 ///////////////////////////////////////////////////////////////////
00129 PartnershipTechnique::EN_PartnershipTechnique
00130 PartnershipTechnique::getTechnique() const {
00131     return _technique;
00132 }
00133
00134 ///////////////////////////////////////////////////////////////////
00135 char PartnershipTechnique::getTechniqueAsChar() const {
00136     const char oTechniqueChar = _techniqueLabels[_technique];
00137     return oTechniqueChar;
00138 }
00139
00140 ///////////////////////////////////////////////////////////////////
00141 std::string PartnershipTechnique::getTechniqueAsString() const {
00142     std::ostringstream ostr;
00143     ostr << _techniqueLabels[_technique];
00144     return ostr.str();
00145 }
00146
00147 ///////////////////////////////////////////////////////////////////
00148 const std::string PartnershipTechnique::describe() const {
00149     std::ostringstream ostr;
00150     ostr << _labels[_technique];
00151     return ostr.str();
00152 }
00153
00154 ///////////////////////////////////////////////////////////////////
00155 bool PartnershipTechnique::
00156 operator== (const EN_PartnershipTechnique& iTechnique) const {
00157     return (_technique == iTechnique);
00158 }
00159
00160 }

```

35.85 stdair/basic/PartnershipTechnique.hpp File Reference

```
#include <string>
```

```
#include <stdair/basic/StructAbstract.hpp>
```

Classes

- struct [stdair::PartnershipTechnique](#)
Enumeration of partnership techniques.

Namespaces

- namespace `stdair`

Handle on the StdAir library context.

35.86 PartnershipTechnique.hpp

```

00001 #ifndef __STDAIR_BAS_PARTNERSHIPTECHNIQUE_HPP
00002 #define __STDAIR_BAS_PARTNERSHIPTECHNIQUE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/basic/StructAbstract.hpp>
00011
00012 namespace stdair {
00013
00017     struct PartnershipTechnique : public StructAbstract {
00018     public:
00019         typedef enum {
00020             NONE = 0,
00021             RAE_DA,
00022             RAE_YP,
00023             IBP_DA,
00024             IBP_YP,
00025             IBP_YP_U,
00026             RMC,
00027             A_RMC,
00028             LAST_VALUE
00029         } EN_PartnershipTechnique;
00030
00034         static const std::string& getLabel (const EN_PartnershipTechnique&);
00035
00039         static EN_PartnershipTechnique getTechnique (const char);
00040
00044         static char getTechniqueLabel (const EN_PartnershipTechnique&);
00045
00049         static std::string getTechniqueLabelAsString (const EN_PartnershipTechnique&);
00050     };
00054     static std::string describeLabels();
00055
00059     EN_PartnershipTechnique getTechnique() const;
00060
00064     char getTechniqueAsChar() const;
00065
00069     std::string getTechniqueAsString() const;
00070
00075     const std::string describe() const;
00076
00077     public:
00081         bool operator== (const EN_PartnershipTechnique&) const;
00082
00083     public:
00087         PartnershipTechnique (const EN_PartnershipTechnique&);
00091         PartnershipTechnique (const char iTechnique);
00095         PartnershipTechnique (const std::string& iTechnique);

```

```

00099     PartnershipTechnique (const PartnershipTechnique&);
00100
00101 private:
00102     PartnershipTechnique();
00103
00104 private:
00105     static const std::string _labels[LAST_VALUE];
00106     static const char _techniqueLabels[LAST_VALUE];
00107
00108 private:
00109     // ////////// Attributes //////////
00110     EN_PartnershipTechnique _technique;
00111 };
00112
00113 }
00114
00115 #endif // __STDAIR_BASIC_PARTNERSHIPTECHNIQUE_HPP

```

35.87 stdair/basic/PassengerType.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/PassengerType.hpp>

```

Namespaces

- namespace `stdair`
Handle on the *StdAir* library context.

35.88 PassengerType.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/stdair_exceptions.hpp>
00009 #include <stdair/basic/PassengerType.hpp>
00010
00011 namespace stdair {
00012
00013     // //////////////////////////////////////
00014     const std::string PassengerType::_labels[LAST_VALUE] =
00015         { "Leisure", "Business", "First" };
00016
00017     const char PassengerType::_typeLabels[LAST_VALUE] = { 'L', 'B', 'F' };
00018
00019
00020     // //////////////////////////////////////
00021     PassengerType::PassengerType (const EN_PassengerType& iPassengerType)

```

```

00022     : _type (iPassengerType) {
00023     }
00024
00025     // //////////////////////////////////////
00026     PassengerType::PassengerType (const char iType) {
00027         switch (iType) {
00028             case 'L': _type = LEISURE; break;
00029             case 'B': _type = BUSINESS; break;
00030             case 'F': _type = FIRST; break;
00031             default: _type = LAST_VALUE; break;
00032         }
00033
00034         if (_type == LAST_VALUE) {
00035             const std::string& lLabels = describeLabels();
00036             std::ostringstream oMessage;
00037             oMessage << "The passenger type '" << iType
00038                 << "' is not known. Known passenger types: " << lLabels;
00039             throw CodeConversionException (oMessage.str());
00040         }
00041     }
00042
00043     // //////////////////////////////////////
00044     const std::string& PassengerType::getLabel (const EN_PassengerType& iType) {
00045         return _labels[iType];
00046     }
00047
00048     // //////////////////////////////////////
00049     char PassengerType::getTypeLabel (const EN_PassengerType& iType) {
00050         return _typeLabels[iType];
00051     }
00052
00053     // //////////////////////////////////////
00054     std::string PassengerType::
00055     getTypeLabelAsString (const EN_PassengerType& iType) {
00056         std::ostringstream oStr;
00057         oStr << _typeLabels[iType];
00058         return oStr.str();
00059     }
00060
00061     // //////////////////////////////////////
00062     std::string PassengerType::describeLabels() {
00063         std::ostringstream ostr;
00064         for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
00065             if (idx != 0) {
00066                 ostr << ", ";
00067             }
00068             ostr << _labels[idx];
00069         }
00070         return ostr.str();
00071     }
00072
00073     // //////////////////////////////////////
00074     PassengerType::EN_PassengerType PassengerType::getType() const {
00075         return _type;
00076     }
00077
00078     // //////////////////////////////////////
00079     std::string PassengerType::getTypeAsString() const {
00080         std::ostringstream oStr;
00081         oStr << _typeLabels[_type];
00082         return oStr.str();
00083     }

```

```

00084
00085 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00086 const std::string PassengerType::describe() const {
00087     std::ostringstream ostr;
00088     ostr << _labels[_type];
00089     return ostr.str();
00090 }
00091
00092 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00093 bool PassengerType::operator== (const EN_PassengerType& iType) const {
00094     return (_type == iType);
00095 }
00096
00097 }

```

35.89 stdair/basic/PassengerType.hpp File Reference

```
#include <string>
```

```
#include <stdair/basic/StructAbstract.hpp>
```

Classes

- struct [stdair::PassengerType](#)

Namespaces

- namespace [stdair](#)

Handle on the StdAir library context.

35.90 PassengerType.hpp

```

00001 #ifndef __STDAIR_BAS_PASSENGERTYPE_HPP
00002 #define __STDAIR_BAS_PASSENGERTYPE_HPP
00003
00004 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00005 // Import section
00006 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/basic/StructAbstract.hpp>
00011
00012 namespace stdair {
00013
00015     struct PassengerType : public StructAbstract {
00016     public:
00017         typedef enum {
00018             LEISURE = 0,
00019             BUSINESS,
00020             FIRST,
00021             LAST_VALUE
00022         } EN_PassengerType;
00023

```

```

00025     static const std::string& getLabel (const EN_PassengerType&);
00026
00028     static char getTypeLabel (const EN_PassengerType&);
00029
00031     static std::string getTypeLabelAsString (const EN_PassengerType&);
00032
00034     static std::string describeLabels();
00035
00037     EN_PassengerType getType() const;
00038
00040     std::string getTypeAsString() const;
00041
00043     const std::string describe() const;
00044
00045 public:
00047     bool operator== (const EN_PassengerType&) const;
00048
00049 public:
00051     PassengerType (const EN_PassengerType&);
00053     PassengerType (const char iType);
00054
00055 private:
00056     static const std::string _labels[LAST_VALUE];
00058     static const char _typeLabels[LAST_VALUE];
00061
00062 private:
00063     // ////////// Attributes //////////
00064     EN_PassengerType _type;
00067 };
00068
00069 }
00070 #endif // __STDAIR_BAS_PASSENGERTYPE_HPP

```

35.91 stdair/basic/ProgressStatus.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/BasConst_Event.hpp>
#include <stdair/basic/ProgressStatus.hpp>

```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

35.92 ProgressStatus.cpp

```

00001 // //////////////////////////////////////
00002 // Import section

```

```

00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/stdair_exceptions.hpp>
00009 #include <stdair/basic/BasConst_Event.hpp>
00010 #include <stdair/basic/ProgressStatus.hpp>
00011
00012 namespace stdair {
00013
00014 // //////////////////////////////////////
00015 ProgressStatus::ProgressStatus (const Count_T& iCurrentNb,
00016                               const Count_T& iExpectedNb,
00017                               const Count_T& iActualNb)
00018 : _currentNb (iCurrentNb),
00019   _expectedNb (iExpectedNb), _actualNb (iActualNb) {
00020 }
00021
00022 // //////////////////////////////////////
00023 ProgressStatus::ProgressStatus (const Count_T& iExpectedNb,
00024                               const Count_T& iActualNb)
00025 : _currentNb (DEFAULT_PROGRESS_STATUS),
00026   _expectedNb (iExpectedNb), _actualNb (iActualNb) {
00027 }
00028
00029 // //////////////////////////////////////
00030 ProgressStatus::ProgressStatus (const Count_T& iActualNb)
00031 : _currentNb (DEFAULT_PROGRESS_STATUS),
00032   _expectedNb (iActualNb), _actualNb (DEFAULT_PROGRESS_STATUS) {
00033 }
00034
00035 // //////////////////////////////////////
00036 ProgressStatus::ProgressStatus ()
00037 : _currentNb (DEFAULT_PROGRESS_STATUS),
00038   _expectedNb (DEFAULT_PROGRESS_STATUS),
00039   _actualNb (DEFAULT_PROGRESS_STATUS) {
00040 }
00041
00042 // //////////////////////////////////////
00043 ProgressStatus::ProgressStatus (const ProgressStatus& iProgressStatus)
00044 : _currentNb (iProgressStatus._currentNb),
00045   _expectedNb (iProgressStatus._expectedNb),
00046   _actualNb (iProgressStatus._actualNb) {
00047 }
00048
00049 // //////////////////////////////////////
00050 void ProgressStatus::reset () {
00051   _currentNb = DEFAULT_PROGRESS_STATUS;
00052   _actualNb = DEFAULT_PROGRESS_STATUS;
00053 }
00054
00055 // //////////////////////////////////////
00056 const std::string ProgressStatus::describe() const {
00057   std::ostringstream oStr;
00058   oStr << _currentNb << " / {" << _expectedNb << ", " << _actualNb << "}";
00059   return oStr.str();
00060 }
00061
00062 }

```

35.93 stdair/basic/ProgressStatus.hpp File Reference

```
#include <string>
#include <boost/progress.hpp>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
```

Classes

- struct [stdair::ProgressStatus](#)

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.94 ProgressStatus.hpp

```
00001 #ifndef __STDAIR_BAS_PROGRESSSTATUS_HPP
00002 #define __STDAIR_BAS_PROGRESSSTATUS_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // Boost Progress
00010 #include <boost/progress.hpp>
00011 // StdAir
00012 #include <stdair/stdair_basic_types.hpp>
00013 #include <stdair/basic/StructAbstract.hpp>
00014
00015 namespace stdair {
00016
00025     struct ProgressStatus : public StructAbstract {
00026     public:
00027         // ////////////////////////////////////// Getters //////////////////////////////////////
00029         const Count_T& count() const {
00030             return _currentNb;
00031         }
00032
00034         const Count_T& getCurrentNb() const {
00035             return _currentNb;
00036         }
00037
00039         const Count_T& getExpectedNb() const {
00040             return _expectedNb;
00041         }
00042
00044         const Count_T& getActualNb() const {
00045             return _actualNb;
00046         }
00047
00048     }
```

```

00049     const ProgressPercentage_T progress() const {
00050         if (_actualNb == 0) {
00051             return 100.0;
00052         }
00053         return (static_cast<Percentage_T> (_currentNb)
00054             / static_cast<Percentage_T> (_actualNb));
00055     }
00056
00057
00058     // ////////////////////////////////// Setters //////////////////////////////////
00060     void setCurrentNb (const Count_T& iCurrentNb) {
00061         _currentNb = iCurrentNb;
00062     }
00063
00065     void setExpectedNb (const Count_T& iExpectedNb) {
00066         _expectedNb = iExpectedNb;
00067     }
00068
00070     void setActualNb (const Count_T& iActualNb) {
00071         _actualNb = iActualNb;
00072     }
00073
00075     void reset();
00076
00078     Count_T operator+= (Count_T iIncrement) {
00079         _currentNb += iIncrement;
00080         return _currentNb;
00081     }
00082
00084     Count_T operator++() {
00085         ++_currentNb;
00086         return _currentNb;
00087     }
00088
00089
00090     public:
00091         // ////////////////////////////////// Display Support Methods //////////////////////////////////
00093         const std::string describe() const;
00094
00095
00096     public:
00104         ProgressStatus (const Count_T& iCurrentNb, const Count_T& iExpectedNb,
00105             const Count_T& iActualNb);
00106
00115         ProgressStatus (const Count_T& iExpectedNb, const Count_T& iActualNb);
00116
00125         ProgressStatus (const Count_T& iActualNb);
00126
00132         ProgressStatus();
00133
00137         ProgressStatus (const ProgressStatus&);
00138
00139     private:
00140         // ////////////////////////////////// Attributes //////////////////////////////////
00142         Count_T _currentNb;
00143
00145         Count_T _expectedNb;
00146
00148         Count_T _actualNb;
00149     };
00150
00151 }

```

```
00152 #endif // __STDAIR_BAS_PROGRESSSTATUS_HPP
```

35.95 stdair/basic/ProgressStatusSet.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/ProgressStatusSet.hpp>
```

Namespaces

- namespace `stdair`
Handle on the `StdAir` library context.

35.96 ProgressStatusSet.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/ProgressStatusSet.hpp>
00009
00010 namespace stdair {
00011
00012 // //////////////////////////////////////
00013 ProgressStatusSet::ProgressStatusSet()
00014 : _eventType (EventType::LAST_VALUE), _typeSpecificProgressStatus(),
00015   _generatorProgressStatus(), _overallProgressStatus(), _generatorKey ("") {
00016     assert (false);
00017 }
00018
00019 // //////////////////////////////////////
00020 ProgressStatusSet::ProgressStatusSet (const EventType::EN_EventType& iType)
00021 : _eventType (iType), _typeSpecificProgressStatus(),
00022   _generatorProgressStatus(), _overallProgressStatus(), _generatorKey ("") {
00023 }
00024
00025 // //////////////////////////////////////
00026 ProgressStatusSet::
00027 ProgressStatusSet (const ProgressStatusSet& iProgressStatusSet)
00028 : _eventType (iProgressStatusSet._eventType),
00029   _typeSpecificProgressStatus(iProgressStatusSet._typeSpecificProgressStatu
00030 s),
00031   _generatorProgressStatus (iProgressStatusSet._generatorProgressStatus),
00032   _overallProgressStatus (iProgressStatusSet._overallProgressStatus),
00033   _generatorKey (iProgressStatusSet._generatorKey) {
00034 }
00035 // //////////////////////////////////////
00036 ProgressStatusSet::~ProgressStatusSet () {
00037 }
00038
```

```

00039 ///////////////////////////////////////////////////////////////////
00040 void ProgressStatusSet::fromStream (std::istream& ioIn) {
00041 }
00042
00043 ///////////////////////////////////////////////////////////////////
00044 const std::string ProgressStatusSet::describe() const {
00045     std::ostringstream oStr;
00046
00047     oStr << "-[Overall]"
00048         << "[" << _overallProgressStatus.getCurrentNb()
00049         << "/" << _overallProgressStatus.getExpectedNb()
00050         << "," << _overallProgressStatus.getActualNb()
00051         << "]" ";
00052
00053     oStr << "[" << EventType (_eventType) << "]"
00054         << "[" << _typeSpecificProgressStatus.getCurrentNb()
00055         << "/" << _typeSpecificProgressStatus.getExpectedNb()
00056         << "," << _typeSpecificProgressStatus.getActualNb()
00057         << "]" ";
00058
00059     oStr << " [Specific generator: " << _generatorKey << "]"
00060         << "[" << _generatorProgressStatus.getCurrentNb()
00061         << "/" << _generatorProgressStatus.getExpectedNb()
00062         << "," << _generatorProgressStatus.getActualNb()
00063         << "]" ";
00064
00065     return oStr.str();
00066 }
00067
00068 }

```

35.97 stdair/basic/ProgressStatusSet.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_event_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/basic/EventType.hpp>
#include <stdair/basic/ProgressStatus.hpp>

```

Classes

- struct [stdair::ProgressStatusSet](#)

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.98 ProgressStatusSet.hpp

```

00001 #ifndef __STDAIR_BAS_PROGRESSSTATUSSET_HPP
00002 #define __STDAIR_BAS_PROGRESSSTATUSSET_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_basic_types.hpp>
00012 #include <stdair/stdair_event_types.hpp>
00013 #include <stdair/basic/StructAbstract.hpp>
00014 #include <stdair/basic/EventType.hpp>
00015 #include <stdair/basic/ProgressStatus.hpp>
00016
00017 namespace stdair {
00018
00022     struct ProgressStatusSet : public StructAbstract {
00023         // ////////////////////////////////// Getters //////////////////////////////////
00031         const ProgressStatus& getTypeSpecificStatus() const {
00032             return _typeSpecificProgressStatus;
00033         }
00034
00043         const ProgressStatus& getSpecificGeneratorStatus() const {
00044             return _generatorProgressStatus;
00045         }
00046
00054         const ProgressStatus& getOverallStatus() const {
00055             return _overallProgressStatus;
00056         }
00057
00058
00059         // ////////////////////////////////// Setters //////////////////////////////////
00060     public:
00062         void setTypeSpecificStatus (const ProgressStatus& iProgressStatus) {
00063             _typeSpecificProgressStatus = iProgressStatus;
00064         }
00065
00068         void setSpecificGeneratorStatus (const ProgressStatus& iProgressStatus,
00069                                         const EventGeneratorKey_T& iKey) {
00070             _generatorProgressStatus = iProgressStatus;
00071             _generatorKey = iKey;
00072         }
00073
00076         void setOverallStatus (const ProgressStatus& iProgressStatus) {
00077             _overallProgressStatus = iProgressStatus;
00078         }
00079
00080
00081         // ////////////////////////////////// Display methods //////////////////////////////////
00082     public:
00085         void fromStream (std::istream& ioIn);
00086
00088         const std::string describe() const;
00089
00090
00091         // ////////////////////////////////// Constructors and destructors //////////////////////////////////
00092     public:
00094         ProgressStatusSet (const EventType::EN_EventType&);

```

```

00096     ProgressStatusSet (const ProgressStatusSet&);
00097     ~ProgressStatusSet ();
00098
00099 private:
00100     ProgressStatusSet ();
00101
00102     // ////////////////////////////////// Attributes //////////////////////////////////
00103 private:
00104     const EventType::EN_EventType _eventType;
00105
00106     ProgressStatus _typeSpecificProgressStatus;
00107
00108     ProgressStatus _generatorProgressStatus;
00109
00110     ProgressStatus _overallProgressStatus;
00111
00112     EventGeneratorKey_T _generatorKey;
00113 };
00114
00115 }
00116 #endif // __STDAIR_BASIC_PROGRESSSTATUSSET_HPP

```

35.99 stdair/basic/RandomGeneration.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <boost/version.hpp>
#include <stdair/basic/RandomGeneration.hpp>

```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.100 RandomGeneration.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost
00008 #include <boost/version.hpp>
00009 #if BOOST_VERSION >= 103500
00010 #include <boost/math/distributions/normal.hpp>
00011 #endif // BOOST_VERSION >= 103500
00012 // StdAir
00013 #include <stdair/basic/RandomGeneration.hpp>
00014
00015 namespace stdair {
00016
00017 // //////////////////////////////////////

```

```

00023 RandomGeneration::RandomGeneration() : _generator (1) {
00024 }
00025
00026 // //////////////////////////////////////
00027 RandomGeneration::RandomGeneration (const RandomSeed_T& iSeed)
00028 : _generator (iSeed) {
00029 }
00030
00031 // //////////////////////////////////////
00032 RandomGeneration::RandomGeneration (const RandomGeneration& iRandomGeneration)
00033 : _generator (iRandomGeneration._generator) {
00034 }
00035
00036 // //////////////////////////////////////
00037 RandomGeneration::~RandomGeneration() {
00038 }
00039
00040 // //////////////////////////////////////
00041 void RandomGeneration::init (const RandomSeed_T& iSeed) {
00042     _generator.seed (iSeed);
00043 }
00044
00045 // //////////////////////////////////////
00046 const std::string RandomGeneration::describe() const {
00047     std::ostringstream ostr;
00048     ostr << _generator;
00049     return ostr.str();
00050 }
00051
00052 // //////////////////////////////////////
00053 RealNumber_T RandomGeneration::generateUniform01() {
00054     UniformGenerator_T lGenerator (_generator, boost::uniform_real<>(0, 1));
00055     return lGenerator();
00056 }
00057
00058 // //////////////////////////////////////
00059 RealNumber_T RandomGeneration::generateUniform(const RealNumber_T& iMinValue,
00060                                                const RealNumber_T& iMaxValue) {
00061     const Probability_T lVariateUnif01 = generateUniform01();
00062     const RealNumber_T lVariateUnif =
00063         iMinValue + lVariateUnif01 * (iMaxValue - iMinValue);
00064     return lVariateUnif;
00065 }
00066
00067 // //////////////////////////////////////
00068 RealNumber_T RandomGeneration::generateNormal (const RealNumber_T& mu,
00069                                                const RealNumber_T& sigma) {
00070
00071     #if BOOST_VERSION >= 103500
00072         const Probability_T lVariateUnif = generateUniform01();
00073         const boost::math::normal lNormal (mu, sigma);
00074         const RealNumber_T lRealNumberOfRequestsToBeGenerated =
00075             boost::math::quantile (lNormal, lVariateUnif);
00076     #else // BOOST_VERSION >= 103500
00077         // TODO: rely on GSL when Boost version smaller than 1.35
00078         const RealNumber_T lRealNumberOfRequestsToBeGenerated = 0.0;
00079     #endif // BOOST_VERSION >= 103500
00080
00081     return lRealNumberOfRequestsToBeGenerated;
00082 }
00083 }

```

```

00084
00085 // //////////////////////////////////////
00086 RealNumber_T RandomGeneration::generateExponential (const RealNumber_T& lambda)
00087 {
00091     ExponentialDistribution_T lExponentialDistribution (lambda);
00092
00094     ExponentialGenerator_T lExponentialDistributionGenerator (_generator,
00095                                                                lExponentialDistrib
00096                                                                ution);
00097
00098     // Generate a random variate, expressed in (fractional) day
00099     const RealNumber_T lExponentialVariateInDays =
00100         lExponentialDistributionGenerator();
00101
00102     return lExponentialVariateInDays;
00103 }
00104 }

```

35.101 stdair/basic/RandomGeneration.hpp File Reference

```

#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/basic/StructAbstract.hpp>

```

Classes

- struct [stdair::RandomGeneration](#)
Class holding a random generator.

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.102 RandomGeneration.hpp

```

00001 #ifndef __STDAIR_BAS_RANDOMGENERATION_HPP
00002 #define __STDAIR_BAS_RANDOMGENERATION_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009 #include <stdair/stdair_maths_types.hpp>
00010 #include <stdair/basic/StructAbstract.hpp>
00011
00012 namespace stdair {
00013
00017     struct RandomGeneration : public StructAbstract {
00018     public:

```

```

00019 // //////////// Business Methods ////////////
00024 RealNumber_T generateUniform01();
00025
00030 RealNumber_T operator() () {
00031     return generateUniform01();
00032 }
00033
00039 RealNumber_T generateUniform (const RealNumber_T&, const RealNumber_T&);
00040
00045 RealNumber_T generateNormal (const RealNumber_T&, const RealNumber_T&);
00046
00051 RealNumber_T generateExponential (const RealNumber_T&);
00052
00056 BaseGenerator_T& getBaseGenerator () { return _generator; }
00057
00058
00059 public:
00060 // //////////// Display Support Methods ////////////
00064 const std::string describe() const;
00065
00066
00067 public:
00068 // //////////// Constructors and destructors ////////////
00072 RandomGeneration (const RandomSeed_T&);
00076 RandomGeneration();
00077
00078 private:
00082 RandomGeneration (const RandomGeneration&);
00086 RandomGeneration& operator= (const RandomGeneration& iRandomGeneration) {
00087     _generator = iRandomGeneration._generator;
00088     return *this;
00089 }
00090 public:
00094 ~RandomGeneration();
00095
00103 void init (const RandomSeed_T&);
00104
00105 // //////////// Attributes ////////////
00112 BaseGenerator_T _generator;
00113 };
00114
00115 }
00116 #endif // __STDAIR_BAS_RANDOMGENERATION_HPP

```

35.103 stdair/basic/SampleType.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/SampleType.hpp>

```

Namespaces

- namespace `stdair`

Handle on the StdAir library context.

35.104 SampleType.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/stdair_exceptions.hpp>
00009 #include <stdair/basic/SampleType.hpp>
00010
00011 namespace stdair {
00012
00013 // //////////////////////////////////////
00014 const std::string SampleType::_labels[LAST_VALUE] =
00015     { "All", "AllForPartnerships", "RevenueManagement", "Inventory", "Schedule",
00016       "RevenueAccounting", "FareQuote", "CRS", "DemandGeneration", "EventManagemen
00017       nt",
00018       "CustomerChoice" };
00019
00020 // //////////////////////////////////////
00021 const char SampleType::_typeLabels[LAST_VALUE] = { 'A', 'P', 'R', 'I', 'S', 'T', 'F', 'C', 'D', 'E', '
00022       M' };
00023
00024 // //////////////////////////////////////
00025 SampleType::SampleType()
00026     : _type (LAST_VALUE) {
00027     assert (false);
00028 }
00029
00030 // //////////////////////////////////////
00031 SampleType::SampleType (const SampleType& iSampleType)
00032     : _type (iSampleType._type) {
00033 }
00034
00035 // //////////////////////////////////////
00036 SampleType::SampleType (const EN_SampleType& iSampleType)
00037     : _type (iSampleType) {
00038 }
00039
00040 // //////////////////////////////////////
00041 SampleType::SampleType (const char iType) {
00042     switch (iType) {
00043     case 'A': _type = ALL; break;
00044     case 'P': _type = A4P; break;
00045     case 'R': _type = RMS; break;
00046     case 'I': _type = INV; break;
00047     case 'S': _type = SCH; break;
00048     case 'T': _type = RAC; break;
00049     case 'F': _type = FQT; break;
00050     case 'C': _type = CRS; break;
00051     case 'D': _type = DEM; break;
00052     case 'E': _type = EVT; break;
00053     case 'M': _type = CCM; break;
00054     default: _type = LAST_VALUE; break;
00055     }
00056
00057     if (_type == LAST_VALUE) {
00058         const std::string& lLabels = describeLabels();

```

```

00059         std::ostringstream oMessage;
00060         oMessage << "The sample type '" << iType
00061             << "' is not known. Known sample types: " << lLabels;
00062         throw CodeConversionException (oMessage.str());
00063     }
00064 }
00065
00066 // //////////////////////////////////////
00067 const std::string& SampleType::getLabel (const EN_SampleType& iType) {
00068     return _labels[iType];
00069 }
00070
00071 // //////////////////////////////////////
00072 char SampleType::getTypeLabel (const EN_SampleType& iType) {
00073     return _typeLabels[iType];
00074 }
00075
00076 // //////////////////////////////////////
00077 std::string SampleType::getTypeLabelAsString (const EN_SampleType& iType) {
00078     std::ostringstream ostr;
00079     ostr << _typeLabels[iType];
00080     return ostr.str();
00081 }
00082
00083 // //////////////////////////////////////
00084 std::string SampleType::describeLabels() {
00085     std::ostringstream ostr;
00086     for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
00087         if (idx != 0) {
00088             ostr << ", ";
00089         }
00090         ostr << _labels[idx];
00091     }
00092     return ostr.str();
00093 }
00094
00095 // //////////////////////////////////////
00096 SampleType::EN_SampleType SampleType::getType() const {
00097     return _type;
00098 }
00099
00100 // //////////////////////////////////////
00101 std::string SampleType::getTypeAsString() const {
00102     std::ostringstream ostr;
00103     ostr << _typeLabels[_type];
00104     return ostr.str();
00105 }
00106
00107 // //////////////////////////////////////
00108 const std::string SampleType::describe() const {
00109     std::ostringstream ostr;
00110     ostr << _labels[_type];
00111     return ostr.str();
00112 }
00113
00114 // //////////////////////////////////////
00115 bool SampleType::operator==(const EN_SampleType& iType) const {
00116     return (_type == iType);
00117 }
00118
00119 }

```

35.105 stdair/basic/SampleType.hpp File Reference

```
#include <string>
#include <stdair/basic/StructAbstract.hpp>
```

Classes

- struct [stdair::SampleType](#)
Enumeration of BOM sample types.

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.106 SampleType.hpp

```
00001 #ifndef __STDAIR_BAS_SAMPLETYPE_HPP
00002 #define __STDAIR_BAS_SAMPLETYPE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/basic/StructAbstract.hpp>
00011
00012 namespace stdair {
00013
00025     struct SampleType : public StructAbstract {
00026     public:
00027         typedef enum {
00028             ALL = 0,
00029             A4P,
00030             RMS,
00031             INV,
00032             SCH,
00033             RAC,
00034             FQT,
00035             CRS,
00036             DEM,
00037             EVT,
00038             CCM,
00039             LAST_VALUE
00040         } EN_SampleType;
00041
00045         static const std::string& getLabel (const EN_SampleType&);
00046
00050         static char getTypeLabel (const EN_SampleType&);
00051
00055         static std::string getTypeLabelAsString (const EN_SampleType&);
00056
00060         static std::string describeLabels();

```

```

00061
00065     EN_SampleType getType() const;
00066
00070     std::string getTypeAsString() const;
00071
00075     const std::string describe() const;
00076
00077 public:
00081     bool operator== (const EN_SampleType&) const;
00082
00083 public:
00087     SampleType (const EN_SampleType&);
00091     SampleType (const char iType);
00095     SampleType (const SampleType&);
00096
00097 private:
00101     SampleType();
00102
00103
00104 private:
00108     static const std::string _labels[LAST_VALUE];
00109
00113     static const char _typeLabels[LAST_VALUE];
00114
00115
00116 private:
00117     // ////////// Attributes //////////
00121     EN_SampleType _type;
00122 };
00123
00124 }
00125 #endif // __STDAIR_BAS_SAMPLETYPE_HPP

```

35.107 stdair/basic/ServiceInitialisationType.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/ServiceInitialisationType.hpp>

```

Namespaces

- namespace `stdair`
Handle on the *StdAir* library context.

35.108 ServiceInitialisationType.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>

```

```

00007 // StdAir
00008 #include <stdair/stdair_exceptions.hpp>
00009 #include <stdair/basic/ServiceInitialisationType.hpp>
00010
00011 namespace stdair {
00012
00013     // //////////////////////////////////////
00014     const std::string ServiceInitialisationType::_labels[LAST_VALUE] =
00015         { "Not yet initialised", "File parsing", "Built-in sample BOM" };
00016
00017     // //////////////////////////////////////
00018     const char ServiceInitialisationType::_typeLabels[LAST_VALUE] =
00019         { 'N', 'F', 'B' };
00020
00021
00022     // //////////////////////////////////////
00023     ServiceInitialisationType::ServiceInitialisationType()
00024         : _type (LAST_VALUE) {
00025         assert (false);
00026     }
00027
00028     // //////////////////////////////////////
00029     ServiceInitialisationType::
00030     ServiceInitialisationType (const ServiceInitialisationType& iServiceInitialisat
00031 ionType)
00032         : _type (iServiceInitialisationType._type) {
00033     }
00034
00035     // //////////////////////////////////////
00036     ServiceInitialisationType::
00037     ServiceInitialisationType (const EN_ServiceInitialisationType& iServiceInitiali
00038 sationType)
00039         : _type (iServiceInitialisationType) {
00040     }
00041
00042     // //////////////////////////////////////
00043     ServiceInitialisationType::EN_ServiceInitialisationType
00044     ServiceInitialisationType::getType (const char iTypeChar) {
00045         EN_ServiceInitialisationType oType;
00046         switch (iTypeChar) {
00047             case 'N': oType = NOT_YET_INITIALISED; break;
00048             case 'F': oType = FILE_PARSING; break;
00049             case 'B': oType = BUILTIN_SAMPLE; break;
00050             default: oType = LAST_VALUE; break;
00051         }
00052
00053         if (oType == LAST_VALUE) {
00054             const std::string& lLabels = describeLabels();
00055             std::ostringstream oMessage;
00056             oMessage << "The service initialisation type '" << iTypeChar
00057                 << "' is not known. "
00058                 << "Known service initialisation types: " << lLabels;
00059             throw CodeConversionException (oMessage.str());
00060         }
00061
00062         return oType;
00063     }
00064
00065     // //////////////////////////////////////
00066     ServiceInitialisationType::
00067     ServiceInitialisationType (const char iTypeChar)
00068         : _type (getType (iTypeChar)) {

```

```

00067     }
00068
00069     // //////////////////////////////////////
00070     ServiceInitialisationType::
00071     ServiceInitialisationType (const std::string& iTypeStr) {
00072         //
00073         const size_t lSize = iTypeStr.size();
00074         assert (lSize == 1);
00075         const char lTypeChar = iTypeStr[0];
00076         _type = getType (lTypeChar);
00077     }
00078
00079     // //////////////////////////////////////
00080     const std::string& ServiceInitialisationType::
00081     getLabel (const EN_ServiceInitialisationType& iType) {
00082         return _labels[iType];
00083     }
00084
00085     // //////////////////////////////////////
00086     char ServiceInitialisationType::
00087     getTypeLabel (const EN_ServiceInitialisationType& iType) {
00088         return _typeLabels[iType];
00089     }
00090
00091     // //////////////////////////////////////
00092     std::string ServiceInitialisationType::
00093     getTypeLabelAsString (const EN_ServiceInitialisationType& iType) {
00094         std::ostringstream ostr;
00095         ostr << _typeLabels[iType];
00096         return ostr.str();
00097     }
00098
00099     // //////////////////////////////////////
00100     std::string ServiceInitialisationType::describeLabels() {
00101         std::ostringstream ostr;
00102         for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
00103             if (idx != 0) {
00104                 ostr << ", ";
00105             }
00106             ostr << _labels[idx];
00107         }
00108         return ostr.str();
00109     }
00110
00111     // //////////////////////////////////////
00112     ServiceInitialisationType::EN_ServiceInitialisationType
00113     ServiceInitialisationType::getType() const {
00114         return _type;
00115     }
00116
00117     // //////////////////////////////////////
00118     char ServiceInitialisationType::getTypeAsChar() const {
00119         const char oTypeChar = _typeLabels[_type];
00120         return oTypeChar;
00121     }
00122
00123     // //////////////////////////////////////
00124     std::string ServiceInitialisationType::getTypeAsString() const {
00125         std::ostringstream ostr;
00126         ostr << _typeLabels[_type];
00127         return ostr.str();
00128     }

```

```

00129
00130 // //////////////////////////////////////
00131 const std::string ServiceInitialisationType::describe() const {
00132     std::ostringstream ostr;
00133     ostr << _labels[_type];
00134     return ostr.str();
00135 }
00136
00137 // //////////////////////////////////////
00138 bool ServiceInitialisationType::
00139 operator== (const EN_ServiceInitialisationType& iType) const {
00140     return (_type == iType);
00141 }
00142
00143 }

```

35.109 stdair/basic/ServiceInitialisationType.hpp File Reference

```

#include <string>
#include <stdair/basic/StructAbstract.hpp>

```

Classes

- struct [stdair::ServiceInitialisationType](#)
Enumeration of service initialisation types.

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.110 ServiceInitialisationType.hpp

```

00001 #ifndef __STDAIR_BAS_SERVICEINITIALISATIONTYPE_HPP
00002 #define __STDAIR_BAS_SERVICEINITIALISATIONTYPE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/basic/StructAbstract.hpp>
00011
00012 namespace stdair {
00013
00017     struct ServiceInitialisationType : public StructAbstract {
00018     public:
00019         typedef enum {
00020             NOT_YET_INITIALISED = 0,
00021             FILE_PARSING,
00022             BUILTIN_SAMPLE,

```

```

00023         LAST_VALUE
00024     } EN_ServiceInitialisationType;
00025
00030     static const std::string& getLabel (const EN_ServiceInitialisationType&);
00031
00035     static EN_ServiceInitialisationType getType (const char);
00036
00040     static char getTypeLabel (const EN_ServiceInitialisationType&);
00041
00045     static std::string
00046     getTypeLabelAsString (const EN_ServiceInitialisationType&);
00047
00051     static std::string describeLabels();
00052
00056     EN_ServiceInitialisationType getType() const;
00057
00061     char getTypeAsChar() const;
00062
00066     std::string getTypeAsString() const;
00067
00072     const std::string describe() const;
00073
00074 public:
00078     bool operator== (const EN_ServiceInitialisationType&) const;
00079
00080 public:
00084     ServiceInitialisationType (const EN_ServiceInitialisationType&);
00088     ServiceInitialisationType (const char iType);
00092     ServiceInitialisationType (const std::string& iType);
00096     ServiceInitialisationType (const ServiceInitialisationType&);
00097
00098 private:
00102     ServiceInitialisationType();
00103
00104
00105 private:
00109     static const std::string _labels[LAST_VALUE];
00113     static const char _typeLabels[LAST_VALUE];
00114
00115 private:
00116     // ////////// Attributes //////////
00120     EN_ServiceInitialisationType _type;
00121 };
00122
00123 }
00124 #endif // __STDAIR_BAS_SERVICEINITIALISATIONTYPE_HPP

```

35.111 stdair/basic/StructAbstract.hpp File Reference

```

#include <iosfwd>
#include <string>

```

Classes

- struct [stdair::StructAbstract](#)

Base class for the light structures.

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

Functions

- `template<class charT , class traits >`
`std::basic_ostream< charT, traits > & operator<< (std::basic_ostream< charT,`
`traits > &ioOut, const stdair::StructAbstract &iStruct)`
- `template<class charT , class traits >`
`std::basic_istream< charT, traits > & operator>> (std::basic_istream< charT,`
`traits > &ioln, stdair::StructAbstract &ioStruct)`

35.111.1 Function Documentation

35.111.1.1 `template<class charT , class traits > std::basic_ostream<charT, traits>&`
`operator<< (std::basic_ostream< charT, traits > & ioOut, const`
`stdair::StructAbstract & iStruct) [inline]`

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (p653) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line [61](#) of file [StructAbstract.hpp](#).

References [stdair::StructAbstract::toStream\(\)](#).

35.111.1.2 `template<class charT , class traits > std::basic_istream<charT, traits>&`
`operator>> (std::basic_istream< charT, traits > & ioln, stdair::StructAbstract`
`& ioStruct) [inline]`

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (pp655-657) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line [89](#) of file [StructAbstract.hpp](#).

References [stdair::StructAbstract::fromStream\(\)](#).

35.112 StructAbstract.hpp

```
00001 #ifndef __STDAIR_BAS_STRUCTABSTRACT_HPP
00002 #define __STDAIR_BAS_STRUCTABSTRACT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010
```

```

00011 namespace stdair {
00012
00016     struct StructAbstract {
00017     public:
00018
00022         virtual ~StructAbstract() {}
00023
00029         void toStream (std::ostream& ioOut) const {
00030             ioOut << describe();
00031         }
00032
00038         virtual void fromStream (std::istream& ioIn) {}
00039
00043         virtual const std::string describe() const = 0;
00044
00045     protected:
00049         StructAbstract() {}
00050     };
00051 }
00052
00058 template <class charT, class traits>
00059 inline
00060 std::basic_ostream<charT, traits>&
00061 operator<< (std::basic_ostream<charT, traits>& ioOut,
00062            const stdair::StructAbstract& iStruct) {
00068     std::basic_ostringstream<charT,traits> ostr;
00069     ostr.copyfmt (ioOut);
00070     ostr.width (0);
00071
00072     // Fill string stream
00073     iStruct.toStream (ostr);
00074
00075     // Print string stream
00076     ioOut << ostr.str();
00077
00078     return ioOut;
00079 }
00080
00086 template <class charT, class traits>
00087 inline
00088 std::basic_istream<charT, traits>&
00089 operator>> (std::basic_istream<charT, traits>& ioIn,
00090            stdair::StructAbstract& ioStruct) {
00091     // Fill the Structure object with the input stream.
00092     ioStruct.fromStream (ioIn);
00093     return ioIn;
00094 }
00095 }
00096 #endif // __STDAIR_BAS_STRUCTABSTRACT_HPP

```

35.113 stdair/basic/YieldRange.cpp File Reference

```

#include <limits>

#include <sstream>

#include <stdair/basic/YieldRange.hpp>

```

Namespaces

- namespace `stdair`

Handle on the StdAir library context.

35.114 YieldRange.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <limits>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/YieldRange.hpp>
00009
00010 namespace stdair {
00011
00012 // //////////////////////////////////////
00013 YieldRange::YieldRange() :
00014     _upperYield (std::numeric_limits<Yield_T>::max()),
00015     _averageYield (std::numeric_limits<Yield_T>::max()),
00016     _lowerYield (std::numeric_limits<Yield_T>::min()) {
00017 }
00018
00019 // //////////////////////////////////////
00020 YieldRange::YieldRange (const YieldRange& iYieldRange) :
00021     _upperYield (iYieldRange.getUpperYield()),
00022     _averageYield (iYieldRange.getAverageYield()),
00023     _lowerYield (std::numeric_limits<Yield_T>::min()) {
00024 }
00025
00026 // //////////////////////////////////////
00027 YieldRange::YieldRange (const Yield_T iUpperYield) :
00028     _upperYield (iUpperYield), _averageYield (iUpperYield),
00029     _lowerYield (iUpperYield) {
00030 }
00031
00032 // //////////////////////////////////////
00033 YieldRange::YieldRange (const Yield_T iUpperYield,
00034                         const Yield_T iAverageYield) :
00035     _upperYield (iUpperYield), _averageYield (iAverageYield),
00036     _lowerYield (std::numeric_limits<Yield_T>::min()) {
00037 }
00038
00039 // //////////////////////////////////////
00040 YieldRange::YieldRange (const Yield_T iUpperYield,
00041                         const Yield_T iAverageYield,
00042                         const Yield_T iLowerYield) :
00043     _upperYield (iUpperYield), _averageYield (iAverageYield),
00044     _lowerYield (iLowerYield) {
00045 }
00046
00047 // //////////////////////////////////////
00048 YieldRange::~YieldRange() {
00049 }
00050
00051 // //////////////////////////////////////
00052 void YieldRange::toStream (std::ostream& ioOut) const {
00053     ioOut << _averageYield << "[" << _lowerYield << ", "

```

```

00054         << _upperYield << "]";
00055     }
00056
00057     // //////////////////////////////////////
00058     void YieldRange::fromStream (std::istream& ioIn) {
00059     }
00060
00061     // //////////////////////////////////////
00062     const std::string YieldRange::describe() const {
00063         std::ostringstream oStr;
00064
00065         return oStr.str();
00066     }
00067
00068 }

```

35.115 stdair/basic/YieldRange.hpp File Reference

```

#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/StructAbstract.hpp>

```

Classes

- class [stdair::YieldRange](#)

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.116 YieldRange.hpp

```

00001 #ifndef __STDAIR_BAS_YELDRANGE_HPP
00002 #define __STDAIR_BAS_YELDRANGE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STDAIR
00008 #include <stdair/stdair_inventory_types.hpp>
00009 #include <stdair/basic/StructAbstract.hpp>
00010
00011 namespace stdair {
00012
00023     class YieldRange : public StructAbstract {
00024     public:
00026         YieldRange ();
00027         YieldRange (const YieldRange&);
00028         YieldRange (const Yield_T iUpperYield);
00029         YieldRange (const Yield_T iUpperYield, const Yield_T iAverageYield);
00030         YieldRange (const Yield_T iUpperYield, const Yield_T iAverageYield,
00031                     const Yield_T iLowerYield);
00032

```

```

00034     virtual ~YieldRange();
00035
00036
00037     // //////////// Getters ////////////
00039     Yield_T getUpperYield() const {
00040         return _upperYield;
00041     }
00043     Yield_T getAverageYield() const {
00044         return _averageYield;
00045     }
00047     Yield_T getLowerYield() const {
00048         return _lowerYield;
00049     }
00050
00051     // //////////// Setters ////////////
00053     void setUpperYield (const Yield_T iUpperYield) {
00054         _upperYield = iUpperYield;
00055     }
00057     void setAverageYield (const Yield_T iAverageYield) {
00058         _averageYield = iAverageYield;
00059     }
00061     void setLowerYield (const Yield_T iLowerYield) {
00062         _lowerYield = iLowerYield;
00063     }
00064
00065
00066     // //////////// Display methods ////////////
00069     void toStream (std::ostream&) const;
00070
00073     void fromStream (std::istream&);
00074
00076     const std::string describe() const;
00077
00078 private:
00079     // //////////// Attributes ////////////
00081     Yield_T _upperYield;
00082
00084     Yield_T _averageYield;
00085
00087     Yield_T _lowerYield;
00088 };
00089 }
00090 #endif // __STDAIR_BAS_YIELDRANGE_HPP

```

35.117 stdair/bom/AirlineClassList.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/AirlineClassList.hpp>

```

Namespaces

- namespace `stdair`

Handle on the StdAir library context.

35.118 AirlineClassList.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/access.hpp>
00011 // StdAir
00012 #include <stdair/basic/BasConst_Inventory.hpp>
00013 #include <stdair/service/Logger.hpp>
00014 #include <stdair/bom/AirlineClassList.hpp>
00015
00016 namespace stdair {
00017
00018 // //////////////////////////////////////
00019 AirlineClassList::AirlineClassList()
00020 : _key (DEFAULT_AIRLINE_CODE_LIST, DEFAULT_CLASS_CODE_LIST),
00021   _parent (NULL) {
00022     assert (false);
00023 }
00024
00025 // //////////////////////////////////////
00026 AirlineClassList::AirlineClassList (const AirlineClassList& iACL)
00027 : _key (iACL._key), _parent (NULL) {
00028     assert (false);
00029 }
00030
00031 // //////////////////////////////////////
00032 AirlineClassList::AirlineClassList (const Key_T& iKey)
00033 : _key (iKey), _parent (NULL) {
00034 }
00035
00036 // //////////////////////////////////////
00037 AirlineClassList::~AirlineClassList() {
00038 }
00039
00040 // //////////////////////////////////////
00041 std::string AirlineClassList::toString() const {
00042     std::ostringstream oStr;
00043     oStr << describeKey() << ", " << _yield << ", " << _fare;
00044     return oStr.str();
00045 }
00046
00047 // //////////////////////////////////////
00048 void AirlineClassList::serialisationImplementationExport() const {
00049     std::ostringstream oStr;
00050     boost::archive::text_oarchive oa (oStr);
00051     oa << *this;
00052 }
00053

```

```

00054 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00055 void AirlineClassList::serialisationImplementationImport() {
00056     std::istringstream iStr;
00057     boost::archive::text_iarchive ia (iStr);
00058     ia >> *this;
00059 }
00060
00061 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00062 template<class Archive>
00063 void AirlineClassList::serialize (Archive& ioArchive,
00064                                   const unsigned int iFileVersion) {
00065     ioArchive & _key & _yield & _fare;
00066 }
00067
00068 }
00069
00070
00071

```

35.119 stdair/bom/AirlineClassList.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/AirlineClassListKey.hpp>
#include <stdair/bom/AirlineClassListTypes.hpp>

```

Classes

- class [stdair::AirlineClassList](#)
Class representing the actual attributes for a segment-features.

Namespaces

- namespace [boost](#)
Forward declarations.
- namespace [boost::serialization](#)
- namespace [stdair](#)
Handle on the StdAir library context.

35.120 AirlineClassList.hpp

```

00001 #ifndef __STDAIR_BOM_AIRLINECLASSLIST_HPP
00002 #define __STDAIR_BOM_AIRLINECLASSLIST_HPP
00003
00004 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00005 // Import section
00006 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00007 // STL

```

```

00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/bom/BomAbstract.hpp>
00012 #include <stdair/bom/AirlineClassListKey.hpp>
00013 #include <stdair/bom/AirlineClassListTypes.hpp>
00014
00016 namespace boost {
00017     namespace serialization {
00018         class access;
00019     }
00020 }
00021
00022 namespace stdair {
00023
00027     class AirlineClassList : public BomAbstract {
00028     template <typename BOM> friend class FacBom;
00029     friend class FacBomManager;
00030     friend class boost::serialization::access;
00031
00032     public:
00033         // /////////// Type definitions ///////////
00037         typedef AirlineClassListKey Key_T;
00038
00039
00040     public:
00041         // /////////// Getters ///////////
00043         const Key_T& getKey() const {
00044             return _key;
00045         }
00046
00048         BomAbstract* const getParent() const {
00049             return _parent;
00050         }
00051
00053         const AirlineCodeList_T& getAirlineCodeList() const {
00054             return _key.getAirlineCodeList();
00055         }
00056
00058         const ClassList_StringList_T& getClassCodeList() const {
00059             return _key.getClassCodeList();
00060         }
00061
00063         const HolderMap_T& getHolderMap() const {
00064             return _holderMap;
00065         }
00066
00068         const stdair::Yield_T& getYield() const {
00069             return _yield;
00070         }
00071
00073         const stdair::Fare_T& getFare() const {
00074             return _fare;
00075         }
00076
00077     public:
00078         // /////////// Setters ///////////
00079         void setYield (const Yield_T& iYield) {
00080             _yield = iYield;
00081         }
00082
00083         void setFare (const Fare_T& iFare) {

```

```

00084     _fare = iFare;
00085 }
00086
00087 public:
00088     // //////////// Display support methods ////////////
00094     void toStream (std::ostream& ioOut) const {
00095         ioOut << toString();
00096     }
00097
00103     void fromStream (std::istream& ioIn) {
00104     }
00105
00109     std::string toString() const;
00110
00114     const std::string describeKey() const {
00115         return _key.toString();
00116     }
00117
00118
00119 public:
00120     // //////////// (Boost) Serialisation support methods ////////////
00124     template<class Archive>
00125     void serialize (Archive& ar, const unsigned int iFileVersion);
00126
00127 private:
00132     void serialisationImplementationExport() const;
00133     void serialisationImplementationImport();
00134
00135
00136 protected:
00137     // //////////// Constructors and destructors ////////////
00141     AirlineClassList (const Key_T&);
00145     virtual ~AirlineClassList();
00146
00147 private:
00151     AirlineClassList();
00152
00156     AirlineClassList (const AirlineClassList&);
00157
00158
00159 protected:
00160     // //////////// Attributes ////////////
00164     Key_T _key;
00165
00169     BomAbstract* _parent;
00170
00174     HolderMap_T _holderMap;
00175
00176     /*
00177      * Yield value.
00178      */
00179     Yield_T _yield;
00180
00181     /*
00182      * Fare value.
00183      */
00184     Fare_T _fare;
00185 };
00186
00187 }
00188 #endif // __STDAIR_BOM_AIRLINECLASSLIST_HPP
00189

```

35.121 stdair/bom/AirlineClassListKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/AirlineClassListKey.hpp>
```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

Functions

- template void [stdair::AirlineClassListKey::serialize< ba::text_oarchive >](#) (ba::text_oarchive &, unsigned int)
- template void [stdair::AirlineClassListKey::serialize< ba::text_iarchive >](#) (ba::text_iarchive &, unsigned int)

35.122 AirlineClassListKey.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/access.hpp>
00011 // StdAir
00012 #include <stdair/basic/BasConst_BomDisplay.hpp>
00013 #include <stdair/bom/AirlineClassListKey.hpp>
00014
00015 namespace stdair {
00016
00017 // //////////////////////////////////////
00018 AirlineClassListKey::AirlineClassListKey() {
00019     assert (false);
00020 }
00021
00022 // //////////////////////////////////////
00023 AirlineClassListKey::
00024 AirlineClassListKey (const AirlineCodeList_T& iAirlineCodeList,
00025                     const ClassList_StringList_T& iClassCodeList)
00026     : _airlineCodeList (iAirlineCodeList), _classCodeList (iClassCodeList) {
```

```

00027     }
00028
00029     // //////////////////////////////////////
00030     AirlineClassListKey::AirlineClassListKey (const AirlineClassListKey& iKey)
00031         : _airlineCodeList (iKey._airlineCodeList),
00032           _classCodeList (iKey._classCodeList) {
00033     }
00034
00035     // //////////////////////////////////////
00036     AirlineClassListKey::~AirlineClassListKey() {
00037     }
00038
00039     // //////////////////////////////////////
00040     void AirlineClassListKey::toStream (std::ostream& ioOut) const {
00041         ioOut << "AirlineClassListKey: " << toString() << std::endl;
00042     }
00043
00044     // //////////////////////////////////////
00045     void AirlineClassListKey::fromStream (std::istream& ioIn) {
00046     }
00047
00048     // //////////////////////////////////////
00049     const std::string AirlineClassListKey::toString() const {
00050         std::ostringstream oStr;
00051         assert (_airlineCodeList.size() == _classCodeList.size());
00052
00053         unsigned short idx = 0;
00054         AirlineCodeList_T::const_iterator itAirlineCode = _airlineCodeList.begin();
00055         for (ClassList_StringList_T::const_iterator itClassCode =
00056             _classCodeList.begin(); itClassCode != _classCodeList.end();
00057             ++itClassCode, ++itAirlineCode, ++idx) {
00058             if (idx != 0) {
00059                 oStr << DEFAULT_KEY_SUB_FLD_DELIMITER << " ";
00060             }
00061
00062             const AirlineCode_T& lAirlineCode = *itAirlineCode;
00063             const ClassCode_T& lClassCode = *itClassCode;
00064             oStr << lAirlineCode << " " << lClassCode;
00065         }
00066
00067         return oStr.str();
00068     }
00069
00070     // //////////////////////////////////////
00071     void AirlineClassListKey::serialisationImplementationExport() const {
00072         std::ostringstream oStr;
00073         boost::archive::text_oarchive oa (oStr);
00074         oa << *this;
00075     }
00076
00077     // //////////////////////////////////////
00078     void AirlineClassListKey::serialisationImplementationImport() {
00079         std::istringstream iStr;
00080         boost::archive::text_iarchive ia (iStr);
00081         ia >> *this;
00082     }
00083
00084     // //////////////////////////////////////
00085     template<class Archive>
00086     void AirlineClassListKey::serialize (Archive& ioArchive,
00087                                         const unsigned int iFileVersion) {
00088         AirlineCodeList_T::const_iterator itAirlineCode = _airlineCodeList.begin();

```

```

00097     for (ClassList_StringList_T::const_iterator itClassCode =
00098           _classCodeList.begin(); itClassCode != _classCodeList.end();
00099           ++itClassCode, ++itAirlineCode) {
00100         AirlineCode_T lAirlineCode = *itAirlineCode;
00101         ClassCode_T lClassCode = *itClassCode;
00102
00103         ioArchive & lAirlineCode & lClassCode;
00104     }
00105 }
00106
00107 // //////////////////////////////////////
00108 // Explicit template instantiation
00109 namespace ba = boost::archive;
00110 template void AirlineClassListKey::
00111     serialize<ba::text_oarchive> (ba::text_oarchive&, unsigned int);
00112 template void AirlineClassListKey::
00113     serialize<ba::text_iarchive> (ba::text_iarchive&, unsigned int);
00114 // //////////////////////////////////////
00115
00116 }

```

35.123 stdair/bom/AirlineClassListKey.hpp File Reference

```

#include <iosfwd>

#include <stdair/stdair_inventory_types.hpp>

#include <stdair/bom/KeyAbstract.hpp>

```

Classes

- struct [stdair::AirlineClassListKey](#)
Key of airport-pair.

Namespaces

- namespace [boost](#)
Forward declarations.
- namespace [boost::serialization](#)
- namespace [stdair](#)
Handle on the StdAir library context.

35.124 AirlineClassListKey.hpp

```

00001 #ifndef __STDAIR_BOM_AIRLINECLASSLISTKEY_HPP
00002 #define __STDAIR_BOM_AIRLINECLASSLISTKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>

```

```

00009 // StdAir
00010 #include <stdair/stdair_inventory_types.hpp>
00011 #include <stdair/bom/KeyAbstract.hpp>
00012
00013 namespace boost {
00014     namespace serialization {
00015         class access;
00016     }
00017 }
00018
00019 namespace stdair {
00020
00021     struct AirlineClassListKey : public KeyAbstract {
00022         friend class boost::serialization::access;
00023
00024         // ////////// Constructors and destructors //////////
00025     private:
00026         AirlineClassListKey();
00027
00028     public:
00029         AirlineClassListKey (const AirlineCodeList_T&,
00030                             const ClassList_StringList_T&);
00031
00032         AirlineClassListKey (const AirlineClassListKey&);
00033
00034         ~AirlineClassListKey();
00035
00036     public:
00037         // ////////// Getters //////////
00038         const AirlineCodeList_T& getAirlineCodeList() const {
00039             return _airlineCodeList;
00040         }
00041
00042         const ClassList_StringList_T& getClassCodeList() const {
00043             return _classCodeList;
00044         }
00045
00046     public:
00047         // ////////// Display support methods //////////
00048         void toStream (std::ostream& ioOut) const;
00049
00050         void fromStream (std::istream& ioIn);
00051
00052         const std::string toString() const;
00053
00054     public:
00055         // ////////// (Boost) Serialisation support methods //////////
00056         template<class Archive>
00057         void serialize (Archive& ar, const unsigned int iFileVersion);
00058
00059     private:
00060         void serialisationImplementationExport() const;
00061         void serialisationImplementationImport();
00062
00063     private:
00064         // ////////// Attributes //////////
00065         AirlineCodeList_T _airlineCodeList;
00066
00067
00068
00069
00070
00071
00072
00073
00074
00075
00076
00077
00078
00079
00080
00081
00082
00083
00084
00085
00086
00087
00088
00089
00090
00091
00092
00093
00094
00095
00096
00097
00098
00099
00100
00101
00102
00103
00104
00105
00106
00107
00108
00109
00110
00111
00112
00113
00114
00115
00116
00117

```

```

00121     ClassList_StringList_T _classCodeList;
00122 };
00123
00124 }
00125 #endif // __STDAIR_BOM_AIRLINECLASSLISTKEY_HPP

```

35.125 stdair/bom/AirlineClassListTypes.hpp File Reference

```

#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>

```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

Typedefs

- typedef std::list< AirlineClassList * > [stdair::AirlineClassListList_T](#)
- typedef std::map< const MapKey_T, AirlineClassList * > [stdair::AirlineClassListMap_T](#)
- typedef std::pair< MapKey_T, AirlineClassList * > [stdair::AirlineClassListWithKey_T](#)
- typedef std::list< AirlineClassListWithKey_T > [stdair::AirlineClassListDetailedList_T](#)

35.126 AirlineClassListTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_AIRLINECLASSLISTTYPES_HPP
00003 #define __STDAIR_BOM_AIRLINECLASSLISTTYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // STDAIR
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016     // Forward declarations.
00017     class AirlineClassList;
00018
00020     typedef std::list<AirlineClassList*> AirlineClassListList_T;
00021
00023     typedef std::map<const MapKey_T, AirlineClassList*> AirlineClassListMap_T;

```

```

00024
00026     typedef std::pair<MapKey_T, AirlineClassList*> AirlineClassListWithKey_T;
00027     typedef std::list<AirlineClassListWithKey_T> AirlineClassListDetailedList_T;
00028 }
00029 #endif // __STDAIR_BOM_AIRLINECLASSLISTTYPES_HPP

```

35.127 stdair/bom/AirlineFeature.cpp File Reference

```

#include <cassert>

#include <stdair/stdair_types.hpp>

#include <stdair/bom/AirlineFeature.hpp>

```

Namespaces

- namespace `stdair`
Handle on the `StdAir` library context.

35.128 AirlineFeature.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // StdAir
00007 #include <stdair/stdair_types.hpp>
00008 #include <stdair/bom/AirlineFeature.hpp>
00009
00010 namespace stdair {
00011
00012 // //////////////////////////////////////
00013 AirlineFeature::AirlineFeature (const Key_T& iKey) : _key (iKey) {
00014 }
00015
00016 // //////////////////////////////////////
00017 AirlineFeature::~AirlineFeature () {
00018 }
00019
00020 // //////////////////////////////////////
00021 void AirlineFeature::init (const ForecasterMode_T& iForecastMode,
00022                          const HistoricalDataLimit_T& iHistoricalDataLimit,
00023                          const ControlMode_T& iControlMode) {
00024     _forecasterMode = iForecastMode;
00025     _historicalDataLimit = iHistoricalDataLimit;
00026     _controlMode = iControlMode;
00027 }
00028
00029 // //////////////////////////////////////
00030 std::string AirlineFeature::toString() const {
00031     std::ostringstream ostr;
00032     ostr << describeKey()
00033         << ", " << _forecasterMode
00034         << ", " << _historicalDataLimit
00035         << ", " << _controlMode;

```

```

00036     return ostr.str();
00037 }
00038
00039 }
00040

```

35.129 stdair/bom/AirlineFeature.hpp File Reference

```

#include <stdair/stdair_rm_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/AirlineFeatureKey.hpp>
#include <stdair/bom/AirlineFeatureTypes.hpp>

```

Classes

- class [stdair::AirlineFeature](#)

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.130 AirlineFeature.hpp

```

00001 #ifndef __STDAIR_BOM_AIRLINEFEATURE_HPP
00002 #define __STDAIR_BOM_AIRLINEFEATURE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_rm_types.hpp>
00009 #include <stdair/bom/BomAbstract.hpp>
00010 #include <stdair/bom/AirlineFeatureKey.hpp>
00011 #include <stdair/bom/AirlineFeatureTypes.hpp>
00012
00013 namespace stdair {
00014
00016     class AirlineFeature : public BomAbstract {
00017     template <typename BOM> friend class FacBom;
00018
00019     public:
00020         // Type definitions.
00022         typedef AirlineFeatureKey Key_T;
00023
00024     public:
00025         // //////////// Getters ////////////
00027         const Key_T& getKey() const {
00028             return _key;
00029         }
00030

```

```

00031 public:
00032     // //////////// Setters ////////////
00034     void init (const ForecasterMode_T&, const HistoricalDataLimit_T&,
00035               const ControlMode_T&);
00036
00037 public:
00038     // //////////// Display support methods ////////////
00041     void toStream (std::ostream& ioOut) const { ioOut << toString(); }
00042
00045     void fromStream (std::istream& ioIn) { }
00046
00048     std::string toString() const;
00049
00051     const std::string describeKey() const { return _key.toString(); }
00052
00053 protected:
00055     AirlineFeature ();
00056     AirlineFeature (const AirlineFeature&);
00057     AirlineFeature (const Key_T&);
00059     virtual ~AirlineFeature();
00060
00061 protected:
00062     // Attributes
00064     Key_T _key;
00065
00067     ForecasterMode_T _forecasterMode;
00068
00070     HistoricalDataLimit_T _historicalDataLimit;
00071
00073     ControlMode_T _controlMode;
00074 };
00075
00076 }
00077 #endif // __STDAIR_BOM_AIRLINEFEATURE_HPP
00078

```

35.131 stdair/bom/AirlineFeatureKey.cpp File Reference

```

#include <sstream>

#include <stdair/bom/AirlineFeatureKey.hpp>

```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.132 AirlineFeatureKey.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <sstream>
00006 // StdAir
00007 #include <stdair/bom/AirlineFeatureKey.hpp>

```

```

00008
00009 namespace stdair {
00010
00011 // //////////////////////////////////////
00012 AirlineFeatureKey::AirlineFeatureKey (const AirlineCode_T& iAirlineCode)
00013 : _airlineCode (iAirlineCode) {
00014 }
00015
00016 // //////////////////////////////////////
00017 AirlineFeatureKey::~AirlineFeatureKey () {
00018 }
00019
00020 // //////////////////////////////////////
00021 void AirlineFeatureKey::toStream (std::ostream& ioOut) const {
00022     ioOut << "AirlineFeatureKey: " << toString() << std::endl;
00023 }
00024
00025 // //////////////////////////////////////
00026 void AirlineFeatureKey::fromStream (std::istream& ioIn) {
00027 }
00028
00029 // //////////////////////////////////////
00030 const std::string AirlineFeatureKey::toString() const {
00031     std::ostringstream oStr;
00032     oStr << _airlineCode;
00033     return oStr.str();
00034 }
00035
00036 }

```

35.133 stdair/bom/AirlineFeatureKey.hpp File Reference

```

#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>

```

Classes

- struct [stdair::AirlineFeatureKey](#)

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.134 AirlineFeatureKey.hpp

```

00001 #ifndef __STDAIR_BOM_AIRLINEFEATUREKEY_HPP
00002 #define __STDAIR_BOM_AIRLINEFEATUREKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section

```

```

00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/stdair_inventory_types.hpp>
00011 #include <stdair/bom/KeyAbstract.hpp>
00012
00013 namespace stdair {
00015     struct AirlineFeatureKey : public KeyAbstract {
00016
00017     public:
00018         // ////////////////////////////////// Construction //////////////////////////////////
00020         AirlineFeatureKey (const AirlineCode_T& iAirlineCode);
00021
00023         ~AirlineFeatureKey ();
00024
00025         // ////////////////////////////////// Getters //////////////////////////////////
00027         const AirlineCode_T& getAirlineCode() const { return _airlineCode; }
00028
00029         // ////////////////////////////////// Display support methods //////////////////////////////////
00032         void toStream (std::ostream& ioOut) const;
00033
00036         void fromStream (std::istream& ioIn);
00037
00043         const std::string toString() const;
00044
00045     private:
00046         // Attributes
00048         AirlineCode_T _airlineCode;
00049     };
00050
00051 }
00052 #endif // __STDAIR_BOM_AIRLINEFEATUREKEY_HPP

```

35.135 stdair/bom/AirlineFeatureTypes.hpp File Reference

```

#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>

```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

Typedefs

- typedef std::list< AirlineFeature * > [stdair::AirlineFeatureList_T](#)
- typedef std::map< const MapKey_T, AirlineFeature * > [stdair::AirlineFeatureMap_T](#)

35.136 AirlineFeatureTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_AIRLINEFEATURETYPES_HPP
00003 #define __STDAIR_BOM_AIRLINEFEATURETYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // StdAir
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016     // Forward declarations.
00017     class AirlineFeature;
00018
00019     typedef std::list<AirlineFeature*> AirlineFeatureList_T;
00020
00021     typedef std::map<const MapKey_T, AirlineFeature*> AirlineFeatureMap_T;
00022
00023 }
00024
00025 #endif // __STDAIR_BOM_AIRLINEFEATURETYPES_HPP
00026
00027

```

35.137 stdair/bom/AirlineStruct.cpp File Reference

```

#include <cassert>
#include <istream>
#include <ostream>
#include <sstream>
#include <stdair/bom/AirlineStruct.hpp>

```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

35.138 AirlineStruct.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <istream>
00007 #include <ostream>
00008 #include <sstream>
00009 // StdAir

```

```

00010 #include <stdair/bom/AirlineStruct.hpp>
00011
00012 namespace stdair {
00013
00014 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00015 AirlineStruct::AirlineStruct () {
00016 }
00017
00018 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00019 AirlineStruct::AirlineStruct (const AirlineStruct& iAirlineStruct)
00020 : _code (iAirlineStruct._code), _name (iAirlineStruct._name) {
00021 }
00022
00023 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00024 AirlineStruct::AirlineStruct (const AirlineCode_T& iAirlineCode,
00025                               const std::string& iAirlineName)
00026 : _code (iAirlineCode), _name (iAirlineName) {
00027 }
00028
00029 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00030 AirlineStruct::~AirlineStruct () {
00031 }
00032
00033 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00034 void AirlineStruct::toStream (std::ostream& ioOut) const {
00035     ioOut << describe();
00036 }
00037
00038 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00039 void AirlineStruct::fromStream (std::istream& ioIn) {
00040 }
00041
00042 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00043 const std::string AirlineStruct::describe() const {
00044     std::ostringstream oStr;
00045     oStr << _code << " " << _name;
00046     return oStr.str();
00047 }
00048
00049 }

```

35.139 stdair/bom/AirlineStruct.hpp File Reference

```

#include <iosfwd>

#include <string>

#include <vector>

#include <stdair/stdair_inventory_types.hpp>

#include <stdair/basic/StructAbstract.hpp>

```

Classes

- struct [stdair::AirlineStruct](#)

Namespaces

- namespace `stdair`

Handle on the StdAir library context.

35.140 AirlineStruct.hpp

```

00001 #ifndef __STDAIR_BOM_AIRLINESTRUCT_HPP
00002 #define __STDAIR_BOM_AIRLINESTRUCT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 #include <vector>
00011 // StdAir
00012 #include <stdair/stdair_inventory_types.hpp>
00013 #include <stdair/basic/StructAbstract.hpp>
00014
00015 namespace stdair {
00016
00017     struct AirlineStruct : public StructAbstract {
00018     public:
00019         // ////////////////////////////////// Getters //////////////////////////////////
00020         const AirlineCode_T& getAirlineCode() const {
00021             return _code;
00022         }
00023
00024         const std::string& getAirlineName() const {
00025             return _name;
00026         }
00027
00028         // ////////////////////////////////// Setters //////////////////////////////////
00029         void setAirlineCode (const AirlineCode_T& iAirlineCode) {
00030             _code = iAirlineCode;
00031         }
00032
00033         void setAirlineName (const std::string& iAirlineName) {
00034             _name = iAirlineName;
00035         }
00036
00037     public:
00038         // ////////////////////////////////// Display support method //////////////////////////////////
00039         void toStream (std::ostream& ioOut) const;
00040
00041         void fromStream (std::istream& ioIn);
00042
00043         const std::string describe() const;
00044
00045     public:
00046         // ////////////////////////////////// Constructors & Destructor //////////////////////////////////
00047         AirlineStruct (const AirlineCode_T&, const std::string& iAirlineName);
00048         AirlineStruct ();
00049         AirlineStruct (const AirlineStruct&);
00050         ~AirlineStruct ();
00051
00052
00053
00054
00055
00056
00057
00058
00059
00060
00061
00062
00063
00064
00065
00066
00067

```

```

00068
00069     private:
00070         // ////////////////////////////////// Attributes //////////////////////////////////
00072         AirlineCode_T _code;
00073
00075         std::string _name;
00076     };
00077
00078 }
00079 #endif // __STDAIR_BOM_AIRLINESTRUCT_HPP

```

35.141 stdair/bom/AirportPair.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/AirportPair.hpp>

```

Namespaces

- namespace `stdair`

Handle on the StdAir library context.

35.142 AirportPair.cpp

```

00001 // //////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_Inventory.hpp>
00009 #include <stdair/service/Logger.hpp>
00010 #include <stdair/bom/AirportPair.hpp>
00011
00012 namespace stdair {
00013
00014     // //////////////////////////////////
00015     AirportPair::AirportPair()
00016         : _key (DEFAULT_ORIGIN, DEFAULT_DESTINATION),
00017         _parent (NULL) {
00018         // That constructor is used by the serialisation process
00019     }
00020
00021     // //////////////////////////////////
00022     AirportPair::AirportPair (const AirportPair& iAirportPair)
00023         : _key (iAirportPair.getKey()), _parent (NULL) {
00024         assert (false);
00025     }
00026

```

```

00027 // //////////////////////////////////////
00028 AirportPair::AirportPair (const Key_T& iKey)
00029     : _key (iKey), _parent (NULL)  {
00030 }
00031
00032 // //////////////////////////////////////
00033 AirportPair::~AirportPair () {
00034 }
00035
00036 // //////////////////////////////////////
00037 std::string AirportPair::toString() const {
00038     std::ostringstream ostr;
00039     ostr << describeKey();
00040     return ostr.str();
00041 }
00042 }
00043
00044

```

35.143 stdair/bom/AirportPair.hpp File Reference

```

#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/AirportPairKey.hpp>
#include <stdair/bom/AirportPairTypes.hpp>

```

Classes

- class [stdair::AirportPair](#)
Class representing the actual attributes for an airport-pair.

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.144 AirportPair.hpp

```

00001 #ifndef __STDAIR_BOM_AIRPORTPAIR_HPP
00002 #define __STDAIR_BOM_AIRPORTPAIR_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STDAIR
00008 #include <stdair/bom/BomAbstract.hpp>
00009 #include <stdair/bom/AirportPairKey.hpp>
00010 #include <stdair/bom/AirportPairTypes.hpp>
00011
00012 // Forward declaration
00013 namespace stdair {
00014
00018     class AirportPair : public BomAbstract {

```

```

00019     template <typename BOM> friend class FacBom;
00020     friend class FacBomManager;
00021
00022 public:
00023     // //////////// Type definitions ////////////
00027     typedef AirportPairKey Key_T;
00028
00029 public:
00030     // //////////// Display support methods ////////////
00036     void toStream (std::ostream& ioOut) const {
00037         ioOut << toString();
00038     }
00039
00045     void fromStream (std::istream& ioIn) {
00046     }
00047
00051     std::string toString() const;
00052
00056     const std::string describeKey() const {
00057         return _key.toString();
00058     }
00059
00060 public:
00061     // //////////// Getters ////////////
00065     const Key_T& getKey() const {
00066         return _key;
00067     }
00068
00072     const AirportCode_T& getBoardingPoint() const {
00073         return _key.getBoardingPoint();
00074     }
00075
00079     const AirportCode_T& getOffPoint() const {
00080         return _key.getOffPoint();
00081     }
00082
00086     BomAbstract* const getParent() const {
00087         return _parent;
00088     }
00089
00093     const HolderMap_T& getHolderMap() const {
00094         return _holderMap;
00095     }
00096
00097 protected:
00098     // //////////// Constructors and destructors ////////////
00102     AirportPair (const Key_T&);
00106     virtual ~AirportPair();
00107
00108 private:
00112     AirportPair();
00116     AirportPair (const AirportPair&);
00117
00118 protected:
00119     // //////////// Attributes ////////////
00123     Key_T _key;
00124
00128     BomAbstract* _parent;
00129
00133     HolderMap_T _holderMap;
00134
00135 };

```

```

00136
00137 }
00138 #endif // __STDAIR_BOM_AIRPORTPAIR_HPP
00139

```

35.145 stdair/bom/AirportPairKey.cpp File Reference

```

#include <ostream>
#include <sstream>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/AirportPairKey.hpp>

```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

35.146 AirportPairKey.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <ostream>
00006 #include <sstream>
00007 // STDAIR
00008 #include <stdair/basic/BasConst_BomDisplay.hpp>
00009 #include <stdair/basic/BasConst_Inventory.hpp>
00010 #include <stdair/bom/AirportPairKey.hpp>
00011
00012 namespace stdair {
00013
00014 // //////////////////////////////////////
00015 AirportPairKey::AirportPairKey ()
00016 : _boardingPoint (DEFAULT_ORIGIN),
00017   _offPoint (DEFAULT_DESTINATION) {
00018     assert (false);
00019 }
00020
00021 // //////////////////////////////////////
00022 AirportPairKey::AirportPairKey (const AirportCode_T& iBoardingPoint,
00023                                 const AirportCode_T& iOffPoint)
00024 : _boardingPoint (iBoardingPoint), _offPoint (iOffPoint) {
00025 }
00026
00027 // //////////////////////////////////////
00028 AirportPairKey::AirportPairKey (const AirportPairKey& iKey)
00029 : _boardingPoint (iKey._boardingPoint),
00030   _offPoint (iKey._offPoint) {
00031 }
00032

```

```

00033 // //////////////////////////////////////
00034 AirportPairKey::~AirportPairKey () {
00035 }
00036
00037 // //////////////////////////////////////
00038 void AirportPairKey::toStream (std::ostream& ioOut) const {
00039     ioOut << "AirportPairKey: " << toString() << std::endl;
00040 }
00041
00042 // //////////////////////////////////////
00043 void AirportPairKey::fromStream (std::istream& ioIn) {
00044 }
00045
00046 // //////////////////////////////////////
00047 const std::string AirportPairKey::toString() const {
00048     std::ostringstream oStr;
00049     oStr << _boardingPoint << DEFAULT_KEY_SUB_FLD_DELIMITER
00050     << " " << _offPoint;
00051     return oStr.str();
00052 }
00053
00054 }

```

35.147 stdair/bom/AirportPairKey.hpp File Reference

```

#include <stdair/bom/KeyAbstract.hpp>
#include <stdair/stdair_basic_types.hpp>

```

Classes

- struct [stdair::AirportPairKey](#)
Key of airport-pair.

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.148 AirportPairKey.hpp

```

00001 #ifndef __STDAIR_BOM_AIRPORTPAIRKEY_HPP
00002 #define __STDAIR_BOM_AIRPORTPAIRKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STDAIR
00008 #include <stdair/bom/KeyAbstract.hpp>
00009 #include <stdair/stdair_basic_types.hpp>
00010
00011 namespace stdair {
00012

```

```

00016 struct AirportPairKey : public KeyAbstract {
00017
00018 public:
00019     // ////////// Construction //////////
00021     AirportPairKey (const stdair::AirportCode_T&,
00022                    const stdair::AirportCode_T&);
00024     AirportPairKey (const AirportPairKey&);
00026     ~AirportPairKey ();
00027 private:
00029     AirportPairKey ();
00030
00031 public:
00032     // ////////// Getters //////////
00036     const stdair::AirportCode_T& getBoardingPoint() const {
00037         return _boardingPoint;
00038     }
00039
00043     const stdair::AirportCode_T& getOffPoint() const {
00044         return _offPoint;
00045     }
00046
00047     // ////////// Display support methods //////////
00053     void toStream (std::ostream& ioOut) const;
00054
00060     void fromStream (std::istream& ioIn);
00061
00067     const std::string toString() const;
00068
00069 private:
00070     // ////////////////////////////////// Attributes //////////////////////////////////
00074     AirportCode_T _boardingPoint;
00075
00079     AirportCode_T _offPoint;
00080 };
00081
00082 }
00083 #endif // __SIMFQT_BOM_AIRPORTPAIRKEY_HPP

```

35.149 stdair/bom/AirportPairTypes.hpp File Reference

```

#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>

```

Namespaces

- namespace `stdair`
Handle on the *StdAir* library context.

Typedefs

- typedef `std::list< AirportPair * >` `stdair::AirportPairList_T`
- typedef `std::map< const MapKey_T, AirportPair * >` `stdair::AirportPairMap_T`
- typedef `std::pair< MapKey_T, AirportPair * >` `stdair::AirportPairWithKey_T`

- typedef std::list< AirportPairWithKey_T > [stdair::AirportPairDetailedList_T](#)

35.150 AirportPairTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_AIRPORTPAIRTYPES_HPP
00003 #define __STDAIR_BOM_AIRPORTPAIRTYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // STDAIR
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016     // Forward declarations.
00017     class AirportPair;
00018
00019     typedef std::list<AirportPair*> AirportPairList_T;
00020
00021     typedef std::map<const MapKey_T, AirportPair*> AirportPairMap_T;
00022
00023     typedef std::pair<MapKey_T, AirportPair*> AirportPairWithKey_T;
00024     typedef std::list<AirportPairWithKey_T> AirportPairDetailedList_T;
00025 }
00026
00027 #endif // __STDAIR_BOM_AIRPORTPAIRTYPES_HPP
00028
00029
00030

```

35.151 stdair/bom/BomAbstract.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <map>
#include <typeinfo>

```

Classes

- class [stdair::BomAbstract](#)
Base class for the Business Object Model (BOM) layer.

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

Typedefs

- typedef std::map< const std::type_info *, BomAbstract * > [stdair::HolderMap_T](#)

Functions

- template<class charT , class traits >
std::basic_ostream< charT, traits > & [operator<<](#) (std::basic_ostream< charT, traits > &ioOut, const [stdair::BomAbstract](#) &iBom)
- template<class charT , class traits >
std::basic_istream< charT, traits > & [operator>>](#) (std::basic_istream< charT, traits > &ioln, [stdair::BomAbstract](#) &iBom)

35.151.1 Function Documentation

35.151.1.1 template<class charT , class traits > std::basic_ostream<charT, traits>&
operator<< (std::basic_ostream< charT, traits > & *ioOut*, const
[stdair::BomAbstract](#) & *iBom*) [inline]

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (p653) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 59 of file [BomAbstract.hpp](#).

References [stdair::BomAbstract::toStream\(\)](#).

35.151.1.2 template<class charT , class traits > std::basic_istream<charT, traits>&
operator>> (std::basic_istream< charT, traits > & *ioln*, [stdair::BomAbstract](#)
& *ioBom*) [inline]

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (pp655-657) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 87 of file [BomAbstract.hpp](#).

References [stdair::BomAbstract::fromStream\(\)](#).

35.152 BomAbstract.hpp

```
00001
00006 #ifndef __STDAIR_BOM_BOMABSTRACT_HPP
00007 #define __STDAIR_BOM_BOMABSTRACT_HPP
00008
00009 // //////////////////////////////////////
00010 // Import section
00011 // //////////////////////////////////////
00012 // STL
00013 #include <iosfwd>
00014 #include <string>
00015 #include <map>
```

```

00016 #include <typeinfo>
00017
00018 namespace stdair {
00019
00023     class BomAbstract {
00024     public:
00025         // //////////// Display support methods ////////////
00028         virtual void toStream (std::ostream& ioOut) const = 0;
00029
00032         virtual void fromStream (std::istream& ioIn) = 0;
00033
00035         virtual std::string toString() const = 0;
00036
00037
00038     protected:
00040         BomAbstract() {}
00041         BomAbstract(const BomAbstract&) {}
00042     public:
00044         virtual ~BomAbstract() {}
00045     };
00046
00047     /* Define the map of object holder type. */
00048     typedef std::map<const std::type_info*, BomAbstract*> HolderMap_T;
00049 }
00050
00056 template <class charT, class traits>
00057 inline
00058 std::basic_ostream<charT, traits>&
00059 operator<< (std::basic_ostream<charT, traits>& ioOut,
00060           const stdair::BomAbstract& iBom) {
00066     std::basic_ostringstream<charT, traits> ostr;
00067     ostr.copyfmt (ioOut);
00068     ostr.width (0);
00069
00070     // Fill string stream
00071     iBom.toStream (ostr);
00072
00073     // Print string stream
00074     ioOut << ostr.str();
00075
00076     return ioOut;
00077 }
00078
00084 template <class charT, class traits>
00085 inline
00086 std::basic_istream<charT, traits>&
00087 operator>> (std::basic_istream<charT, traits>& ioIn,
00088           stdair::BomAbstract& ioBom) {
00089     // Fill Bom object with input stream
00090     ioBom.fromStream (ioIn);
00091     return ioIn;
00092 }
00093
00094 #endif // __STDAIR_BOM_BOMABSTRACT_HPP

```

35.153 stdair/bom/BomArchive.cpp File Reference

```

#include <cassert>

#include <sstream>

```

```

#include <boost/archive/tmpdir.hpp>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/base_object.hpp>
#include <boost/serialization/utility.hpp>
#include <boost/serialization/list.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/BookingRequestStruct.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomArchive.hpp>

```

Namespaces

- namespace `stdair`

Handle on the StdAir library context.

35.154 BomArchive.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/tmpdir.hpp>
00009 #include <boost/archive/text_iarchive.hpp>
00010 #include <boost/archive/text_oarchive.hpp>
00011 #include <boost/serialization/base_object.hpp>
00012 #include <boost/serialization/utility.hpp>
00013 #include <boost/serialization/list.hpp>
00014 // #include <boost/serialization/assume_abstract.hpp>
00015 // StdAir
00016 #include <stdair/bom/BomRoot.hpp>
00017 #include <stdair/bom/Inventory.hpp>

```

```

00018 #include <stdair/bom/FlightDate.hpp>
00019 #include <stdair/bom/LegDate.hpp>
00020 #include <stdair/bom/SegmentDate.hpp>
00021 #include <stdair/bom/LegCabin.hpp>
00022 #include <stdair/bom/SegmentCabin.hpp>
00023 #include <stdair/bom/FareFamily.hpp>
00024 #include <stdair/bom/BookingClass.hpp>
00025 #include <stdair/bom/BookingRequestStruct.hpp>
00026 #include <stdair/bom/BomManager.hpp>
00027 #include <stdair/bom/BomArchive.hpp>
00028
00029 namespace stdair {
00030
00031 // //////////////////////////////////////
00032 void BomArchive::archive (const BomRoot& iBomRoot) {
00033 }
00034
00035 // //////////////////////////////////////
00036 std::string BomArchive::archive (const Inventory& iInventory) {
00037     std::ostringstream oStr;
00038     boost::archive::text_oarchive oa (oStr);
00039     oa << iInventory;
00040     return oStr.str();
00041 }
00042
00043 // //////////////////////////////////////
00044 void BomArchive::restore (const std::string& iArchive,
00045                          Inventory& ioInventory) {
00046     std::istringstream iStr;
00047     boost::archive::text_iarchive ia (iStr);
00048     ia >> ioInventory;
00049 }
00050
00051 // //////////////////////////////////////
00052 void BomArchive::archive (const FlightDate& iFlightDate) {
00053 }
00054
00055 }

```

35.155 stdair/bom/BomArchive.hpp File Reference

```
#include <iosfwd>
```

Classes

- class [stdair::BomArchive](#)
Utility class to archive/restore BOM objects with Boost serialisation.

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.156 BomArchive.hpp

```

00001 #ifndef __STDAIR_BOM_BOMARCHIVE_HPP
00002 #define __STDAIR_BOM_BOMARCHIVE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009
00010 namespace stdair {
00011
00012     class BomRoot;
00013     class Inventory;
00014     class FlightDate;
00015     class LegDate;
00016     class SegmentDate;
00017     class LegCabin;
00018     class SegmentCabin;
00019     class FareFamily;
00020     class BookingClass;
00021     struct BookingRequestStruct;
00022
00023     class BomArchive {
00024     public:
00025         static void archive (const BomRoot&);
00026
00027         static std::string archive (const Inventory&);
00028
00029         static void restore (const std::string& iArchive, Inventory&);
00030
00031         static void archive (const FlightDate&);
00032     };
00033 }
00034 #endif // __STDAIR_BOM_BOMARCHIVE_HPP

```

35.157 stdair/bom/BomDisplay.cpp File Reference

35.158 BomDisplay.cpp

```

00001
00002 // //////////////////////////////////////
00003 // Import section
00004 // //////////////////////////////////////
00005 // STL
00006 #include <cassert>
00007 #include <ostream>
00008 // StdAir
00009 #include <stdair/basic/BasConst_BomDisplay.hpp>
00010 #include <stdair/bom/BomManager.hpp>
00011 #include <stdair/bom/BomRoot.hpp>
00012 #include <stdair/bom/EventQueue.hpp>
00013 #include <stdair/bom/Inventory.hpp>
00014 #include <stdair/bom/FlightDate.hpp>
00015 #include <stdair/bom/LegDate.hpp>
00016 #include <stdair/bom/SegmentDate.hpp>
00017 #include <stdair/bom/LegCabin.hpp>
00018

```

```

00021 #include <stdair/bom/SegmentCabin.hpp>
00022 #include <stdair/bom/FareFamily.hpp>
00023 #include <stdair/bom/BookingClass.hpp>
00024 #include <stdair/bom/AirportPair.hpp>
00025 #include <stdair/bom/PosChannel.hpp>
00026 #include <stdair/bom/DatePeriod.hpp>
00027 #include <stdair/bom/TimePeriod.hpp>
00028 #include <stdair/bom/FareFeatures.hpp>
00029 #include <stdair/bom/YieldFeatures.hpp>
00030 #include <stdair/bom/AirlineClassList.hpp>
00031 #include <stdair/bom/Bucket.hpp>
00032 #include <stdair/bom/TravelSolutionTypes.hpp>
00033 #include <stdair/bom/TravelSolutionStruct.hpp>
00034 #include <stdair/bom/BomDisplay.hpp>
00035 #include <stdair/bom/OnDDate.hpp>
00036
00037 namespace stdair {
00038
00044     struct FlagSaver {
00045     public:
00047         FlagSaver (std::ostream& oStream)
00048             : _oStream (oStream), _streamFlags (oStream.flags()) {
00049         }
00050
00052         ~FlagSaver() {
00053             // Reset formatting flags of the given output stream
00054             _oStream.flags (_streamFlags);
00055         }
00056
00057     private:
00059         std::ostream& _oStream;
00061         std::ios::fmtflags _streamFlags;
00062     };
00063
00064     // //////////////////////////////////////
00065     std::string BomDisplay::csvDisplay (const EventQueue& iEventQueue) {
00066         std::ostringstream oStream;
00067
00071         oStream << std::endl;
00072         oStream << "=====
00073         << std::endl;
00074         oStream << "EventQueue: " << iEventQueue.describeKey() << std::endl;
00075         oStream << "=====
00076         << std::endl;
00077
00078         return oStream.str();
00079     }
00080
00081     // //////////////////////////////////////
00082     void BomDisplay::list (std::ostream& oStream, const BomRoot& iBomRoot,
00083                          const AirlineCode_T& iAirlineCode,
00084                          const FlightNumber_T& iFlightNumber) {
00085         // Save the formatting flags for the given STL output stream
00086         FlagSaver flagSaver (oStream);
00087
00088         // Check whether there are Inventory objects
00089         if (BomManager::hasList<Inventory> (iBomRoot) == false) {
00090             return;
00091         }
00092
00093         // Browse the inventories
00094         unsigned short invIdx = 1;

```

```

00095     const InventoryList_T& lInventoryList =
00096         BomManager::getList<Inventory> (iBomRoot);
00097     for (InventoryList_T::const_iterator itInv = lInventoryList.begin();
00098         itInv != lInventoryList.end(); ++itInv, ++invIdx) {
00099         const Inventory* lInv_ptr = *itInv;
00100         assert (lInv_ptr != NULL);
00101
00102         // Retrieve the inventory key (airline code)
00103         const AirlineCode_T& lAirlineCode = lInv_ptr->getAirlineCode();
00104
00105         // Display only the requested inventories
00106         if (iAirlineCode == "all" || iAirlineCode == lAirlineCode) {
00107             // Get the list of flight-dates for that inventory
00108             list (oStream, *lInv_ptr, invIdx, iFlightNumber);
00109         }
00110     }
00111 }
00112
00113 ///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00114 void BomDisplay::list (std::ostream& oStream, const Inventory& iInventory,
00115                     const unsigned short iInventoryIndex,
00116                     const FlightNumber_T& iFlightNumber) {
00117     // Save the formatting flags for the given STL output stream
00118     FlagSaver flagSaver (oStream);
00119
00120     // Check whether there are FlightDate objects
00121     if (BomManager::hasMap<FlightDate> (iInventory) == false) {
00122         return;
00123     }
00124
00125     //
00126     const AirlineCode_T& lAirlineCode = iInventory.getAirlineCode();
00127     oStream << iInventoryIndex << ". " << lAirlineCode << std::endl;
00128
00129     // Browse the flight-dates
00130     unsigned short lCurrentFlightNumber = 0;
00131     unsigned short flightNumberIdx = 0;
00132     unsigned short departureDateIdx = 1;
00133     const FlightDateMap_T& lFlightDateList =
00134         BomManager::getMap<FlightDate> (iInventory);
00135     for (FlightDateMap_T::const_iterator itFD = lFlightDateList.begin();
00136         itFD != lFlightDateList.end(); ++itFD, ++departureDateIdx) {
00137         const FlightDate* lFD_ptr = itFD->second;
00138         assert (lFD_ptr != NULL);
00139
00140         // Retrieve the key of the flight-date
00141         const FlightNumber_T& lFlightNumber = lFD_ptr->getFlightNumber();
00142         const Date_T& lFlightDateDate = lFD_ptr->getDepartureDate();
00143
00144         // Display only the requested flight number
00145         if (iFlightNumber == 0 || iFlightNumber == lFlightNumber) {
00146             //
00147             if (lCurrentFlightNumber != lFlightNumber) {
00148                 lCurrentFlightNumber = lFlightNumber;
00149                 ++flightNumberIdx; departureDateIdx = 1;
00150                 oStream << " " << iInventoryIndex << "." << flightNumberIdx << ". "
00151                     << lAirlineCode << lFlightNumber << std::endl;
00152             }
00153
00154             oStream << " " << iInventoryIndex << "." << flightNumberIdx
00155                 << "." << departureDateIdx << ". "
00156                 << lAirlineCode << lFlightNumber << " / " << lFlightDateDate

```

```

00165         << std::endl;
00166     }
00167 }
00168 }
00169
00170 // //////////////////////////////////////
00171 void BomDisplay::listAirportPairDateRange (std::ostream& oStream,
00172                                           const BomRoot& iBomRoot) {
00173     // Save the formatting flags for the given STL output stream
00174     FlagSaver flagSaver (oStream);
00175
00176     // Check whether there are AirportPair objects
00177     if (BomManager::hasList<AirportPair> (iBomRoot) == false) {
00178         return;
00179     }
00180
00181     const AirportPairList_T& lAirportPairList =
00182         BomManager::getList<AirportPair> (iBomRoot);
00183     for (AirportPairList_T::const_iterator itAir = lAirportPairList.begin();
00184          itAir != lAirportPairList.end(); ++itAir) {
00185         const AirportPair* lAir_ptr = *itAir;
00186         assert (lAir_ptr != NULL);
00187
00188         // Check whether there are date-period objects
00189         assert (BomManager::hasList<DatePeriod> (*lAir_ptr) == true);
00190
00191         // Browse the date-period objects
00192         const DatePeriodList_T& lDatePeriodList =
00193             BomManager::getList<DatePeriod> (*lAir_ptr);
00194
00195         for (DatePeriodList_T::const_iterator itDP = lDatePeriodList.begin();
00196              itDP != lDatePeriodList.end(); ++itDP) {
00197             const DatePeriod* lDP_ptr = *itDP;
00198             assert (lDP_ptr != NULL);
00199
00200             // Display the date-period object
00201             oStream << lAir_ptr->describeKey()
00202                 << " / " << lDP_ptr->describeKey() << std::endl;
00203         }
00204     }
00205 }
00206 }
00207
00208 // //////////////////////////////////////
00209 void BomDisplay::csvDisplay (std::ostream& oStream,
00210                             const BomRoot& iBomRoot) {
00211     // Save the formatting flags for the given STL output stream
00212     FlagSaver flagSaver (oStream);
00213
00214     oStream << std::endl;
00215     oStream << "===== "
00216         << std::endl;
00217     oStream << "BomRoot: " << iBomRoot.describeKey() << std::endl;
00218     oStream << "===== "
00219         << std::endl;
00220
00221     // Check whether there are Inventory objects
00222     if (BomManager::hasList<Inventory> (iBomRoot) == false) {
00223         return;
00224     }
00225
00226     // Browse the inventories

```

```

00230     const InventoryList_T& lInventoryList =
00231         BomManager::getList<Inventory> (iBomRoot);
00232     for (InventoryList_T::const_iterator itInv = lInventoryList.begin();
00233         itInv != lInventoryList.end(); ++itInv) {
00234         const Inventory* lInv_ptr = *itInv;
00235         assert (lInv_ptr != NULL);
00236
00237         // Display the inventory
00238         csvDisplay (oStream, *lInv_ptr);
00239     }
00240 }
00241
00242 // //////////////////////////////////////
00243 void BomDisplay::csvDisplay (std::ostream& oStream,
00244                             const Inventory& iInventory) {
00245     // Save the formatting flags for the given STL output stream
00246     FlagSaver flagSaver (oStream);
00247
00251     oStream << "+++++" << std::endl;
00252     oStream << "Inventory: " << iInventory.describeKey() << std::endl;
00253     oStream << "+++++" << std::endl;
00254
00255     // Check whether there are FlightDate objects
00256     if (BomManager::hasList<FlightDate> (iInventory) == false) {
00257         return;
00258     }
00259
00260     // Browse the flight-dates
00261     const FlightDateList_T& lFlightDateList =
00262         BomManager::getList<FlightDate> (iInventory);
00263     for (FlightDateList_T::const_iterator itFD = lFlightDateList.begin();
00264         itFD != lFlightDateList.end(); ++itFD) {
00265         const FlightDate* lFD_ptr = *itFD;
00266         assert (lFD_ptr != NULL);
00267
00268         // Display the flight-date
00269         csvDisplay (oStream, *lFD_ptr);
00270     }
00271
00272     // Check if the inventory contains a list of partners
00273     if (BomManager::hasList<Inventory> (iInventory)){
00274
00275         // Browse the partner's inventories
00276         const InventoryList_T& lPartnerInventoryList =
00277             BomManager::getList<Inventory> (iInventory);
00278
00279         for (InventoryList_T::const_iterator itInv = lPartnerInventoryList.begin();
00280
00281             itInv != lPartnerInventoryList.end(); ++itInv) {
00282
00283             oStream << "-----" << std::en
dl;
00284             oStream << "Partner inventory:" << std::endl;
00285             oStream << "-----" << std::en
dl;
00286             const Inventory* lInv_ptr = *itInv;
00287             assert (lInv_ptr != NULL);
00288
00289             // Display the inventory
00290             csvDisplay (oStream, *lInv_ptr);
00291         }

```

```

00292         oStream << "*****" << std::endl;
00293         oStream << std::endl;
00294     }
00295
00296     // Check if the inventory contains a list of O&D dates
00297
00298     if (BomManager::hasList<OnDDate> (iInventory)){
00299
00300         //Browse the O&Ds
00301         const OnDDateList_T& lOnDDateList =
00302             BomManager::getList<OnDDate> (iInventory);
00303
00304         for (OnDDateList_T::const_iterator itOnD = lOnDDateList.begin();
00305             itOnD != lOnDDateList.end(); ++itOnD) {
00306             oStream << "*****" << std::endl;
00307             oStream << "O&D-Date:" << std::endl;
00308             oStream << "-----" << std::endl;
00309             oStream << "Airline, Date, Origin-Destination, Segments, " << std::endl;
00310
00311             const OnDDate* lOnDDate_ptr = *itOnD;
00312             assert (lOnDDate_ptr != NULL);
00313
00314             // Display the O&D date
00315             csvDisplay (oStream, *lOnDDate_ptr);
00316         }
00317         oStream << "*****" << std::endl;
00318     }
00319 }
00320
00321 // //////////////////////////////////////
00322 void BomDisplay::csvDisplay (std::ostream& oStream,
00323                             const OnDDate& iOnDDate) {
00324     // Save the formatting flags for the given STL output stream
00325     FlagSaver flagSaver (oStream);
00326
00327     const AirlineCode_T& lAirlineCode = iOnDDate.getAirlineCode();
00328     const Date_T& lDate = iOnDDate.getDate();
00329     const AirportCode_T& lOrigin = iOnDDate.getOrigin();
00330     const AirportCode_T& lDestination = iOnDDate.getDestination();
00331
00332     oStream << lAirlineCode << ", " << lDate << ", "<< lOrigin << "-"
00333         << lDestination << ", " << iOnDDate.describeKey() << ", "
00334         << std::endl;
00335
00336     const StringDemandStructMap_T& lDemandInfoMap =
00337         iOnDDate.getDemandInfoMap();
00338
00339     // Check if the map contains information.
00340     const bool isInfoMapEmpty = lDemandInfoMap.empty();
00341     if (isInfoMapEmpty) {
00342         return;
00343     }
00344     assert (lDemandInfoMap.empty() ==false);
00345
00346     oStream << "-----" << std::endl;
00347     oStream << "Cabin-Class path, Demand mean, Demand std dev, Yield, "
00348         << std::endl;
00349
00350     for (StringDemandStructMap_T::const_iterator itDI = lDemandInfoMap.begin();
00351         itDI != lDemandInfoMap.end(); ++itDI) {
00352
00353         const std::string& lCabinClassPath = itDI->first;

```

```

00357     const YieldDemandPair_T lYieldDemandPair =
00358         itDI->second;
00359     const Yield_T lYield = lYieldDemandPair.first;
00360     const MeanStdDevPair_T lMeanStdDevPair =
00361         lYieldDemandPair.second;
00362     const MeanValue_T lDemandMean = lMeanStdDevPair.first;
00363     const StdDevValue_T lDemandStdDev = lMeanStdDevPair.second;
00364
00365     ostream << lCabinClassPath << ", "
00366         << lDemandMean << ", "
00367         << lDemandStdDev << ", "
00368         << lYield << ", "
00369         << std::endl;
00370 }
00371
00372 }
00373
00374 // //////////////////////////////////////
00375 void BomDisplay::csvDisplay (std::ostream& ostream,
00376                             const FlightDate& iFlightDate) {
00377     // Save the formatting flags for the given STL output stream
00378     FlagSaver flagSaver (ostream);
00379
00380     const AirlineCode_T& lAirlineCode = iFlightDate.getAirlineCode();
00381     ostream << "*****" << std::endl;
00382     ostream << "FlightDate: " << lAirlineCode << iFlightDate.describeKey()
00383         << std::endl;
00384     ostream << "*****" << std::endl;
00385
00386     //
00387     csvSegmentDateDisplay (ostream, iFlightDate);
00388     //
00389     csvLegDateDisplay (ostream, iFlightDate);
00390
00391     //
00392     csvLegCabinDisplay (ostream, iFlightDate);
00393
00394     //
00395     csvBucketDisplay (ostream, iFlightDate);
00396
00397     //
00398     csvFareFamilyDisplay (ostream, iFlightDate);
00399
00400     //
00401     csvBookingClassDisplay (ostream, iFlightDate);
00402 }
00403
00404 // //////////////////////////////////////
00405 void BomDisplay::csvLegDateDisplay (std::ostream& ostream,
00406                                     const FlightDate& iFlightDate) {
00407     // Save the formatting flags for the given STL output stream
00408     FlagSaver flagSaver (ostream);
00409
00410     ostream << "*****" << std::endl;
00411     ostream << "Leg-Dates:" << std::endl
00412         << "-----" << std::endl;
00413     ostream << "Flight, Leg, BoardDate, BoardTime, "
00414         << "OffDate, OffTime, Date Offset, Time Offset, Elapsed, "
00415         << "Distance, Capacity, " << std::endl;
00416
00417     // Retrieve the key of the flight-date
00418     const AirlineCode_T& lAirlineCode = iFlightDate.getAirlineCode();

```

```

00427     const FlightNumber_T& lFlightNumber = iFlightDate.getFlightNumber();
00428     const Date_T& lFlightDateDate = iFlightDate.getDepartureDate();
00429
00430     // Check whether there are LegDate objects
00431     if (BomManager::hasList<LegDate> (iFlightDate) == false) {
00432         return;
00433     }
00434
00435     // Browse the leg-dates
00436     const LegDateList_T& lLegDateList =
00437         BomManager::getList<LegDate> (iFlightDate);
00438     for (LegDateList_T::const_iterator itLD = lLegDateList.begin();
00439         itLD != lLegDateList.end(); ++itLD) {
00440         const LegDate* lLD_ptr = *itLD;
00441         assert (lLD_ptr != NULL);
00442
00443         oStream << lAirlineCode << lFlightNumber << " "
00444             << lFlightDateDate << ", ";
00445
00446         oStream << lLD_ptr->getBoardingPoint() << "-"
00447             << lLD_ptr->getOffPoint() << ", "
00448             << lLD_ptr->getBoardingDate() << ", "
00449             << lLD_ptr->getBoardingTime() << ", "
00450             << lLD_ptr->getOffDate() << ", "
00451             << lLD_ptr->getOffTime() << ", "
00452             << lLD_ptr->getElapsedTime() << ", "
00453             << lLD_ptr->getDateOffset().days() << ", "
00454             << lLD_ptr->getTimeOffset() << ", "
00455             << lLD_ptr->getDistance() << ", "
00456             << lLD_ptr->getCapacity() << ", " << std::endl;
00457     }
00458     oStream << "*****" << std::endl;
00459 }
00460
00461 // //////////////////////////////////////
00462 void BomDisplay::csvSegmentDateDisplay (std::ostream& oStream,
00463     const FlightDate& iFlightDate) {
00464     // Save the formatting flags for the given STL output stream
00465     FlagSaver flagSaver (oStream);
00466
00467     oStream << "*****" << std::endl;
00468     oStream << "SegmentDates:" << std::endl
00469         << "-----" << std::endl;
00470     oStream << "Flight, Segment, Date"
00471         << std::endl;
00472
00473     // Retrieve the key of the flight-date
00474     const AirlineCode_T& lAirlineCode = iFlightDate.getAirlineCode();
00475     const FlightNumber_T& lFlightNumber = iFlightDate.getFlightNumber();
00476     const Date_T& lFlightDateDate = iFlightDate.getDepartureDate();
00477
00478     // Check whether there are SegmentDate objects
00479     if (BomManager::hasList<SegmentDate> (iFlightDate) == false) {
00480         return;
00481     }
00482
00483     // Browse the segment-dates
00484     const SegmentDateList_T& lSegmentDateList =
00485         BomManager::getList<SegmentDate> (iFlightDate);
00486     for (SegmentDateList_T::const_iterator itSD = lSegmentDateList.begin();
00487         itSD != lSegmentDateList.end(); ++itSD) {
00488         const SegmentDate* lSD_ptr = *itSD;

```

```

00492     assert (lSD_ptr != NULL);
00493
00494     // Retrieve the key of the segment-date, as well as its dates
00495     const Date_T& lSegmentDateDate = lSD_ptr->getBoardingDate();
00496     const AirportCode_T& lBoardPoint = lSD_ptr->getBoardingPoint();
00497     const AirportCode_T& lOffPoint = lSD_ptr->getOffPoint();
00498     ostream << lAirlineCode << lFlightNumber << " " << lFlightDateDate << ", "
00499         << lBoardPoint << "-" << lOffPoint << ", " << lSegmentDateDate << s
td::endl;
00500
00501     // Check if the current segment has corresponding marketing segments.
00502     const bool isMarketingSDListEmpty = BomManager::hasList<SegmentDate>(*lSD_ptr);
00503     if (isMarketingSDListEmpty == false) {
00504         //
00505         const SegmentDateList_T& lMarketingSDList = BomManager::getList<SegmentDate>(*lSD_ptr);
00506
00507         ostream << " *** Marketed by ";
00508         for (SegmentDateList_T::const_iterator itMarketingSD = lMarketingSDList.begin();
00509             itMarketingSD != lMarketingSDList.end(); ++itMarketingSD) {
00510             SegmentDate* lMarketingSD_ptr = *itMarketingSD;
00511             FlightDate* lMarketingFD_ptr = BomManager::getParentPtr<FlightDate>(*lMarketingSD_ptr);
00512             Inventory* lMarketingInv_ptr = BomManager::getParentPtr<Inventory>(*lMarketingFD_ptr);
00513             ostream << lMarketingInv_ptr->toString() << lMarketingFD_ptr->toString() << " * ";
00514         }
00515     }
00516
00517     // Check if the current segment is operated by another segment date.
00518     const SegmentDate* lOperatingSD_ptr = lSD_ptr->getOperatingSegmentDate();
00519     if (lOperatingSD_ptr != NULL) {
00520
00521         const FlightDate* lOperatingFD_ptr = BomManager::getParentPtr<FlightDate>(*lOperatingSD_ptr);
00522         const Inventory* lOperatingInv_ptr = BomManager::getParentPtr<Inventory>(*lOperatingFD_ptr);
00523         ostream << " *** Operated by " << lOperatingInv_ptr->toString() << lOperatingFD_ptr->toString() << std::endl;
00524     }
00525 }
00526
00527 ostream << std::endl;
00528 }
00529 }
00530
00531 // //////////////////////////////////////
00532 void BomDisplay::csvLegCabinDisplay (std::ostream& ostream,
00533     const FlightDate& iFlightDate) {
00534     // Save the formatting flags for the given STL output stream
00535     FlagSaver flagSaver (ostream);
00536
00537     ostream << "*****" << std::endl;
00538     ostream << "LegCabins:" << std::endl;
00539     ostream << "-----" << std::endl;
00540     ostream << "Flight, Leg, Cabin, "
00541         << "OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, "
00542         << "CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice, "
00543         << std::endl;
00544 }

```

```

00548 // Retrieve the key of the flight-date
00549 const AirlineCode_T& lAirlineCode = iFlightDate.getAirlineCode();
00550 const FlightNumber_T& lFlightNumber = iFlightDate.getFlightNumber();
00551 const Date_T& lFlightDateDate = iFlightDate.getDepartureDate();
00552
00553 // Check whether there are LegDate objects
00554 if (BomManager::hasList<LegDate> (iFlightDate) == false) {
00555     return;
00556 }
00557
00558 // Browse the leg-dates
00559 const LegDateList_T& lLegDateList =
00560     BomManager::getList<LegDate> (iFlightDate);
00561 for (LegDateList_T::const_iterator itLD = lLegDateList.begin();
00562      itLD != lLegDateList.end(); ++itLD) {
00563     const LegDate* lLD_ptr = *itLD;
00564     assert (lLD_ptr != NULL);
00565
00566     // Retrieve the key of the leg-date, as well as its off point
00567     const Date_T& lLegDateDate = lLD_ptr->getBoardingDate();
00568     const AirportCode_T& lBoardPoint = lLD_ptr->getBoardingPoint();
00569     const AirportCode_T& lOffPoint = lLD_ptr->getOffPoint();
00570
00571     // Browse the leg-cabins
00572     const LegCabinList_T& lLegCabinList =
00573         BomManager::getList<LegCabin> (*lLD_ptr);
00574     for (LegCabinList_T::const_iterator itLC = lLegCabinList.begin();
00575          itLC != lLegCabinList.end(); ++itLC) {
00576         const LegCabin* lLC_ptr = *itLC;
00577         assert (lLC_ptr != NULL);
00578
00579         ostream << lAirlineCode << lFlightNumber << " "
00580             << lFlightDateDate << ", ";
00581
00582         ostream << lBoardPoint << "-" << lOffPoint
00583             << " " << lLegDateDate << ", ";
00584
00585         ostream << lLC_ptr->getCabinCode() << ", ";
00586
00587         ostream << lLC_ptr->getOfferedCapacity() << ", "
00588             << lLC_ptr->getPhysicalCapacity() << ", "
00589             << lLC_ptr->getRegradeAdjustment() << ", "
00590             << lLC_ptr->getAuthorizationLevel() << ", "
00591             << lLC_ptr->getUPR() << ", "
00592             << lLC_ptr->getSoldSeat() << ", "
00593             << lLC_ptr->getStaffNbOfSeats() << ", "
00594             << lLC_ptr->getWLNbOfSeats() << ", "
00595             << lLC_ptr->getGroupNbOfSeats() << ", "
00596             << lLC_ptr->getCommittedSpace() << ", "
00597             << lLC_ptr->getAvailabilityPool() << ", "
00598             << lLC_ptr->getAvailability() << ", "
00599             << lLC_ptr->getNetAvailability() << ", "
00600             << lLC_ptr->getGrossAvailability() << ", "
00601             << lLC_ptr->getAvgCancellationPercentage() << ", "
00602             << lLC_ptr->getETB() << ", "
00603             << lLC_ptr->getCurrentBidPrice() << ", "
00604             << std::endl;
00605     }
00606 }
00607 ostream << "*****" << std::endl;
00608 }
00609

```

```

00610 // //////////////////////////////////////
00611 void BomDisplay::csvSegmentCabinDisplay (std::ostream& oStream,
00612                                         const FlightDate& iFlightDate) {
00613     // Save the formatting flags for the given STL output stream
00614     FlagSaver flagSaver (oStream);
00615
00619 }
00620
00621 // //////////////////////////////////////
00622 void BomDisplay::csvFareFamilyDisplay (std::ostream& oStream,
00623                                       const FlightDate& iFlightDate) {
00624     // Save the formatting flags for the given STL output stream
00625     FlagSaver flagSaver (oStream);
00626
00631     oStream << "*****" << std::endl;
00632     oStream << "SegmentCabins:" << std::endl
00633             << "-----" << std::endl;
00634     oStream << "Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, "
00635             << "CommSpace, AvPool, BP, " << std::endl;
00636
00637     // Retrieve the key of the flight-date
00638     const AirlineCode_T& lAirlineCode = iFlightDate.getAirlineCode();
00639     const FlightNumber_T& lFlightNumber = iFlightDate.getFlightNumber();
00640     const Date_T& lFlightDateDate = iFlightDate.getDepartureDate();
00641
00642     // Check whether there are SegmentDate objects
00643     if (BomManager::hasList<SegmentDate> (iFlightDate) == false) {
00644         return;
00645     }
00646
00647     // Browse the segment-dates
00648     const SegmentDateList_T& lSegmentDateList =
00649         BomManager::getList<SegmentDate> (iFlightDate);
00650     for (SegmentDateList_T::const_iterator itSD = lSegmentDateList.begin();
00651          itSD != lSegmentDateList.end(); ++itSD) {
00652         const SegmentDate* lSD_ptr = *itSD;
00653         assert (lSD_ptr != NULL);
00654
00655         // Retrieve the key of the segment-date, as well as its dates
00656         const Date_T& lSegmentDateDate = lSD_ptr->getBoardingDate();
00657         const AirportCode_T& lBoardPoint = lSD_ptr->getBoardingPoint();
00658         const AirportCode_T& lOffPoint = lSD_ptr->getOffPoint();
00659
00660         // Browse the segment-cabins
00661         const SegmentCabinList_T& lSegmentCabinList =
00662             BomManager::getList<SegmentCabin> (*lSD_ptr);
00663         for (SegmentCabinList_T::const_iterator itSC = lSegmentCabinList.begin();
00664              itSC != lSegmentCabinList.end(); ++itSC) {
00665             const SegmentCabin* lSC_ptr = *itSC;
00666             assert (lSC_ptr != NULL);
00667
00668             // Retrieve the key of the segment-cabin
00669             const CabinCode_T& lCabinCode = lSC_ptr->getCabinCode();
00670
00671             // Check whether there are fare family objects
00672             if (BomManager::hasList<FareFamily> (*lSC_ptr) == false) {
00673                 continue;
00674             }
00675
00676             // Browse the fare families
00677             const FareFamilyList_T& lFareFamilyList =
00678                 BomManager::getList<FareFamily> (*lSC_ptr);

```

```

00679         for (FareFamilyList_T::const_iterator itFF = lFareFamilyList.begin();
00680              itFF != lFareFamilyList.end(); ++itFF) {
00681             const FareFamily* lFF_ptr = *itFF;
00682             assert (lFF_ptr != NULL);
00683
00684             oStream << lAirlineCode << lFlightNumber << " "
00685                 << lFlightDate << ", ";
00686
00687             oStream << lBoardPoint << "-" << lOffPoint << " "
00688                 << lSegmentDate << ", ";
00689
00690             oStream << lCabinCode << ", " << lFF_ptr->getFamilyCode() << ", ";
00691
00692             oStream << lSC_ptr->getBookingCounter() << ", "
00693                 << lSC_ptr->getMIN() << ", "
00694                 << lSC_ptr->getUPR() << ", "
00695                 << lSC_ptr->getCommittedSpace() << ", "
00696                 << lSC_ptr->getAvailabilityPool() << ", "
00697                 << lSC_ptr->getCurrentBidPrice() << ", "
00698                 << std::endl;
00699         }
00700     }
00701 }
00702 oStream << "*****" << std::endl;
00703 }
00704
00705 // //////////////////////////////////////
00706 void BomDisplay::csvBucketDisplay (std::ostream& oStream,
00707                                   const FlightDate& iFlightDate) {
00708     // Save the formatting flags for the given STL output stream
00709     FlagSaver flagSaver (oStream);
00710
00711     oStream << "*****" << std::endl;
00712     oStream << "Buckets:" << std::endl
00713         << "-----" << std::endl;
00714     oStream << "Flight, Leg, Cabin, Yield, AU/SI, SS, AV, "
00715         << std::endl;
00716
00717     // Retrieve the key of the flight-date
00718     const AirlineCode_T& lAirlineCode = iFlightDate.getAirlineCode();
00719     const FlightNumber_T& lFlightNumber = iFlightDate.getFlightNumber();
00720     const Date_T& lFlightDateDate = iFlightDate.getDepartureDate();
00721
00722     // Check whether there are LegDate objects
00723     if (BomManager::hasList<LegDate> (iFlightDate) == false) {
00724         return;
00725     }
00726
00727     // Browse the leg-dates
00728     const LegDateList_T& lLegDateList =
00729         BomManager::getList<LegDate> (iFlightDate);
00730     for (LegDateList_T::const_iterator itLD = lLegDateList.begin();
00731          itLD != lLegDateList.end(); ++itLD) {
00732         const LegDate* lLD_ptr = *itLD;
00733         assert (lLD_ptr != NULL);
00734
00735         // Retrieve the key of the leg-date, as well as its off point
00736         const Date_T& lLegDateDate = lLD_ptr->getBoardingDate();
00737         const AirportCode_T& lBoardPoint = lLD_ptr->getBoardingPoint();
00738         const AirportCode_T& lOffPoint = lLD_ptr->getOffPoint();
00739
00740         // Browse the leg-cabins

```

```

00744     const LegCabinList_T& lLegCabinList =
00745         BomManager::getList<LegCabin> (*lLD_ptr);
00746     for (LegCabinList_T::const_iterator itLC = lLegCabinList.begin();
00747         itLC != lLegCabinList.end(); ++itLC) {
00748         const LegCabin* lLC_ptr = *itLC;
00749         assert (lLC_ptr != NULL);
00750
00751         // Check whether there are bucket objects
00752         if (BomManager::hasList<Bucket> (*lLC_ptr) == false) {
00753             continue;
00754         }
00755
00756         // Retrieve the key of the leg-cabin
00757         const CabinCode_T& lCabinCode = lLC_ptr->getCabinCode();
00758
00759         // Browse the buckets
00760         const BucketList_T& lBucketList = BomManager::getList<Bucket> (*lLC_ptr);
00761
00762         for (BucketList_T::const_iterator itBuck = lBucketList.begin();
00763             itBuck != lBucketList.end(); ++itBuck) {
00764             const Bucket* lBucket_ptr = *itBuck;
00765             assert (lBucket_ptr != NULL);
00766
00767             oStream << lAirlineCode << lFlightNumber << " "
00768                 << lFlightDate << ", ";
00769
00770             oStream << lBoardPoint << "-" << lOffPoint << " "
00771                 << lLegDate << ", " << lCabinCode << ", ";
00772
00773             oStream << lBucket_ptr->getYieldRangeUpperValue() << ", "
00774                 << lBucket_ptr->getSeatIndex() << ", "
00775                 << lBucket_ptr->getSoldSeats() << ", "
00776                 << lBucket_ptr->getAvailability() << ", ";
00777             oStream << std::endl;
00778         }
00779     }
00780     oStream << "*****" << std::endl;
00781 }
00782
00783 // //////////////////////////////////////
00784 void BomDisplay::csvBookingClassDisplay (std::ostream& oStream,
00785     const BookingClass& iBookingClass,
00786     const std::string& iLeadingString) {
00787     // Save the formatting flags for the given STL output stream
00788     FlagSaver flagSaver (oStream);
00789
00790     oStream << iLeadingString << iBookingClass.getClassCode();
00791
00792     if (iBookingClass.getSubclassCode() == 0) {
00793         oStream << ", ";
00794     } else {
00795         oStream << iBookingClass.getSubclassCode() << ", ";
00796     }
00797
00798     oStream << iBookingClass.getAuthorizationLevel() << " ("
00799         << iBookingClass.getProtection() << "), "
00800         << iBookingClass.getNegotiatedSpace() << ", "
00801         << iBookingClass.getNoShowPercentage() << ", "
00802         << iBookingClass.getCancellationPercentage() << ", "
00803         << iBookingClass.getNbOfBookings() << ", "
00804         << iBookingClass.getNbOfGroupBookings() << " ("
00805         << iBookingClass.getNbOfPendingGroupBookings() << "), "

```

```

00811         << iBookingClass.getNbOfStaffBookings() << ", "
00812         << iBookingClass.getNbOfWLBookings() << ", "
00813         << iBookingClass.getETB() << ", "
00814         << iBookingClass.getNetClassAvailability() << ", "
00815         << iBookingClass.getNetRevenueAvailability() << ", "
00816         << iBookingClass.getSegmentAvailability() << ", "
00817         << std::endl;
00818     }
00819
00820     // ////////////////////////////////////////
00821     void BomDisplay::csvBookingClassDisplay (std::ostream& oStream,
00822                                             const FlightDate& iFlightDate) {
00823         // Save the formatting flags for the given STL output stream
00824         FlagSaver flagSaver (oStream);
00825
00826         // Headers
00827         oStream << "*****" << std::endl;
00828         oStream << "Subclasses:" << std::endl
00829         << "-----" << std::endl;
00830         oStream << "Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), "
00831         << "Nego, NS%, OB%, "
00832         << "Bkgs, GrpBks (pdg), StfBkgs, WLBkgs, ETB, "
00833         << "ClassAvl, RevAvl, SegAvl, "
00834         << std::endl;
00835
00836         // Retrieve the key of the flight-date
00837         const AirlineCode_T& lAirlineCode = iFlightDate.getAirlineCode();
00838         const FlightNumber_T& lFlightNumber = iFlightDate.getFlightNumber();
00839         const Date_T& lFlightDateDate = iFlightDate.getDepartureDate();
00840
00841         // Check whether there are SegmentDate objects
00842         if (BomManager::hasList<SegmentDate> (iFlightDate) == false) {
00843             return;
00844         }
00845
00846         // Browse the segment-dates
00847         const SegmentDateList_T& lSegmentDateList =
00848             BomManager::getList<SegmentDate> (iFlightDate);
00849         for (SegmentDateList_T::const_iterator itSD = lSegmentDateList.begin();
00850             itSD != lSegmentDateList.end(); ++itSD) {
00851             const SegmentDate* lSD_ptr = *itSD;
00852             assert (lSD_ptr != NULL);
00853
00854             // Retrieve the key of the segment-date, as well as its dates
00855             const Date_T& lSegmentDateDate = lSD_ptr->getBoardingDate();
00856             const AirportCode_T& lBoardPoint = lSD_ptr->getBoardingPoint();
00857             const AirportCode_T& lOffPoint = lSD_ptr->getOffPoint();
00858
00859             // Browse the segment-cabins
00860             const SegmentCabinList_T& lSegmentCabinList =
00861                 BomManager::getList<SegmentCabin> (*lSD_ptr);
00862             for (SegmentCabinList_T::const_iterator itSC = lSegmentCabinList.begin();
00863                 itSC != lSegmentCabinList.end(); ++itSC) {
00864                 const SegmentCabin* lSC_ptr = *itSC;
00865                 assert (lSC_ptr != NULL);
00866
00867                 // Retrieve the key of the segment-cabin
00868                 const CabinCode_T& lCabinCode = lSC_ptr->getCabinCode();
00869
00870                 // Build the leading string to be displayed
00871                 std::ostringstream oSCLeadingStr;
00872                 oSCLeadingStr << lAirlineCode << lFlightNumber << " "

```

```

00873         << lFlightDateDate << ", "
00874         << lBoardPoint << "-" << lOffPoint << " "
00875         << lSegmentDateDate << ", "
00876         << lCabinCode << ", ";
00877
00878     // Default Fare Family code, when there are no FF
00879     FamilyCode_T lFamilyCode ("NoFF");
00880
00881     // Check whether there are FareFamily objects
00882     if (BomManager::hasList<FareFamily> (*lSC_ptr) == true) {
00883
00884         // Browse the fare families
00885         const FareFamilyList_T& lFareFamilyList =
00886             BomManager::getList<FareFamily> (*lSC_ptr);
00887         for (FareFamilyList_T::const_iterator itFF = lFareFamilyList.begin();
00888             itFF != lFareFamilyList.end(); ++itFF) {
00889             const FareFamily* lFF_ptr = *itFF;
00890             assert (lFF_ptr != NULL);
00891
00892             // Retrieve the key of the segment-cabin
00893             lFamilyCode = lFF_ptr->getFamilyCode();
00894
00895             // Complete the leading string to be displayed
00896             std::ostringstream oFFLeadingStr;
00897             oFFLeadingStr << oSCLeadingStr.str() << lFamilyCode << ", ";
00898
00899             // Browse the booking-classes
00900             const BookingClassList_T& lBookingClassList =
00901                 BomManager::getList<BookingClass> (*lFF_ptr);
00902             for (BookingClassList_T::const_iterator itBC =
00903                 lBookingClassList.begin();
00904                 itBC != lBookingClassList.end(); ++itBC) {
00905                 const BookingClass* lBC_ptr = *itBC;
00906                 assert (lBC_ptr != NULL);
00907
00908                 //
00909                 csvBookingClassDisplay (oStream, *lBC_ptr, oFFLeadingStr.str());
00910             }
00911         }
00912
00913         // Go on to the next segment-cabin
00914         continue;
00915     }
00916     assert (BomManager::hasList<FareFamily> (*lSC_ptr) == false);
00917
00918     // The fare family code is a fake one ('NoFF'), and therefore
00919     // does not vary
00920     std::ostringstream oFFLeadingStr;
00921     oFFLeadingStr << oSCLeadingStr.str() << lFamilyCode << ", ";
00922
00923     // Browse the booking-classes, directly from the segment-cabin object
00924     const BookingClassList_T& lBookingClassList =
00925         BomManager::getList<BookingClass> (*lSC_ptr);
00926     for (BookingClassList_T::const_iterator itBC =
00927         lBookingClassList.begin();
00928         itBC != lBookingClassList.end(); ++itBC) {
00929         const BookingClass* lBC_ptr = *itBC;
00930         assert (lBC_ptr != NULL);
00931
00932         //
00933         csvBookingClassDisplay (oStream, *lBC_ptr, oFFLeadingStr.str());
00934     }

```

```

00935     }
00936 }
00937 oStream << "*****" << std::endl;
00938 }
00939
00940 // //////////////////////////////////////
00941 void BomDisplay::
00942 csvDisplay (std::ostream& oStream,
00943             const TravelSolutionList_T& iTravelSolutionList) {
00944
00945     // Save the formatting flags for the given STL output stream
00946     FlagSaver flagSaver (oStream);
00947
00948     oStream << "Travel solutions:";
00949     unsigned short idx = 0;
00950     for (TravelSolutionList_T::const_iterator itTS =
00951          iTravelSolutionList.begin();
00952          itTS != iTravelSolutionList.end(); ++itTS, ++idx) {
00953         const TravelSolutionStruct& lTS = *itTS;
00954
00955         oStream << std::endl;
00956         oStream << "    [" << idx << "] " << lTS.display();
00957     }
00958 }
00959
00960 // //////////////////////////////////////
00961 void BomDisplay::
00962 csvDisplay (std::ostream& oStream,
00963             const DatePeriodList_T& iDatePeriodList) {
00964
00965     // Save the formatting flags for the given STL output stream
00966     FlagSaver flagSaver (oStream);
00967
00968     // Browse the date-period objects
00969     for (DatePeriodList_T::const_iterator itDP = iDatePeriodList.begin();
00970          itDP != iDatePeriodList.end(); ++itDP) {
00971         const DatePeriod* lDP_ptr = *itDP;
00972         assert (lDP_ptr != NULL);
00973
00974         // Display the date-period object
00975         csvDateDisplay (oStream, *lDP_ptr);
00976     }
00977 }
00978
00979 // //////////////////////////////////////
00980 void BomDisplay::csvSimFQTAirRACDisplay (std::ostream& oStream,
00981                                          const BomRoot& iBomRoot) {
00982     // Save the formatting flags for the given STL output stream
00983     FlagSaver flagSaver (oStream);
00984
00985     oStream << std::endl;
00986     oStream << "=====
00987     << std::endl;
00988     oStream << "BomRoot: " << iBomRoot.describeKey() << std::endl;
00989     oStream << "=====
00990     << std::endl;
00991
00992     // Check whether there are airport-pair objects
00993     if (BomManager::hasList<AirportPair> (iBomRoot) == false) {
00994         return;
00995     }
00996 }
00997
00998 }
00999

```

```

01000 // Browse the airport-pair objects
01001 const AirportPairList_T& lAirportPairList =
01002     BomManager::getList<AirportPair> (iBomRoot);
01003 for (AirportPairList_T::const_iterator itAir = lAirportPairList.begin();
01004     itAir != lAirportPairList.end(); ++itAir ) {
01005     const AirportPair* lAir_ptr = *itAir;
01006     assert (lAir_ptr != NULL);
01007
01008     // Display the airport pair object
01009     csvAirportPairDisplay (oStream, *lAir_ptr);
01010 }
01011 }
01012
01013 // //////////////////////////////////////
01014 void BomDisplay::csvAirportPairDisplay (std::ostream& oStream,
01015     const AirportPair& iAirportPair) {
01016     // Save the formatting flags for the given STL output stream
01017     FlagSaver flagSaver (oStream);
01018
01019     oStream << "++++++" << std::endl;
01020     oStream << "AirportPair: " << iAirportPair.describeKey() << std::endl;
01021     oStream << "++++++" << std::endl;
01022
01023     // Check whether there are date-period objects
01024     if (BomManager::hasList<DatePeriod> (iAirportPair) == false) {
01025         return;
01026     }
01027
01028     // Browse the date-period objects
01029     const DatePeriodList_T& lDatePeriodList =
01030         BomManager::getList<DatePeriod> (iAirportPair);
01031     for (DatePeriodList_T::const_iterator itDP = lDatePeriodList.begin();
01032         itDP != lDatePeriodList.end(); ++itDP) {
01033         const DatePeriod* lDP_ptr = *itDP;
01034         assert (lDP_ptr != NULL);
01035
01036         // Display the date-period object
01037         csvDateDisplay (oStream, *lDP_ptr);
01038     }
01039 }
01040
01041 // //////////////////////////////////////
01042 void BomDisplay::csvDateDisplay (std::ostream& oStream,
01043     const DatePeriod& iDatePeriod) {
01044     // Save the formatting flags for the given STL output stream
01045     FlagSaver flagSaver (oStream);
01046
01047     oStream << "-----" << std::endl;
01048     oStream << "DatePeriod: " << iDatePeriod.describeKey() << std::endl;
01049     oStream << "-----" << std::endl;
01050
01051     // Check whether there are pos-channel objects
01052     if (BomManager::hasList<PosChannel> (iDatePeriod) == false) {
01053         return;
01054     }
01055
01056     // Browse the pos-channel objects
01057     const PosChannelList_T& lPosChannelList =
01058         BomManager::getList<PosChannel> (iDatePeriod);
01059     for (PosChannelList_T::const_iterator itPC = lPosChannelList.begin();
01060         itPC != lPosChannelList.end(); ++itPC) {

```

```

01068     const PosChannel* lPC_ptr = *itPC;
01069     assert (lPC_ptr != NULL);
01070
01071     // Display the pos-channel object
01072     csvPosChannelDisplay (oStream, *lPC_ptr);
01073 }
01074 }
01075
01076 // //////////////////////////////////////
01077 void BomDisplay::csvPosChannelDisplay (std::ostream& oStream,
01078                                     const PosChannel& iPosChannel) {
01079     // Save the formatting flags for the given STL output stream
01080     FlagSaver flagSaver (oStream);
01081
01082     oStream << "*****" << std::endl;
01083     oStream << "PosChannel: " << iPosChannel.describeKey() << std::endl;
01084     oStream << "*****" << std::endl;
01085
01086     // Check whether there are time-period objects
01087     if (BomManager::hasList<TimePeriod> (iPosChannel) == false) {
01088         return;
01089     }
01090
01091     // Browse the time-period objects
01092     const TimePeriodList_T& lTimePeriodList =
01093         BomManager::getList<TimePeriod> (iPosChannel);
01094     for (TimePeriodList_T::const_iterator itTP = lTimePeriodList.begin();
01095          itTP != lTimePeriodList.end(); ++itTP) {
01096         const TimePeriod* lTP_ptr = *itTP;
01097         assert (lTP_ptr != NULL);
01098
01099         // Display the time-period object
01100         csvTimeDisplay (oStream, *lTP_ptr);
01101     }
01102 }
01103
01104 // //////////////////////////////////////
01105 void BomDisplay::csvTimeDisplay (std::ostream& oStream,
01106                                const TimePeriod& iTimePeriod) {
01107     // Save the formatting flags for the given STL output stream
01108     FlagSaver flagSaver (oStream);
01109
01110     oStream << "-----" << std::endl;
01111     oStream << "TimePeriod: " << iTimePeriod.describeKey() << std::endl;
01112     oStream << "-----" << std::endl;
01113
01114     // Only one of the fare/yield feature list exists. Each of the following
01115     // two methods will check for the existence of the list. So, only the
01116     // existing list will be actually displayed.
01117     csvFeatureListDisplay<FareFeatures> (oStream, iTimePeriod);
01118     csvFeatureListDisplay<YieldFeatures> (oStream, iTimePeriod);
01119 }
01120
01121 // //////////////////////////////////////
01122 template <typename FEATURE_TYPE>
01123 void BomDisplay::csvFeatureListDisplay (std::ostream& oStream,
01124                                       const TimePeriod& iTimePeriod) {
01125     // Check whether there are fare/yield-feature objects
01126     if (BomManager::hasList<FEATURE_TYPE> (iTimePeriod) == false) {
01127         return;
01128     }

```

```

01136     }
01137
01138     // Browse the fare/yield-feature objects
01139     typedef typename BomHolder<FEATURE_TYPE>::BomList_T FeaturesList_T;
01140     const FeaturesList_T& lFeaturesList =
01141         BomManager::getList<FEATURE_TYPE> (iTimePeriod);
01142     for (typename FeaturesList_T::const_iterator itFF = lFeaturesList.begin();
01143          itFF != lFeaturesList.end(); ++itFF) {
01144         const FEATURE_TYPE* lFF_ptr = *itFF;
01145         assert (lFF_ptr != NULL);
01146
01147         // Display the fare-features object
01148         csvFeaturesDisplay (oStream, *lFF_ptr);
01149     }
01150 }
01151
01152 // //////////////////////////////////////
01153 template <typename FEATURE_TYPE>
01154 void BomDisplay::csvFeaturesDisplay (std::ostream& oStream,
01155                                     const FEATURE_TYPE& iFeatures) {
01156     // Save the formatting flags for the given STL output stream
01157     FlagSaver flagSaver (oStream);
01158
01159     oStream << "-----" << std::endl;
01160     oStream << "Fare/yield-Features: " << iFeatures.describeKey() << std::endl;
01161     oStream << "-----" << std::endl;
01162
01163     // Check whether there are airlineClassList objects
01164     if (BomManager::hasList<AirlineClassList> (iFeatures) == false) {
01165         return;
01166     }
01167
01168     // Browse the airlineClassList objects
01169     const AirlineClassListList_T& lAirlineClassListList =
01170         BomManager::getList<AirlineClassList> (iFeatures);
01171     for (AirlineClassListList_T::const_iterator itACL =
01172          lAirlineClassListList.begin();
01173          itACL != lAirlineClassListList.end(); ++itACL) {
01174         const AirlineClassList* lACL_ptr = *itACL;
01175         assert (lACL_ptr != NULL);
01176
01177         // Display the airlineClassList object
01178         csvAirlineClassDisplay (oStream, *lACL_ptr);
01179     }
01180 }
01181
01182 // //////////////////////////////////////
01183 void BomDisplay::
01184 csvAirlineClassDisplay (std::ostream& oStream,
01185                        const AirlineClassList& iAirlineClassList) {
01186     // Save the formatting flags for the given STL output stream
01187     FlagSaver flagSaver (oStream);
01188
01189     oStream << "-----" << std::endl;
01190     oStream << "AirlineClassList: "
01191             << iAirlineClassList.describeKey() << std::endl;
01192     oStream << "-----" << std::endl;
01193 }
01194
01195
01196
01197
01198
01199
01200
01201 }
01202

```

35.159 stdair/bom/BomDisplay.hpp File Reference

```
#include <iosfwd>
#include <stdair/bom/TravelSolutionTypes.hpp>
#include <stdair/bom/DatePeriodTypes.hpp>
```

Classes

- class [stdair::BomDisplay](#)
Utility class to display StdAir objects with a pretty format.

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.160 BomDisplay.hpp

```
00001 #ifndef __STDAIR_BOM_BOMDISPLAY_HPP
00002 #define __STDAIR_BOM_BOMDISPLAY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 // StdAir
00010 #include <stdair/bom/TravelSolutionTypes.hpp>
00011 #include <stdair/bom/DatePeriodTypes.hpp>
00012
00013 namespace stdair {
00014
00016     class BomRoot;
00017     class EventQueue;
00018     class Inventory;
00019     class FlightDate;
00020     class LegDate;
00021     class SegmentDate;
00022     class LegCabin;
00023     class SegmentCabin;
00024     class FareFamily;
00025     class BookingClass;
00026     class AirportPair;
00027     class PosChannel;
00028     class DatePeriod;
00029     class TimePeriod;
00030     class FareFeatures;
00031     class YieldFeatures;
00032     class AirlineClassList;
00033     class OnDDate;
00034
00039     class BomDisplay {
00040     public:
```

```

00041 // ////////////////////////////////// Display support methods //////////////////////////////////
00050 static std::string csvDisplay (const EventQueue&);
00051
00066 static void list (std::ostream&, const BomRoot&,
00067                  const AirlineCode_T& iAirlineCode = "all",
00068                  const FlightNumber_T& iFlightNumber = 0);
00069
00083 static void list (std::ostream&, const Inventory&,
00084                  const unsigned short iInventoryIndex = 0,
00085                  const FlightNumber_T& iFlightNumber = 0);
00086
00095 static void listAirportPairDateRange (std::ostream&,
00096                                       const BomRoot&);
00097
00106 static void csvDisplay (std::ostream&, const BomRoot&);
00107
00116 static void csvDisplay (std::ostream&, const Inventory&);
00117
00125 static void csvDisplay (std::ostream&, const OnDDate&);
00126
00135 static void csvDisplay (std::ostream&, const FlightDate&);
00136
00145 static void csvLegDateDisplay (std::ostream&, const FlightDate&);
00146
00155 static void csvSegmentDateDisplay (std::ostream&, const FlightDate&);
00156
00165 static void csvLegCabinDisplay (std::ostream&, const FlightDate&);
00166
00175 static void csvSegmentCabinDisplay (std::ostream&, const FlightDate&);
00176
00185 static void csvFareFamilyDisplay (std::ostream&, const FlightDate&);
00186
00195 static void csvBucketDisplay (std::ostream&, const FlightDate&);
00196
00206 static void csvBookingClassDisplay (std::ostream&, const BookingClass&,
00207                                     const std::string& iLeadingString);
00216 static void csvBookingClassDisplay (std::ostream&, const FlightDate&);
00217
00226 static void csvDisplay (std::ostream&, const TravelSolutionList_T&);
00227
00236 static void csvDisplay (std::ostream&, const DatePeriodList_T&);
00237
00246 static void csvSimFQTAirRACDisplay (std::ostream&, const BomRoot&);
00247
00257 static void csvAirportPairDisplay (std::ostream&, const AirportPair&);
00258
00268 static void csvDateDisplay (std::ostream&, const DatePeriod&);
00269
00279 static void csvPosChannelDisplay (std::ostream&, const PosChannel&);
00280
00290 static void csvTimeDisplay (std::ostream&, const TimePeriod&);
00291
00300 template <typename FEATURE_TYPE>
00301 static void csvFeatureListDisplay (std::ostream& oStream, const TimePeriod&);
00302
00311 template <typename FEATURE_TYPE>
00312 static void csvFeaturesDisplay (std::ostream& oStream, const FEATURE_TYPE&);
00313
00322 static void csvAirlineClassDisplay (std::ostream&, const AirlineClassList&);
00323 };
00324

```

```

00325 }
00326 #endif // __STDAIR_BOM_BOMDISPLAY_HPP

```

35.161 stdair/bom/BomHolder.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <list>
#include <map>
#include <stdair/bom/key_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/BomHolderKey.hpp>

```

Classes

- class [stdair::BomHolder< BOM >](#)

Class representing the holder of BOM object containers (list and map).

Namespaces

- namespace [stdair](#)

Handle on the StdAir library context.

35.162 BomHolder.hpp

```

00001 #ifndef __STDAIR_BOM_BOMHOLDER_HPP
00002 #define __STDAIR_BOM_BOMHOLDER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 #include <list>
00011 #include <map>
00012 // StdAir
00013 #include <stdair/bom/key_types.hpp>
00014 #include <stdair/bom/BomAbstract.hpp>
00015 #include <stdair/bom/BomHolderKey.hpp>
00016
00017 namespace stdair {
00018
00023     template <typename BOM>
00024     class BomHolder : public stdair::BomAbstract {
00026         template <typename> friend class FacBom;
00027         friend class FacBomManager;
00028

```

```

00029 public:
00030     // //////////////// Type definitions ////////////////
00034     typedef stdair::BomHolderKey Key_T;
00035
00039     typedef std::list<BOM*> BomList_T;
00040
00044     typedef std::map<const MapKey_T, BOM*> BomMap_T;
00045
00046
00047 public:
00048     // //////////////// Display support methods ////////////////
00054     void toStream (std::ostream& ioOut) const {
00055         ioOut << toString();
00056     }
00057
00063     void fromStream (std::istream& ioIn) {
00064     }
00065
00069     std::string toString() const {
00070         return "BomHolder";
00071     }
00072
00076     const std::string describeKey() const {
00077         return "BomHolder";
00078     }
00079
00080 protected:
00084     BomHolder();
00085
00089     BomHolder (const BomHolder&);
00090
00094     BomHolder (const Key_T& iKey) : _key (iKey) { }
00095
00099     ~BomHolder() { };
00100
00101 public:
00102     // //////////////// Attributes ////////////////
00106     Key_T _key;
00107
00111     BomList_T _bomList;
00112
00116     BomMap_T _bomMap;
00117 };
00118
00119 }
00120 #endif // __STDAIR_BOM_BOMHOLDER_HPP

```

35.163 stdair/bom/BomHolderKey.cpp File Reference

```

#include <ostream>

#include <sstream>

#include <stdair/bom/BomHolderKey.hpp>

```

Namespaces

- namespace [stdair](#)

Handle on the StdAir library context.

35.164 BomHolderKey.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <ostream>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/bom/BomHolderKey.hpp>
00009
00010 namespace stdair {
00011
00012 // //////////////////////////////////////
00013 BomHolderKey::BomHolderKey () {
00014 }
00015
00016 // //////////////////////////////////////
00017 BomHolderKey::~BomHolderKey () {
00018 }
00019
00020 // //////////////////////////////////////
00021 void BomHolderKey::toStream (std::ostream& ioOut) const {
00022     ioOut << "BomHolderKey: " << toString() << std::endl;
00023 }
00024
00025 // //////////////////////////////////////
00026 void BomHolderKey::fromStream (std::istream& ioIn) {
00027 }
00028
00029 // //////////////////////////////////////
00030 const std::string BomHolderKey::toString() const {
00031     std::ostringstream oStr;
00032     oStr << " -- HOLDER -- ";
00033     return oStr.str();
00034 }
00035
00036 }

```

35.165 stdair/bom/BomHolderKey.hpp File Reference

```
#include <stdair/bom/KeyAbstract.hpp>
```

Classes

- struct [stdair::BomHolderKey](#)

Namespaces

- namespace [stdair](#)

Handle on the StdAir library context.

35.166 BomHolderKey.hpp

```

00001 #ifndef __STDAIR_BOM_BOMHOLDERKEY_HPP
00002 #define __STDAIR_BOM_BOMHOLDERKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STDAIR
00008 #include <stdair/bom/KeyAbstract.hpp>
00009
00010 namespace stdair {
00012     struct BomHolderKey : public KeyAbstract {
00013
00014     public:
00015         // ////////////////////////////////// Construction //////////////////////////////////
00017         BomHolderKey ();
00019         ~BomHolderKey ();
00020
00021         // ////////////////////////////////// Display support methods //////////////////////////////////
00024         void toStream (std::ostream& ioOut) const;
00025
00028         void fromStream (std::istream& ioIn);
00029
00035         const std::string toString() const;
00036
00038         const std::string describe() const;
00039
00040     };
00041
00042 }
00043 #endif // __STDAIR_BOM_BOMHOLDERKEY_HPP

```

35.167 stdair/bom/BomJSONExport.cpp File Reference

```

#include <cassert>
#include <ostream>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/Bucket.hpp>

```

```
#include <stdair/bom/BomJSONExport.hpp>
```

Classes

- struct `stdair::FlagSaver`

Namespaces

- namespace `bpt`
- namespace `stdair`

Handle on the StdAir library context.

Typedefs

- typedef char `bpt::ptree`

Functions

- void `stdair::jsonLegDateExport` (`bpt::ptree` &ioPropertyTree, const FlightDate &iFlightDate)
- void `stdair::jsonSegmentDateExport` (`bpt::ptree` &ioPropertyTree, const FlightDate &iFlightDate)
- void `stdair::jsonLegCabinExport` (`bpt::ptree` &ioPropertyTree, const FlightDate &iFlightDate)
- void `stdair::jsonSegmentCabinExport` (`bpt::ptree` &ioPropertyTree, const FlightDate &iFlightDate)
- void `stdair::jsonFareFamilyExport` (`bpt::ptree` &ioPropertyTree, const FlightDate &iFlightDate)
- void `stdair::jsonBucketExport` (`bpt::ptree` &ioPropertyTree, const FlightDate &iFlightDate)
- void `stdair::jsonBookingClassExport` (`bpt::ptree` &ioPropertyTree, const BookingClass &iBookingClass, const std::string &iLeadingString)
- void `stdair::jsonBookingClassExport` (`bpt::ptree` &ioPropertyTree, const FlightDate &iFlightDate)

35.168 BomJSONExport.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <ostream>
00007 #if BOOST_VERSION >= 104100
00008 // Boost Property Tree
00009 #include <boost/property_tree/ptree.hpp>
00010 #include <boost/property_tree/json_parser.hpp>
```

```

00011 #endif // BOOST_VERSION >= 104100
00012 #if BOOST_VERSION >= 103400
00013 // Boost ForEach
00014 #include <boost/foreach.hpp>
00015 #endif // BOOST_VERSION >= 103400
00016 // StdAir
00017 #include <stdair/basic/BasConst_BomDisplay.hpp>
00018 #include <stdair/bom/BomManager.hpp>
00019 #include <stdair/bom/BomRoot.hpp>
00020 #include <stdair/bom/Inventory.hpp>
00021 #include <stdair/bom/FlightDate.hpp>
00022 #include <stdair/bom/LegDate.hpp>
00023 #include <stdair/bom/SegmentDate.hpp>
00024 #include <stdair/bom/LegCabin.hpp>
00025 #include <stdair/bom/SegmentCabin.hpp>
00026 #include <stdair/bom/FareFamily.hpp>
00027 #include <stdair/bom/BookingClass.hpp>
00028 #include <stdair/bom/Bucket.hpp>
00029 #include <stdair/bom/BomJSONExport.hpp>
00030
00031 #if BOOST_VERSION >= 104100
00032 namespace bpt = boost::property_tree;
00033 #else // BOOST_VERSION >= 104100
00034 namespace bpt {
00035     typedef char ptrtree;
00036 }
00037 #endif // BOOST_VERSION >= 104100
00038
00039 namespace stdair {
00040
00041     struct FlagSaver {
00042     public:
00043         FlagSaver (std::ostream& oStream)
00044             : _oStream (oStream), _streamFlags (oStream.flags()) {
00045         }
00046
00047         ~FlagSaver() {
00048             // Reset formatting flags of the given output stream
00049             _oStream.flags (_streamFlags);
00050         }
00051
00052     private:
00053         std::ostream& _oStream;
00054         std::ios::fmtflags _streamFlags;
00055     };
00056
00057 // //////////////////////////////////////
00058 void jsonLegDateExport (bpt::ptrtree& ioPropertyTree,
00059                        const FlightDate& iFlightDate) {
00060     // Check whether there are LegDate objects
00061     if (BomManager::hasList<LegDate> (iFlightDate) == false) {
00062         return;
00063     }
00064
00065     // Browse the leg-dates
00066     unsigned short idx = 0;
00067     const LegDateList_T& lLegDateList =
00068         BomManager::getList<LegDate> (iFlightDate);
00069     for (LegDateList_T::const_iterator itLD = lLegDateList.begin();
00070          itLD != lLegDateList.end(); ++itLD, ++idx) {
00071         const LegDate* lLD_ptr = *itLD;
00072         assert (lLD_ptr != NULL);
00073     }
00074 }

```

```

00086
00087 #if BOOST_VERSION >= 104100
00088     //
00089     bpt::ptree lLegDateArray;
00090
00091     // Put boarding point in property tree
00092     lLegDateArray.put ("board_point", lLD_ptr->getBoardingPoint());
00093     // Put off point in property tree
00094     lLegDateArray.put ("off_point", lLD_ptr->getOffPoint());
00095     // Put boarding date in property tree
00096     lLegDateArray.put ("board_date", lLD_ptr->getBoardingDate());
00097     // Put off date in property tree
00098     lLegDateArray.put ("off_date", lLD_ptr->getOffDate());
00099     // Put boarding time in property tree
00100     lLegDateArray.put ("board_time", lLD_ptr->getBoardingTime());
00101     // Put off time in property tree
00102     lLegDateArray.put ("off_time", lLD_ptr->getOffTime());
00103     // Put elapsed time in property tree
00104     lLegDateArray.put ("elapsed_time", lLD_ptr->getElapsedTime());
00105     // Put date offset in property tree
00106     lLegDateArray.put ("date_offset", lLD_ptr->getDateOffset());
00107     // Put time offset in property tree
00108     lLegDateArray.put ("time_offset", lLD_ptr->getTimeOffset());
00109     // Put distance in property tree
00110     lLegDateArray.put ("distance", lLD_ptr->getDistance());
00111     // Put capacity in property tree
00112     lLegDateArray.put ("capacity", lLD_ptr->getCapacity());
00113
00114     //
00115     std::ostringstream oStream;
00116     oStream << "flight_date.legs";
00117     ioPropertyTree.put_child (oStream.str(), lLegDateArray);
00118 #endif // BOOST_VERSION >= 104100
00119 }
00120 }
00121
00122 // //////////////////////////////////////
00123 void jsonSegmentDateExport (bpt::ptree& ioPropertyTree,
00124                             const FlightDate& iFlightDate) {
00125 }
00126
00127 // //////////////////////////////////////
00128 void jsonLegCabinExport (bpt::ptree& ioPropertyTree,
00129                          const FlightDate& iFlightDate) {
00130     // Check whether there are LegDate objects
00131     if (BomManager::hasList<LegDate> (iFlightDate) == false) {
00132         return;
00133     }
00134
00135     // Browse the leg-dates
00136     const LegDateList_T& lLegDateList =
00137         BomManager::getList<LegDate> (iFlightDate);
00138     for (LegDateList_T::const_iterator itLD = lLegDateList.begin();
00139          itLD != lLegDateList.end(); ++itLD) {
00140         const LegDate* lLD_ptr = *itLD;
00141         assert (lLD_ptr != NULL);
00142
00143         // Browse the leg-cabins
00144         const LegCabinList_T& lLegCabinList =
00145             BomManager::getList<LegCabin> (*lLD_ptr);
00146         for (LegCabinList_T::const_iterator itLC = lLegCabinList.begin();
00147              itLC != lLegCabinList.end(); ++itLC) {

```

```

00155         const LegCabin* lLC_ptr = *itLC;
00156         assert (lLC_ptr != NULL);
00157
00158 #if BOOST_VERSION >= 104100
00159     // Put boarding point in property tree
00160     ioPropertyTree.put ("flight_date.leg.cabin.code",
00161         lLC_ptr->getCabinCode());
00162
00163     /*
00164     ostream << lLC_ptr->getOfferedCapacity() << ", "
00165         << lLC_ptr->getPhysicalCapacity() << ", "
00166         << lLC_ptr->getRegradeAdjustment() << ", "
00167         << lLC_ptr->getAuthorizationLevel() << ", "
00168         << lLC_ptr->getUPR() << ", "
00169         << lLC_ptr->getSoldSeat() << ", "
00170         << lLC_ptr->getStaffNbOfSeats() << ", "
00171         << lLC_ptr->getWLNbOfSeats() << ", "
00172         << lLC_ptr->getGroupNbOfSeats() << ", "
00173         << lLC_ptr->getCommittedSpace() << ", "
00174         << lLC_ptr->getAvailabilityPool() << ", "
00175         << lLC_ptr->getAvailability() << ", "
00176         << lLC_ptr->getNetAvailability() << ", "
00177         << lLC_ptr->getGrossAvailability() << ", "
00178         << lLC_ptr->getAvgCancellationPercentage() << ", "
00179         << lLC_ptr->getETB() << ", "
00180         << lLC_ptr->getCurrentBidPrice() << ", "
00181         << std::endl;
00182     */
00183 #endif // BOOST_VERSION >= 104100
00184     }
00185 }
00186 }
00187
00188 ///////////////////////////////////////////////////////////////////
00189 void jsonSegmentCabinExport (bpt::ptree& ioPropertyTree,
00190     const FlightDate& iFlightDate) {
00194 }
00195
00196 ///////////////////////////////////////////////////////////////////
00197 void jsonFareFamilyExport (bpt::ptree& ioPropertyTree,
00198     const FlightDate& iFlightDate) {
00203     // Check whether there are SegmentDate objects
00204     if (BomManager::hasList<SegmentDate> (iFlightDate) == false) {
00205         return;
00206     }
00207
00208     // Browse the segment-dates
00209     const SegmentDateList_T& lSegmentDateList =
00210         BomManager::getList<SegmentDate> (iFlightDate);
00211     for (SegmentDateList_T::const_iterator itSD = lSegmentDateList.begin();
00212         itSD != lSegmentDateList.end(); ++itSD) {
00213         const SegmentDate* lSD_ptr = *itSD;
00214         assert (lSD_ptr != NULL);
00215
00216         // Browse the segment-cabins
00217         const SegmentCabinList_T& lSegmentCabinList =
00218             BomManager::getList<SegmentCabin> (*lSD_ptr);
00219         for (SegmentCabinList_T::const_iterator itSC = lSegmentCabinList.begin();
00220             itSC != lSegmentCabinList.end(); ++itSC) {
00221             const SegmentCabin* lSC_ptr = *itSC;
00222             assert (lSC_ptr != NULL);
00223

```

```

00224         // Check whether there are fare family objects
00225         if (BomManager::hasList<FareFamily> (*lSC_ptr) == false) {
00226             continue;
00227         }
00228
00229         // Browse the fare families
00230         unsigned short ffIdx = 0;
00231         const FareFamilyList_T& lFareFamilyList =
00232             BomManager::getList<FareFamily> (*lSC_ptr);
00233         for (FareFamilyList_T::const_iterator itFF = lFareFamilyList.begin();
00234             itFF != lFareFamilyList.end(); ++itFF, ++ffIdx) {
00235             const FareFamily* lFF_ptr = *itFF;
00236             assert (lFF_ptr != NULL);
00237
00238             #if BOOST_VERSION >= 104100
00239                 //
00240                 bpt::ptree lFFArray;
00241
00242                 // Put boarding point in property tree
00243                 lFFArray.put ("code", lFF_ptr->getFamilyCode());
00244
00245                 /*
00246                 oStream << lSC_ptr->getBookingCounter() << ", "
00247                     << lSC_ptr->getMIN() << ", "
00248                     << lSC_ptr->getUPR() << ", "
00249                     << lSC_ptr->getCommittedSpace() << ", "
00250                     << lSC_ptr->getAvailabilityPool() << ", "
00251                     << lSC_ptr->getCurrentBidPrice() << ", "
00252                     << std::endl;
00253                 */
00254
00255                 //
00256                 std::ostringstream oStream;
00257                 oStream << "flight_date.segment.cabin.fare_families";
00258                 ioPropertyTree.put_child (oStream.str(), lFFArray);
00259             #endif // BOOST_VERSION >= 104100
00260         }
00261     }
00262 }
00263
00264 // //////////////////////////////////////
00265 void jsonBucketExport (bpt::ptree& ioPropertyTree,
00266                       const FlightDate& iFlightDate) {
00267     // Check whether there are LegDate objects
00268     if (BomManager::hasList<LegDate> (iFlightDate) == false) {
00269         return;
00270     }
00271
00272     // Browse the leg-dates
00273     const LegDateList_T& lLegDateList =
00274         BomManager::getList<LegDate> (iFlightDate);
00275     for (LegDateList_T::const_iterator itLD = lLegDateList.begin();
00276         itLD != lLegDateList.end(); ++itLD) {
00277         const LegDate* lLD_ptr = *itLD;
00278         assert (lLD_ptr != NULL);
00279
00280         // Browse the leg-cabins
00281         const LegCabinList_T& lLegCabinList =
00282             BomManager::getList<LegCabin> (*lLD_ptr);
00283         for (LegCabinList_T::const_iterator itLC = lLegCabinList.begin();
00284             itLC != lLegCabinList.end(); ++itLC) {

```

```

00290     const LegCabin* lLC_ptr = *itLC;
00291     assert (lLC_ptr != NULL);
00292
00293     // Check whether there are bucket objects
00294     if (BomManager::hasList<Bucket> (*lLC_ptr) == false) {
00295         continue;
00296     }
00297
00298     // Browse the buckets
00299     const BucketList_T& lBucketList = BomManager::getList<Bucket> (*lLC_ptr);
00300
00301     for (BucketList_T::const_iterator itBuck = lBucketList.begin();
00302          itBuck != lBucketList.end(); ++itBuck) {
00303         const Bucket* lBucket_ptr = *itBuck;
00304         assert (lBucket_ptr != NULL);
00305
00306         #if BOOST_VERSION >= 104100
00307             // Put yield range upper value in property tree
00308             ioPropertyTree.put ("flight_date.leg.cabin.bucket.yield_range_upper",
00309                                lBucket_ptr->getYieldRangeUpperValue());
00310             /*
00311             oStream << lBucket_ptr->getSeatIndex() << ", "
00312                     << lBucket_ptr->getSoldSeats() << ", "
00313                     << lBucket_ptr->getAvailability() << ", ";
00314             */
00315         #endif // BOOST_VERSION >= 104100
00316     }
00317 }
00318 }
00319
00320 // //////////////////////////////////////
00321 void jsonBookingClassExport (bpt::ptree& ioPropertyTree,
00322                             const BookingClass& iBookingClass,
00323                             const std::string& iLeadingString) {
00324     std::ostream oStream;
00325     oStream << iBookingClass.getClassCode();
00326
00327     if (iBookingClass.getSubclassCode() != 0) {
00328         oStream << iBookingClass.getSubclassCode();
00329     }
00330
00331     #if BOOST_VERSION >= 104100
00332         // Put boarding point in property tree
00333         ioPropertyTree.put ("flight_date.segment.cabin.fare_family.booking_class.code
00334                             ",
00335                             oStream.str());
00336     #endif
00337
00338     /*
00339     oStream << iBookingClass.getAuthorizationLevel() << " ("
00340             << iBookingClass.getProtection() << "), "
00341             << iBookingClass.getNegotiatedSpace() << ", "
00342             << iBookingClass.getNoShowPercentage() << ", "
00343             << iBookingClass.getOverbookingPercentage() << ", "
00344             << iBookingClass.getNbOfBookings() << ", "
00345             << iBookingClass.getNbOfGroupBookings() << " ("
00346             << iBookingClass.getNbOfPendingGroupBookings() << ")", "
00347             << iBookingClass.getNbOfStaffBookings() << ", "
00348             << iBookingClass.getNbOfWLBookings() << ", "
00349             << iBookingClass.getETB() << ", "
00350             << iBookingClass.getNetClassAvailability() << ", "
00351             << iBookingClass.getNetRevenueAvailability() << ", "

```

```

00356         << iBookingClass.getSegmentAvailability() << ", "
00357         << std::endl;
00358     */
00359 #endif // BOOST_VERSION >= 104100
00360 }
00361
00362 // //////////////////////////////////////
00363 void jsonBookingClassExport (bpt::ptree& ioPropertyTree,
00364                             const FlightDate& iFlightDate) {
00365     // Check whether there are SegmentDate objects
00366     if (BomManager::hasList<SegmentDate> (iFlightDate) == false) {
00367         return;
00368     }
00369
00370     // Browse the segment-dates
00371     const SegmentDateList_T& lSegmentDateList =
00372         BomManager::getList<SegmentDate> (iFlightDate);
00373     for (SegmentDateList_T::const_iterator itSD = lSegmentDateList.begin();
00374          itSD != lSegmentDateList.end(); ++itSD) {
00375         const SegmentDate* lSD_ptr = *itSD;
00376         assert (lSD_ptr != NULL);
00377
00378         // Browse the segment-cabins
00379         const SegmentCabinList_T& lSegmentCabinList =
00380             BomManager::getList<SegmentCabin> (*lSD_ptr);
00381         for (SegmentCabinList_T::const_iterator itSC = lSegmentCabinList.begin();
00382              itSC != lSegmentCabinList.end(); ++itSC) {
00383             const SegmentCabin* lSC_ptr = *itSC;
00384             assert (lSC_ptr != NULL);
00385
00386             // Build the leading string to be displayed
00387             std::ostringstream oLeadingStr;
00388
00389             // Default Fare Family code, when there are no FF
00390             FamilyCode_T lFamilyCode ("NoFF");
00391
00392             // Check whether there are FareFamily objects
00393             if (BomManager::hasList<FareFamily> (*lSC_ptr) == true) {
00394
00395                 // Browse the fare families
00396                 const FareFamilyList_T& lFareFamilyList =
00397                     BomManager::getList<FareFamily> (*lSC_ptr);
00398                 for (FareFamilyList_T::const_iterator itFF = lFareFamilyList.begin();
00399                      itFF != lFareFamilyList.end(); ++itFF) {
00400                     const FareFamily* lFF_ptr = *itFF;
00401                     assert (lFF_ptr != NULL);
00402
00403                     // Retrieve the key of the segment-cabin
00404                     lFamilyCode = lFF_ptr->getFamilyCode();
00405
00406                     // Complete the leading string to be displayed
00407                     oLeadingStr << lFamilyCode << ", ";
00408
00409                     // Browse the booking-classes
00410                     const BookingClassList_T& lBookingClassList =
00411                         BomManager::getList<BookingClass> (*lFF_ptr);
00412                     for (BookingClassList_T::const_iterator itBC =
00413                          lBookingClassList.begin();
00414                          itBC != lBookingClassList.end(); ++itBC) {
00415                         const BookingClass* lBC_ptr = *itBC;
00416                         assert (lBC_ptr != NULL);
00417

```

```

00418         //
00419         jsonBookingClassExport (ioPropertyTree, *lBC_ptr,
00420                                 oLeadingStr.str());
00421     }
00422 }
00423
00424     return;
00425 }
00426 assert (BomManager::hasList<FareFamily> (*lSC_ptr) == false);
00427
00428 // The fare family code is a fake one ('NoFF'), and therefore
00429 // does not vary
00430 oLeadingStr << lFamilyCode << ", ";
00431
00432 // Browse the booking-classes, directly from the segment-cabin object
00433 const BookingClassList_T& lBookingClassList =
00434     BomManager::getList<BookingClass> (*lSC_ptr);
00435 for (BookingClassList_T::const_iterator itBC =
00436     lBookingClassList.begin();
00437     itBC != lBookingClassList.end(); ++itBC) {
00438     const BookingClass* lBC_ptr = *itBC;
00439     assert (lBC_ptr != NULL);
00440
00441     //
00442     jsonBookingClassExport (ioPropertyTree, *lBC_ptr, oLeadingStr.str());
00443 }
00444 }
00445 }
00446 }
00447
00448 // //////////////////////////////////////
00449 void BomJSONExport::jsonExport (std::ostream& oStream,
00450                                 const FlightDate& iFlightDate) {
00451     // Save the formatting flags for the given STL output stream
00452     FlagSaver flagSaver (oStream);
00453
00454     // Create an empty property tree object
00455     bpt::ptree pt;
00456
00457 #if BOOST_VERSION >= 104100
00458
00459     const AirlineCode_T& lAirlineCode = iFlightDate.getAirlineCode();
00460     const FlightNumber_T& lFlightNumber = iFlightDate.getFlightNumber();
00461     const Date_T& lFlightDateDate = iFlightDate.getDepartureDate();
00462
00463     // Put airline code in property tree
00464     pt.put ("flight_date.airline_code", lAirlineCode);
00465
00466     // Put flight number level in property tree
00467     pt.put ("flight_date.flight_number", lFlightNumber);
00468
00469     // Put the flight departure date in property tree
00470     const std::string& lDepartureDateStr =
00471         boost::gregorian::to_simple_string (lFlightDateDate);
00472     pt.put ("flight_date.departure_date", lDepartureDateStr);
00473 #endif // BOOST_VERSION >= 104100
00474
00475     //
00476     jsonLegDateExport (pt, iFlightDate);
00477
00478     //
00479     jsonLegCabinExport (pt, iFlightDate);

```

```

00482
00483     //
00484     //jsonBucketExport (pt, iFlightDate);
00485
00486     //
00487     //jsonFareFamilyExport (pt, iFlightDate);
00488
00489     //
00490     //jsonBookingClassExport (pt, iFlightDate);
00491
00492     #if BOOST_VERSION >= 104100
00493         // Write the property tree into the JSON stream.
00494         write_json (oStream, pt);
00495     #endif // BOOST_VERSION >= 104100
00496     }
00497
00498 }
```

35.169 stdair/bom/BomJSONExport.hpp File Reference

```
#include <iosfwd>
```

Classes

- class [stdair::BomJSONExport](#)

Utility class to export StdAir objects in a JSON format.

Namespaces

- namespace [stdair](#)

Handle on the StdAir library context.

35.170 BomJSONExport.hpp

```

00001 #ifndef __STDAIR_BOM_BOMJSONEXPORT_HPP
00002 #define __STDAIR_BOM_BOMJSONEXPORT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009
00010 namespace stdair {
00011
00012     class FlightDate;
00013
00014     class BomJSONExport {
00015     public:
00016         // ////////////////////////////////// Export support methods //////////////////////////////////
00017         static void jsonExport (std::ostream&, const FlightDate&);
00018     };
00019 }
```

```

00032 }
00033 #endif // __STDAIR_BOM_BOMJSONEXPORT_HPP

```

35.171 stdair/bom/BomJSONImport.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/bom/BomJSONImport.hpp>

```

Namespaces

- namespace [bpt](#)
- namespace [stdair](#)

Handle on the StdAir library context.

35.172 BomJSONImport.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 #if BOOST_VERSION >= 104100
00008 // Boost Property Tree
00009 #include <boost/property_tree/ptree.hpp>
00010 #include <boost/property_tree/json_parser.hpp>
00011 #endif // BOOST_VERSION >= 104100
00012 // StdAir
00013 #include <stdair/bom/BomJSONImport.hpp>
00014
00015 #if BOOST_VERSION >= 104100
00016 namespace bpt = boost::property_tree;
00017 #else // BOOST_VERSION >= 104100
00018 namespace bpt {
00019     typedef char ptree;
00020 }
00021 #endif // BOOST_VERSION >= 104100
00022
00023 namespace stdair {
00024
00025     // //////////////////////////////////////
00026     bool BomJSONImport::jsonImportInventoryKey (const std::string& iBomTree,
00027                                                  AirlineCode_T& ioAirlineCode) {
00028         bool hasKeyBeenSuccessfullyRetrieved = true;
00029
00030         #if BOOST_VERSION >= 104100
00031             // Create an empty property tree object
00032             bpt::ptree pt;
00033
00034             try {
00035
00036                 // Load the JSON formatted string into the property tree.
00037                 // If reading fails (cannot open stream, parse error), an

```

```

00038         // exception is thrown.
00039         std::istringstream iStr (iBomTree);
00040         read_json (iStr, pt);
00041
00042         // Get the airline_code and build an InventoryKey structure.
00043         // If the flight_date.airline_code key is not found, an
00044         // exception is thrown.
00045         ioAirlineCode = pt.get<AirlineCode_T> ("flight_date.airline_code");
00046
00047     } catch (bpt::ptree_error& bptException) {
00048         hasKeyBeenSuccessfullyRetrieved = false;
00049     }
00050 #endif // BOOST_VERSION >= 104100
00051
00052     return hasKeyBeenSuccessfullyRetrieved;
00053 }
00054
00055 // //////////////////////////////////////
00056 bool BomJSONImport::jsonImportFlightDateKey (const std::string& iBomTree,
00057                                               FlightNumber_T& ioFlightNumber,
00058                                               Date_T& ioDepartureDate) {
00059     bool hasKeyBeenSuccessfullyRetrieved = false;
00060
00061     #if BOOST_VERSION >= 104100
00062         // Create an empty property tree object
00063         bpt::ptree pt;
00064
00065         try {
00066
00067             // Load the JSON formatted string into the property tree.
00068             // If reading fails (cannot open stream, parse error), an
00069             // exception is thrown.
00070             std::istringstream iStr (iBomTree);
00071             read_json (iStr, pt);
00072
00073             // Get the flight_number and departure_date and build an
00074             // FlightDateKey structure.
00075             ioFlightNumber = pt.get<FlightNumber_T> ("flight_date.flight_number");
00076
00077             const std::string& lDepartureDateStr =
00078                 pt.get<std::string> ("flight_date.departure_date");
00079             ioDepartureDate = boost::gregorian::from_simple_string (lDepartureDateStr);
00080
00081         } catch (bpt::ptree_error& bptException) {
00082             hasKeyBeenSuccessfullyRetrieved = false;
00083         }
00084     #endif // BOOST_VERSION >= 104100
00085
00086     return hasKeyBeenSuccessfullyRetrieved;
00087 }
00088
00089 }

```

35.173 stdair/bom/BomJSONImport.hpp File Reference

```

#include <string>

#include <stdair/stdair_basic_types.hpp>

```

```
#include <stdair/stdair_date_time_types.hpp>
```

Classes

- class `stdair::BomJSONImport`
Utility class to import StdAir objects in a JSON format.

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

35.174 BomJSONImport.hpp

```
00001 #ifndef __STDAIR_BOM_BOMJSONIMPORT_HPP
00002 #define __STDAIR_BOM_BOMJSONIMPORT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/stdair_basic_types.hpp>
00011 #include <stdair/stdair_date_time_types.hpp>
00012
00013 namespace stdair {
00014
00018     class BomJSONImport {
00019     public:
00020         // ////////////////////////////////// Import support methods //////////////////////////////////
00028         static bool jsonImportInventoryKey (const std::string& iBomKey,
00029                                             AirlineCode_T&);
00030
00039         static bool jsonImportFlightDateKey (const std::string& iBomKey,
00040                                             FlightNumber_T&,
00041                                             Date_T& ioDepartureDate);
00042     };
00043
00044 }
00045 #endif // __STDAIR_BOM_BOMJSONIMPORT_HPP
```

35.175 stdair/bom/BomKeyManager.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/tokenizer.hpp>
#include <boost/lexical_cast.hpp>
#include <boost/date_time/gregorian/parsers.hpp>
```

```

#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/InventoryKey.hpp>
#include <stdair/bom/FlightDateKey.hpp>
#include <stdair/bom/SegmentDateKey.hpp>
#include <stdair/bom/ParsedKey.hpp>
#include <stdair/bom/BomKeyManager.hpp>
#include <stdair/service/Logger.hpp>

```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

Typedefs

- typedef `boost::tokenizer< boost::char_separator< char > > stdair::Tokeniser_T`

35.176 BomKeyManager.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost
00008 #include <boost/tokenizer.hpp>
00009 #include <boost/lexical_cast.hpp>
00010 #include <boost/date_time/gregorian/parsers.hpp>
00011 // StdAir
00012 #include <stdair/stdair_exceptions.hpp>
00013 #include <stdair/basic/BasConst_BomDisplay.hpp>
00014 #include <stdair/bom/InventoryKey.hpp>
00015 #include <stdair/bom/FlightDateKey.hpp>
00016 #include <stdair/bom/SegmentDateKey.hpp>
00017 #include <stdair/bom/ParsedKey.hpp>
00018 #include <stdair/bom/BomKeyManager.hpp>
00019 #include <stdair/service/Logger.hpp>
00020
00021 namespace stdair {
00022
00023 // ////////////////////////////////// Tokenising support //////////////////////////////////
00027 typedef boost::tokenizer<boost::char_separator<char> > Tokeniser_T;
00028
00029 // //////////////////////////////////////
00030 ParsedKey BomKeyManager::extractKeys (const std::string& iFullKeyStr) {
00031     ParsedKey oParsedKey;
00032     oParsedKey._fullKey = iFullKeyStr;
00033 }

```

```

00034     // Token-ise the full key string
00035     Tokeniser_T lTokens (iFullKeyStr, DEFAULT_KEY_TOKEN_DELIMITER);
00036     Tokeniser_T::iterator itToken = lTokens.begin();
00037
00038     // Airline code
00039     if (itToken != lTokens.end()) {
00040         oParsedKey._airlineCode = *itToken;
00041
00042     // Flight number
00043         ++itToken;
00044         if (itToken != lTokens.end()) {
00045             oParsedKey._flightNumber = *itToken;
00046
00047             // Departure date
00048             ++itToken;
00049             if (itToken != lTokens.end()) {
00050                 oParsedKey._departureDate = *itToken;
00051
00052                 // Origin
00053                 ++itToken;
00054                 if (itToken != lTokens.end()) {
00055                     oParsedKey._boardingPoint = *itToken;
00056
00057                     // Destination
00058                     ++itToken;
00059                     if (itToken != lTokens.end()) {
00060                         oParsedKey._offPoint = *itToken;
00061
00062                         // Boarding time
00063                         ++itToken;
00064                         if (itToken != lTokens.end()) {
00065                             oParsedKey._boardingTime = *itToken;
00066                         }
00067                     }
00068                 }
00069             }
00070         }
00071     }
00072
00073     return oParsedKey;
00074 }
00075
00076 // //////////////////////////////////////
00077 InventoryKey BomKeyManager::
00078 extractInventoryKey (const std::string& iFullKeyStr) {
00079     ParsedKey lParsedKey = extractKeys (iFullKeyStr);
00080
00081     return lParsedKey.getInventoryKey();
00082 }
00083
00084 // //////////////////////////////////////
00085 FlightDateKey BomKeyManager::
00086 extractFlightDateKey (const std::string& iFullKeyStr) {
00087     ParsedKey lParsedKey = extractKeys (iFullKeyStr);
00088
00089     return lParsedKey.getFlightDateKey();
00090 }
00091
00092 // //////////////////////////////////////
00093 SegmentDateKey BomKeyManager::
00094 extractSegmentDateKey (const std::string& iFullKeyStr) {
00095     ParsedKey lParsedKey = extractKeys (iFullKeyStr);

```

```

00096
00097     return lParsedKey.getSegmentKey();
00098 }
00099 }

```

35.177 stdair/bom/BomKeyManager.hpp File Reference

```

#include <iosfwd>

#include <stdair/stdair_basic_types.hpp>

```

Classes

- class [stdair::BomKeyManager](#)
Utility class to extract key structures from strings.

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.178 BomKeyManager.hpp

```

00001 #ifndef __STDAIR_BOM_BOMKEYMANAGER_HPP
00002 #define __STDAIR_BOM_BOMKEYMANAGER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 // StdAir
00010 #include <stdair/stdair_basic_types.hpp>
00011
00012 namespace stdair {
00013
00014     struct BomRootKey;
00015     struct InventoryKey;
00016     struct FlightDateKey;
00017     struct LegDateKey;
00018     struct SegmentDateKey;
00019     struct LegCabinKey;
00020     struct SegmentCabinKey;
00021     struct FareFamilyKey;
00022     struct BookingClassKey;
00023     struct ParsedKey;
00024
00025     class BomKeyManager {
00026     public:
00027         // ////////////////////////////////////// Key management support methods //////////////////////////////////////
00028         static ParsedKey extractKeys (const std::string& iFullKeyStr);
00029
00030         static InventoryKey extractInventoryKey (const std::string& iFullKeyStr);
00031
00032     };
00033 }
00034
00035 #endif

```

```

00050
00062     static FlightDateKey extractFlightDateKey (const std::string& iFullKeyStr);
00063
00075     static SegmentDateKey extractSegmentDateKey (const std::string& iFullKeyStr);

00076 };
00077
00078 }
00079 #endif // __STDAIR_BOM_BOMKEYMANAGER_HPP

```

35.179 stdair/bom/BomManager.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <list>
#include <map>
#include <boost/static_assert.hpp>
#include <boost/type_traits/is_same.hpp>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/BomHolder.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/SegmentDate.hpp>

```

Classes

- class [stdair::BomManager](#)
Utility class for StdAir-based objects.

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.180 BomManager.hpp

```

00001 #ifndef __STDAIR_BOM_BOMMANAGER_HPP
00002 #define __STDAIR_BOM_BOMMANAGER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>

```

```

00010 #include <list>
00011 #include <map>
00012 // Boost
00013 #include <boost/static_assert.hpp>
00014 #include <boost/type_traits/is_same.hpp>
00015 // StdAir
00016 #include <stdair/stdair_exceptions.hpp>
00017 #include <stdair/bom/BomAbstract.hpp>
00018 #include <stdair/bom/BomHolder.hpp>
00019 #include <stdair/service/Logger.hpp>
00020 // Stdair BOM Objects
00021 #include <stdair/bom/SegmentDate.hpp>
00022
00023 namespace stdair {
00024
00032     class BomManager {
00033     friend class FacBomManager;
00034
00035     public:
00039         template <typename OBJECT2, typename OBJECT1>
00040         static const typename BomHolder<OBJECT2>::BomList_T& getList(const OBJECT1&);
00041
00045         template <typename OBJECT2, typename OBJECT1>
00046         static const typename BomHolder<OBJECT2>::BomMap_T& getMap (const OBJECT1&);
00047
00051         template <typename OBJECT2, typename OBJECT1>
00052         static bool hasList (const OBJECT1&);
00053
00057         template <typename OBJECT2, typename OBJECT1>
00058         static bool hasMap (const OBJECT1&);
00059
00065         template <typename PARENT, typename CHILD>
00066         static PARENT* getParentPtr (const CHILD&);
00067
00071         template <typename PARENT, typename CHILD>
00072         static PARENT& getParent (const CHILD&);
00073
00079         template <typename OBJECT2, typename OBJECT1>
00080         static OBJECT2* getObjectPtr (const OBJECT1&, const MapKey_T&);
00081
00085         template <typename OBJECT2, typename OBJECT1>
00086         static OBJECT2& getObject (const OBJECT1&, const MapKey_T&);
00087
00088     private:
00094         template <typename OBJECT2, typename OBJECT1>
00095         static const BomHolder<OBJECT2>& getBomHolder (const OBJECT1&);
00096     };
00097
00098     // //////////////////////////////////////
00099     // Private method.
00100     template <typename OBJECT2, typename OBJECT1>
00101     const BomHolder<OBJECT2>& BomManager::getBomHolder (const OBJECT1& iObject1) {
00102
00103         const HolderMap_T& lHolderMap = iObject1.getHolderMap();
00104
00105         HolderMap_T::const_iterator itHolder = lHolderMap.find (&typeid (OBJECT2));
00106
00107         if (itHolder == lHolderMap.end()) {
00108             const std::string lName (typeid (OBJECT2).name());
00109             throw NonInitialisedContainerException ("Cannot find the holder of type "

```

```

00110         + lName + " within: "
00111         + iObject1.describeKey());
00112     }
00113
00114     const BomHolder<OBJECT2>* lBomHolder_ptr =
00115         static_cast<const BomHolder<OBJECT2>*> (itHolder->second);
00116     assert (lBomHolder_ptr != NULL);
00117
00118     return *lBomHolder_ptr;
00119 }
00120
00121 // //////////////////////////////////////
00122 // Public business method.
00123 // This method is specialized for the following couple types:
00124 // <SegmentDate, SegmentDate>
00125 template <typename OBJECT2, typename OBJECT1>
00126 const typename BomHolder<OBJECT2>::BomList_T& BomManager::
00127 getList (const OBJECT1& iObject1) {
00128
00129     const BomHolder<OBJECT2>& lBomHolder = getBomHolder<OBJECT2> (iObject1);
00130     return lBomHolder._bomList;
00131 }
00132
00133 // //////////////////////////////////////
00134 // Public business method.
00135 // Compile time assertion to check OBJECT1 and OBJECT2 types.
00136 template <typename OBJECT2, typename OBJECT1>
00137 const typename BomHolder<OBJECT2>::BomMap_T& BomManager::
00138 getMap (const OBJECT1& iObject1) {
00139
00140     //
00141     // Compile time assertion: this function must never be called with the
00142     // following list of couple types:
00143     // <SegmentDate, SegmentDate>
00144     //
00145     BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, SegmentDate>::value == false
00146         || boost::is_same<OBJECT2, SegmentDate>::value == false
00147     ));
00148
00149     const BomHolder<OBJECT2>& lBomHolder = getBomHolder<OBJECT2> (iObject1);
00150     return lBomHolder._bomMap;
00151 }
00152
00153 // //////////////////////////////////////
00154 // Public business method.
00155 // This method is specialized for the following couple types:
00156 // <SegmentDate, SegmentDate>
00157 template <typename OBJECT2, typename OBJECT1>
00158 bool BomManager::hasList (const OBJECT1& iObject1) {
00159
00160     const HolderMap_T& lHolderMap = iObject1.getHolderMap();
00161     HolderMap_T::const_iterator itHolder = lHolderMap.find (&typeid (OBJECT2));
00162
00163     if (itHolder == lHolderMap.end()) {
00164         return false;
00165     }
00166
00167     const BomHolder<OBJECT2>* lBomHolder_ptr =
00168         static_cast<const BomHolder<OBJECT2>*> (itHolder->second);
00169     assert (lBomHolder_ptr != NULL);
00170
00171     return !lBomHolder_ptr->_bomList.empty();
00172 }

```

```

00171
00172 // //////////////////////////////////////
00173 // Public business method.
00174 // This method is specialized for the following couple types:
00175 // <SegmentDate, SegmentDate>
00176 template <typename OBJECT2, typename OBJECT1>
00177 bool BomManager::hasMap (const OBJECT1& iObject1) {
00178
00179     const HolderMap_T& lHolderMap = iObject1.getHolderMap();
00180     HolderMap_T::const_iterator itHolder = lHolderMap.find (&typeid (OBJECT2));
00181
00182     if (itHolder == lHolderMap.end()) {
00183         return false;
00184     }
00185     const BomHolder<OBJECT2>* lBomHolder_ptr =
00186         static_cast<const BomHolder<OBJECT2>*> (itHolder->second);
00187     assert (lBomHolder_ptr != NULL);
00188
00189     return !lBomHolder_ptr->_bomMap.empty();
00190 }
00191
00192 // //////////////////////////////////////
00193 // Public business method valid for all PARENT and CHILD types.
00194 // (No compile time assertion to check PARENT and CHILD types.)
00195 template <typename PARENT, typename CHILD>
00196 PARENT* BomManager::getParentPtr (const CHILD& iChild) {
00197
00198     PARENT* const lParent_ptr = static_cast<PARENT* const> (iChild.getParent());
00199     return lParent_ptr;
00200 }
00201
00202 // //////////////////////////////////////
00203 // Public business method valid for all PARENT and CHILD types.
00204 // (No compile time assertion to check PARENT and CHILD types.)
00205 template <typename PARENT, typename CHILD>
00206 PARENT& BomManager::getParent (const CHILD& iChild) {
00207
00208     PARENT* const lParent_ptr = getParentPtr<PARENT> (iChild);
00209     assert (lParent_ptr != NULL);
00210     return *lParent_ptr;
00211 }
00212
00213 // //////////////////////////////////////
00214 // Public business method.
00215 // Compile time assertion to check OBJECT1 and OBJECT2 types.
00216 template <typename OBJECT2, typename OBJECT1>
00217 OBJECT2* BomManager::getObjectPtr (const OBJECT1& iObject1,
00218                                     const MapKey_T& iKey) {
00219
00220     //
00221     // Compile time assertion: this function must never be called with the
00222     // following list of couple types:
00223     // <SegmentDate, SegmentDate>
00224     //
00225     BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, SegmentDate>::value == false
00226                             || boost::is_same<OBJECT2, SegmentDate>::value == false
00227     ));
00228
00229     OBJECT2* oBom_ptr = NULL;
00230
00231     const HolderMap_T& lHolderMap = iObject1.getHolderMap();

```

```

00232     typename HolderMap_T::const_iterator itHolder =
00233         lHolderMap.find (&typeid (OBJECT2));
00234
00235     if (itHolder != lHolderMap.end()) {
00236
00237         BomHolder<OBJECT2>* const lBomHolder_ptr =
00238             static_cast<BomHolder<OBJECT2>* const> (itHolder->second);
00239         assert (lBomHolder_ptr != NULL);
00240
00241         //
00242         typedef typename BomHolder<OBJECT2>::BomMap_T BomMap_T;
00243         BomMap_T& lBomMap = lBomHolder_ptr->_bomMap;
00244         typename BomMap_T::iterator itBom = lBomMap.find (iKey);
00245
00246         if (itBom != lBomMap.end()) {
00247             oBom_ptr = itBom->second;
00248             assert (oBom_ptr != NULL);
00249         }
00250     }
00251
00252     return oBom_ptr;
00253 }
00254
00255 // //////////////////////////////////////
00256 // Public business method.
00257 // Compile time assertion to check OBJECT1 and OBJECT2 types.
00258 template <typename OBJECT2, typename OBJECT1>
00259 OBJECT2& BomManager::getObject (const OBJECT1& iObject1,
00260                                 const MapKey_T& iKey) {
00261
00262     //
00263     // Compile time assertion: this function must never be called with the
00264     // following list of couple types:
00265     // <SegmentDate, SegmentDate>
00266     //
00267     BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, SegmentDate>::value == false
00268         || boost::is_same<OBJECT2, SegmentDate>::value == false
00269     ));
00270
00271     OBJECT2* oBom_ptr = NULL;
00272
00273     typedef std::map<const MapKey_T, OBJECT2*> BomMap_T;
00274     const BomMap_T& lBomMap = getMap<OBJECT2> (iObject1);
00275
00276     typename BomMap_T::const_iterator itBom = lBomMap.find (iKey);
00277
00278     if (itBom == lBomMap.end()) {
00279         const std::string lName (typeid (OBJECT2).name());
00280
00281         STDAIR_LOG_ERROR ("Cannot find the objet of type " << lName
00282             << " with key " << iKey << " within: "
00283             << iObject1.describeKey());
00284         assert (false);
00285     }
00286
00287     oBom_ptr = itBom->second;
00288     assert (oBom_ptr != NULL);
00289
00290     return *oBom_ptr;
00291 }
00292 // //////////////////////////////////////

```

```

00293 //
00294 // Specialization of the template methods above for a segment
00295 // date and its corresponding marketing segment dates.
00296 //
00297 // //////////////////////////////////////
00298
00299 // Specialization of the template method hasList above for the types
00300 // <SegmentDate, SegmentDate>.
00301 // Add an element to the marketing segment date list of a segment date.
00302 template<>
00303 inline bool BomManager::hasList<SegmentDate,SegmentDate>
00304 (const SegmentDate& ioSegmentDate) {
00305
00306     const SegmentDateList_T& lMarketingSegmentDateList =
00307         ioSegmentDate.getMarketingSegmentDateList ();
00308     const bool isMarketingSegmentDateListEmpty =
00309         lMarketingSegmentDateList.empty();
00310     return isMarketingSegmentDateListEmpty;
00311 }
00312
00313 // Specialization of the template method hasList above for the types
00314 // <SegmentDate, SegmentDate>.
00315 // Return a boolean saying if the marketing segment date list is empty
00316 // or not.
00317 template<>
00318 inline const BomHolder<SegmentDate>::BomList_T&
00319 BomManager::getList<SegmentDate,SegmentDate> (const SegmentDate& ioSegmentDate)
00320 {
00321     const SegmentDateList_T& lMarketingSegmentDateList =
00322         ioSegmentDate.getMarketingSegmentDateList ();
00323     return lMarketingSegmentDateList;
00324 }
00325
00326 // Specialization of the template method hasMap above for the types
00327 // <SegmentDate, SegmentDate>.
00328 // A segment date does not have a Segment Date Map but it can have a
00329 // Segment Date list (containing its marketing segment dates).
00330 template<>
00331 inline bool BomManager::hasMap<SegmentDate,SegmentDate>
00332 (const SegmentDate& ioSegmentDate) {
00333
00334     const bool hasList = false;
00335     return hasList;
00336 }
00337
00338 }
00339 #endif // __STDAIR_BOM_BOMMANAGER_HPP

```

35.181 stdair/bom/BomRetriever.cpp File Reference

```

#include <cassert>

#include <sstream>

#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomKeyManager.hpp>
#include <stdair/bom/BomManager.hpp>

```

```

#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/BomRetriever.hpp>
#include <stdair/bom/ParsedKey.hpp>
#include <stdair/bom/AirportPair.hpp>
#include <stdair/service/Logger.hpp>

```

Namespaces

- namespace `stdair`

Handle on the StdAir library context.

35.182 BomRetriever.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_Inventory.hpp>
00009 #include <stdair/bom/BomKeyManager.hpp>
00010 #include <stdair/bom/BomManager.hpp>
00011 #include <stdair/bom/BomRoot.hpp>
00012 #include <stdair/bom/Inventory.hpp>
00013 #include <stdair/bom/FlightDate.hpp>
00014 #include <stdair/bom/LegDate.hpp>
00015 #include <stdair/bom/SegmentDate.hpp>
00016 #include <stdair/bom/LegCabin.hpp>
00017 #include <stdair/bom/SegmentCabin.hpp>
00018 #include <stdair/bom/FareFamily.hpp>
00019 #include <stdair/bom/BookingClass.hpp>
00020 #include <stdair/bom/BomRetriever.hpp>
00021 #include <stdair/bom/ParsedKey.hpp>
00022 #include <stdair/bom/AirportPair.hpp>
00023 #include <stdair/service/Logger.hpp>
00024
00025 namespace stdair {
00026
00027 // //////////////////////////////////////

```

```

00028     Inventory* BomRetriever::
00029     retrieveInventoryFromLongKey (const BomRoot& iBomRoot,
00030                                   const std::string& iFullKeyStr) {
00031         Inventory* oInventory_ptr = NULL;
00032
00033         // Extract the inventory key (i.e., airline code)
00034         const InventoryKey& lInventoryKey =
00035             BomKeyManager::extractInventoryKey (iFullKeyStr);
00036
00037         oInventory_ptr = iBomRoot.getInventory (lInventoryKey);
00038
00039         return oInventory_ptr;
00040     }
00041
00042     // //////////////////////////////////////
00043     Inventory* BomRetriever::retrieveInventoryFromKey (const BomRoot& iBomRoot,
00044                                                         const InventoryKey& iKey) {
00045         Inventory* oInventory_ptr = NULL;
00046
00047         //
00048         oInventory_ptr = iBomRoot.getInventory (iKey);
00049
00050         return oInventory_ptr;
00051     }
00052
00053     // //////////////////////////////////////
00054     Inventory* BomRetriever::
00055     retrieveInventoryFromKey (const BomRoot& iBomRoot,
00056                             const AirlineCode_T& iAirlineCode) {
00057         Inventory* oInventory_ptr = NULL;
00058
00059         //
00060         const InventoryKey lKey (iAirlineCode);
00061         oInventory_ptr = iBomRoot.getInventory (lKey);
00062
00063         return oInventory_ptr;
00064     }
00065
00066     // //////////////////////////////////////
00067     FlightDate* BomRetriever::
00068     retrieveFlightDateFromLongKey (const BomRoot& iBomRoot,
00069                                   const std::string& iFullKeyStr) {
00070         FlightDate* oFlightDate_ptr = NULL;
00071
00072         // Retrieve the inventory
00073         Inventory* oInventory_ptr =
00074             BomRetriever::retrieveInventoryFromLongKey (iBomRoot, iFullKeyStr);
00075         if (oInventory_ptr == NULL) {
00076             return oFlightDate_ptr;
00077         }
00078         assert (oInventory_ptr != NULL);
00079
00080         // Extract the flight-date key (i.e., flight number and date)
00081         const FlightDateKey& lFlightDateKey =
00082             BomKeyManager::extractFlightDateKey (iFullKeyStr);
00083
00084         oFlightDate_ptr = oInventory_ptr->getFlightDate (lFlightDateKey);
00085
00086         return oFlightDate_ptr;
00087     }
00088
00089     // //////////////////////////////////////

```

```

00090  FlightDate* BomRetriever::
00091  retrieveFlightDateFromKeySet (const BomRoot& iBomRoot,
00092                               const AirlineCode_T& iAirlineCode,
00093                               const FlightNumber_T& iFlightNumber,
00094                               const Date_T& iFlightDateDate) {
00095      FlightDate* oFlightDate_ptr = NULL;
00096
00097      // Retrieve the inventory
00098      Inventory* oInventory_ptr =
00099      BomRetriever::retrieveInventoryFromKey (iBomRoot, iAirlineCode);
00100      if (oInventory_ptr == NULL) {
00101          return oFlightDate_ptr;
00102      }
00103      assert (oInventory_ptr != NULL);
00104
00105      //
00106      oFlightDate_ptr = retrieveFlightDateFromKey (*oInventory_ptr,
00107                                                    iFlightNumber, iFlightDateDate);
00108
00109      return oFlightDate_ptr;
00110  }
00111
00112  // //////////////////////////////////////
00113  FlightDate* BomRetriever::
00114  retrieveFlightDateFromLongKey (const Inventory& iInventory,
00115                                const std::string& iFullKeyStr) {
00116      FlightDate* oFlightDate_ptr = NULL;
00117
00118      // Extract the flight-date key (i.e., flight number and date)
00119      const FlightDateKey& lFlightDateKey =
00120      BomKeyManager::extractFlightDateKey (iFullKeyStr);
00121
00122      oFlightDate_ptr = iInventory.getFlightDate (lFlightDateKey);
00123
00124      return oFlightDate_ptr;
00125  }
00126
00127  // //////////////////////////////////////
00128  FlightDate* BomRetriever::
00129  retrieveFlightDateFromKey (const Inventory& iInventory,
00130                             const FlightDateKey& iKey) {
00131      FlightDate* oFlightDate_ptr = NULL;
00132
00133      //
00134      oFlightDate_ptr = iInventory.getFlightDate (iKey);
00135
00136      return oFlightDate_ptr;
00137  }
00138
00139  // //////////////////////////////////////
00140  FlightDate* BomRetriever::
00141  retrieveFlightDateFromKey (const Inventory& iInventory,
00142                             const FlightNumber_T& iFlightNumber,
00143                             const Date_T& iFlightDateDate) {
00144      FlightDate* oFlightDate_ptr = NULL;
00145
00146      //
00147      const FlightDateKey lKey (iFlightNumber, iFlightDateDate);
00148      oFlightDate_ptr = iInventory.getFlightDate (lKey);
00149
00150      return oFlightDate_ptr;

```

```

00151     }
00152
00153     // //////////////////////////////////////
00154     SegmentDate* BomRetriever::
00155     retrieveSegmentDateFromLongKey (const BomRoot& iBomRoot,
00156                                     const std::string& iFullKeyStr) {
00157         SegmentDate* oSegmentDate_ptr = NULL;
00158
00159         // Retrieve the flight-date
00160         FlightDate* oFlightDate_ptr =
00161             BomRetriever::retrieveFlightDateFromLongKey (iBomRoot, iFullKeyStr);
00162         if (oFlightDate_ptr == NULL) {
00163             return oSegmentDate_ptr;
00164         }
00165         assert (oFlightDate_ptr != NULL);
00166
00167         // Extract the segment-date key (i.e., origin and destination)
00168         const SegmentDateKey& lSegmentDateKey =
00169             BomKeyManager::extractSegmentDateKey (iFullKeyStr);
00170
00171         oSegmentDate_ptr = oFlightDate_ptr->getSegmentDate (lSegmentDateKey);
00172
00173         return oSegmentDate_ptr;
00174     }
00175
00176     // //////////////////////////////////////
00177     SegmentDate* BomRetriever::
00178     retrieveSegmentDateFromLongKey (const Inventory& iInventory,
00179                                     const std::string& iFullKeyStr) {
00180         SegmentDate* oSegmentDate_ptr = NULL;
00181
00182         ParsedKey lParsedKey = BomKeyManager::extractKeys (iFullKeyStr);
00183
00184         if (iInventory.getAirlineCode() != lParsedKey.__airlineCode) {
00185             STDAIR_LOG_DEBUG ("Airline code: " << lParsedKey.__airlineCode);
00186             return oSegmentDate_ptr;
00187         }
00188
00189         FlightDate* lFlightDate_ptr =
00190             retrieveFlightDateFromKey (iInventory, lParsedKey.getFlightDateKey());
00191         if (lFlightDate_ptr == NULL) {
00192             STDAIR_LOG_DEBUG ("Flight-date key: "
00193                             << lParsedKey.getFlightDateKey().toString());
00194             return oSegmentDate_ptr;
00195         }
00196
00197         oSegmentDate_ptr =
00198             retrieveSegmentDateFromKey (*lFlightDate_ptr, lParsedKey.getSegmentKey());
00199         if (oSegmentDate_ptr == NULL) {
00200             STDAIR_LOG_DEBUG ("Segment-date key: "
00201                             << lParsedKey.getSegmentKey().toString());
00202             return oSegmentDate_ptr;
00203         }
00204
00205         return oSegmentDate_ptr;
00206     }
00207
00208     // //////////////////////////////////////
00209     SegmentDate* BomRetriever::
00210     retrieveSegmentDateFromLongKey (const FlightDate& iFlightDate,
00211                                     const std::string& iFullKeyStr) {
00212         SegmentDate* oSegmentDate_ptr = NULL;

```

```

00213
00214     // Extract the segment-date key (i.e., origin and destination)
00215     const SegmentDateKey& lSegmentDateKey =
00216         BomKeyManager::extractSegmentDateKey (iFullKeyStr);
00217
00218     oSegmentDate_ptr = iFlightDate.getSegmentDate (lSegmentDateKey);
00219
00220     return oSegmentDate_ptr;
00221 }
00222
00223 // //////////////////////////////////////
00224 SegmentDate* BomRetriever::
00225 retrieveSegmentDateFromKey (const FlightDate& iFlightDate,
00226                             const SegmentDateKey& iKey) {
00227     SegmentDate* oSegmentDate_ptr = NULL;
00228
00229     //
00230     oSegmentDate_ptr = iFlightDate.getSegmentDate (iKey);
00231
00232     return oSegmentDate_ptr;
00233 }
00234
00235 // //////////////////////////////////////
00236 SegmentDate* BomRetriever::
00237 retrieveSegmentDateFromKey (const FlightDate& iFlightDate,
00238                             const AirportCode_T& iOrigin,
00239                             const AirportCode_T& iDestination) {
00240     SegmentDate* oSegmentDate_ptr = NULL;
00241
00242     //
00243     const SegmentDateKey lKey (iOrigin, iDestination);
00244     oSegmentDate_ptr = iFlightDate.getSegmentDate (lKey);
00245
00246     return oSegmentDate_ptr;
00247 }
00248
00249 // //////////////////////////////////////
00250 BookingClass* BomRetriever::
00251 retrieveBookingClassFromLongKey (const Inventory& iInventory,
00252                                 const std::string& iFullKeyStr,
00253                                 const ClassCode_T& iClassCode) {
00254     BookingClass* oBookingClass_ptr = NULL;
00255
00256     SegmentDate* lSegmentDate_ptr = retrieveSegmentDateFromLongKey (iInventory,
00257                                                                     iFullKeyStr);
00258
00259     if (lSegmentDate_ptr == NULL) {
00260         return oBookingClass_ptr;
00261     }
00262     assert (lSegmentDate_ptr != NULL);
00263
00264     //
00265     oBookingClass_ptr =
00266         BomManager::getObjectPtr<BookingClass> (*lSegmentDate_ptr, iClassCode);
00267
00268     return oBookingClass_ptr;
00269 }
00270
00271 // //////////////////////////////////////
00272 AirportPair* BomRetriever::
00273 retrieveAirportPairFromKeySet (const BomRoot& iBomRoot,

```

```

00274         const stdair::AirportCode_T& iOrigin,
00275         const stdair::AirportCode_T& iDestination) {
00276
00277     // Get the Airport pair stream of the segment path.
00278     const AirportPairKey lAirportPairKey (iOrigin, iDestination);
00279
00280     // Search for the fare rules having the same origin and
00281     // destination airport as the travel solution
00282     AirportPair* oAirportPair_ptr = BomManager::
00283         getObjectPtr<AirportPair> (iBomRoot, lAirportPairKey.toString());
00284
00285     return oAirportPair_ptr;
00286 }
00287
00288 // //////////////////////////////////////
00289 void BomRetriever::
00290 retrieveDatePeriodListFromKey (const AirportPair& iAirportPair,
00291     const stdair::Date_T& iDepartureDate,
00292     stdair::DatePeriodList_T& ioDatePeriodList) {
00293
00294     // Get the list of date-period
00295     const DatePeriodList_T& lFareDatePeriodList =
00296         BomManager::getList<DatePeriod> (iAirportPair);
00297
00298     // Browse the date-period list
00299     for (DatePeriodList_T::const_iterator itDateRange =
00300         lFareDatePeriodList.begin();
00301         itDateRange != lFareDatePeriodList.end(); ++itDateRange) {
00302
00303         DatePeriod* lCurrentFareDatePeriod_ptr = *itDateRange ;
00304         assert (lCurrentFareDatePeriod_ptr != NULL);
00305
00306         // Select the date-period objects having a corresponding date range
00307         const bool isDepartureDateValid =
00308             lCurrentFareDatePeriod_ptr->isDepartureDateValid (iDepartureDate);
00309
00310         // Add the date-period objects having a corresponding date range
00311         // to the list to display
00312         if (isDepartureDateValid == true) {
00313             ioDatePeriodList.push_back(lCurrentFareDatePeriod_ptr);
00314         }
00315     }
00316 }
00317
00318 // //////////////////////////////////////
00319 void BomRetriever::
00320 retrieveDatePeriodListFromKeySet (const BomRoot& iBomRoot,
00321     const stdair::AirportCode_T& iOrigin,
00322     const stdair::AirportCode_T& iDestination,
00323     const stdair::Date_T& iDepartureDate,
00324     stdair::DatePeriodList_T& ioDatePeriodList) {
00325
00326     // Retrieve the airport-pair
00327     AirportPair* oAirportPair_ptr =
00328         BomRetriever::retrieveAirportPairFromKeySet (iBomRoot, iOrigin,
00329             iDestination);
00330
00331     if (oAirportPair_ptr == NULL) {
00332         return;
00333     }
00334 }

```

```

00335     assert (oAirportPair_ptr != NULL);
00336
00337     // Retrieve the flight date
00338     BomRetriever::retrieveDatePeriodListFromKey (*oAirportPair_ptr, iDepartureDat
e,
00339                                                 ioDatePeriodList);
00340
00341 }
00342
00343 // //////////////////////////////////////
00344 LegCabin& BomRetriever::
00345 retrieveDummyLegCabin (stdair::BomRoot& iBomRoot) {
00346     LegCabin* oLegCabin_ptr = NULL;
00347
00348     // Retrieve the Inventory
00349     const Inventory* lInventory_ptr = BomRetriever::
00350         retrieveInventoryFromKey (iBomRoot, DEFAULT_AIRLINE_CODE);
00351
00352     if (lInventory_ptr == NULL) {
00353         std::ostringstream oStr;
00354         oStr << "The inventory corresponding to the '"
00355             << DEFAULT_AIRLINE_CODE << "' airline can not be found";
00356         throw ObjectNotFoundException (oStr.str());
00357     }
00358
00359     // Retrieve the FlightDate
00360     const FlightDate* lFlightDate_ptr = BomRetriever::
00361         retrieveFlightDateFromKey (*lInventory_ptr, DEFAULT_FLIGHT_NUMBER,
00362                                   DEFAULT_DEPARTURE_DATE);
00363
00364     if (lFlightDate_ptr == NULL) {
00365         std::ostringstream oStr;
00366         oStr << "The flight-date corresponding to ("
00367             << DEFAULT_FLIGHT_NUMBER << ", "
00368             << DEFAULT_DEPARTURE_DATE << ") can not be found";
00369         throw ObjectNotFoundException (oStr.str());
00370     }
00371
00372     // Retrieve the LegDate
00373     const LegDateKey lLegDateKey (DEFAULT_ORIGIN);
00374     const LegDate* lLegDate_ptr =
00375         lFlightDate_ptr->getLegDate (lLegDateKey);
00376
00377     if (lLegDate_ptr == NULL) {
00378         std::ostringstream oStr;
00379         oStr << "The leg-date corresponding to the '"
00380             << DEFAULT_ORIGIN << "' origin can not be found";
00381         throw ObjectNotFoundException (oStr.str());
00382     }
00383
00384     // Retrieve the LegCabin
00385     const LegCabinKey lLegCabinKey (DEFAULT_CABIN_CODE);
00386     oLegCabin_ptr = lLegDate_ptr->getLegCabin (lLegCabinKey);
00387
00388     if (oLegCabin_ptr == NULL) {
00389         std::ostringstream oStr;
00390         oStr << "The leg-cabin corresponding to the '"
00391             << DEFAULT_CABIN_CODE << "' cabin code can not be found";
00392         throw ObjectNotFoundException (oStr.str());
00393     }
00394 }
00395

```

```

00396     assert (oLegCabin_ptr != NULL);
00397     return *oLegCabin_ptr;
00398 }
00399 }
00400
00401 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00402 SegmentCabin& BomRetriever::
00403 retrieveDummySegmentCabin (stdair::BomRoot& iBomRoot) {
00404
00405     SegmentCabin* oSegmentCabin_ptr = NULL;
00406
00407     // Retrieve the Inventory
00408     const Inventory* lInventory_ptr = BomRetriever::
00409         retrieveInventoryFromKey (iBomRoot, DEFAULT_AIRLINE_CODE);
00410
00411     if (lInventory_ptr == NULL) {
00412         std::ostringstream oStr;
00413         oStr << "The inventory corresponding to the '"
00414             << DEFAULT_AIRLINE_CODE << "' airline can not be found";
00415         throw ObjectNotFoundException (oStr.str());
00416     }
00417
00418     // Retrieve the FlightDate
00419     const FlightDate* lFlightDate_ptr = BomRetriever::
00420         retrieveFlightDateFromKey (*lInventory_ptr, DEFAULT_FLIGHT_NUMBER,
00421             DEFAULT_DEPARTURE_DATE);
00422
00423     if (lFlightDate_ptr == NULL) {
00424         std::ostringstream oStr;
00425         oStr << "The flight-date corresponding to ("
00426             << DEFAULT_FLIGHT_NUMBER << ", "
00427             << DEFAULT_DEPARTURE_DATE << ") can not be found";
00428         throw ObjectNotFoundException (oStr.str());
00429     }
00430
00431     // Retrieve the SegmentDate
00432     const SegmentDateKey lSegmentDateKey (DEFAULT_ORIGIN, DEFAULT_DESTINATION);
00433     const SegmentDate* lSegmentDate_ptr =
00434         lFlightDate_ptr->getSegmentDate (lSegmentDateKey);
00435
00436     if (lSegmentDate_ptr == NULL) {
00437         std::ostringstream oStr;
00438         oStr << "The segment-date corresponding to the '"
00439             << DEFAULT_ORIGIN << "' origin and '"
00440             << DEFAULT_DESTINATION << "' destination can not be found";
00441         throw ObjectNotFoundException (oStr.str());
00442     }
00443
00444     // Retrieve the SegmentCabin
00445     const SegmentCabinKey lSegmentCabinKey (DEFAULT_CABIN_CODE);
00446     oSegmentCabin_ptr =
00447         BomManager::getObjectPtr<SegmentCabin> (*lSegmentDate_ptr, lSegmentCabinKey
00448             .toString());
00449
00450     if (oSegmentCabin_ptr == NULL) {
00451         std::ostringstream oStr;
00452         oStr << "The segment-cabin corresponding to the '"
00453             << DEFAULT_CABIN_CODE << "' cabin code can not be found";
00454         throw ObjectNotFoundException (oStr.str());
00455     }
00456     assert (oSegmentCabin_ptr != NULL);

```

```

00457     return *oSegmentCabin_ptr;
00458 }
00459
00460 }
```

35.183 stdair/bom/BomRetriever.hpp File Reference

```

#include <iosfwd>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/bom/DatePeriod.hpp>
```

Classes

- class [stdair::BomRetriever](#)
Utility class to retrieve StdAir objects.

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.184 BomRetriever.hpp

```

00001 #ifndef __STDAIR_BOM_BOMRETRIEVER_HPP
00002 #define __STDAIR_BOM_BOMRETRIEVER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 // StdAir
00010 #include <stdair/stdair_basic_types.hpp>
00011 #include <stdair/stdair_date_time_types.hpp>
00012 #include <stdair/bom/DatePeriod.hpp>
00013
00014 namespace stdair {
00015
00017     class BomRoot;
00018     struct InventoryKey;
00019     class Inventory;
00020     struct FlightDateKey;
00021     class FlightDate;
00022     class LegDate;
00023     struct SegmentDateKey;
00024     class SegmentDate;
00025     class LegCabin;
00026     class SegmentCabin;
00027     class FareFamily;
00028     class BookingClass;
```

```
00029     class DatePeriod;
00030     class AirportPair;
00031
00035     class BomRetriever {
00036     public:
00037         // //////////// Key management support methods ////////////
00050         static Inventory*
00051         retrieveInventoryFromLongKey (const BomRoot&,
00052                                     const std::string& iFullKeyStr);
00053
00061         static Inventory* retrieveInventoryFromKey (const BomRoot&,
00062                                                    const InventoryKey&);
00063
00071         static Inventory* retrieveInventoryFromKey (const BomRoot&,
00072                                                    const AirlineCode_T&);
00073
00086         static FlightDate*
00087         retrieveFlightDateFromLongKey (const BomRoot&,
00088                                       const std::string& iFullKeyStr);
00089
00099         static FlightDate*
00100         retrieveFlightDateFromKeySet (const BomRoot&,
00101                                     const AirlineCode_T&, const FlightNumber_T&,
00102                                     const Date_T& iFlightDateDate);
00103
00116         static FlightDate*
00117         retrieveFlightDateFromLongKey (const Inventory&,
00118                                       const std::string& iFullKeyStr);
00119
00127         static FlightDate* retrieveFlightDateFromKey (const Inventory&,
00128                                                       const FlightDateKey&);
00129
00138         static FlightDate* retrieveFlightDateFromKey (const Inventory&,
00139                                                       const FlightNumber_T&,
00140                                                       const Date_T& iFlightDateDate);
00141
00154         static SegmentDate*
00155         retrieveSegmentDateFromLongKey (const BomRoot&,
00156                                       const std::string& iFullKeyStr);
00157
00170         static SegmentDate*
00171         retrieveSegmentDateFromLongKey (const Inventory&,
00172                                       const std::string& iFullKeyStr);
00173
00186         static SegmentDate*
00187         retrieveSegmentDateFromLongKey (const FlightDate&,
00188                                       const std::string& iFullKeyStr);
00189
00197         static SegmentDate* retrieveSegmentDateFromKey (const FlightDate&,
00198                                                         const SegmentDateKey&);
00199
00208         static SegmentDate*
00209         retrieveSegmentDateFromKey (const FlightDate&,
00210                                    const AirportCode_T& iOrigin,
00211                                    const AirportCode_T& iDestination);
00212
00236         static BookingClass*
00237         retrieveBookingClassFromLongKey (const Inventory&,
00238                                         const std::string& iFullKeyStr,
00239                                         const ClassCode_T&);
00240
```

```

00241
00250     static AirportPair*
00251     retrieveAirportPairFromKeySet (const BomRoot& ,
00252                                   const stdair::AirportCode_T&,
00253                                   const stdair::AirportCode_T&);
00254
00264     static void
00265     retrieveDatePeriodListFromKey (const AirportPair&,
00266                                   const stdair::Date_T&,
00267                                   stdair::DatePeriodList_T&);
00268
00281     static void
00282     retrieveDatePeriodListFromKeySet (const BomRoot&,
00283                                       const stdair::AirportCode_T&,
00284                                       const stdair::AirportCode_T&,
00285                                       const stdair::Date_T&,
00286                                       stdair::DatePeriodList_T&);
00287
00293     static stdair::LegCabin& retrieveDummyLegCabin (stdair::BomRoot&);
00294
00300     static stdair::SegmentCabin& retrieveDummySegmentCabin (stdair::BomRoot&);
00301
00302 };
00303
00304 }
00305 #endif // __STDAIR_BOM_BOMRETRIEVER_HPP

```

35.185 stdair/bom/BomRoot.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/InventoryKey.hpp>
#include <stdair/bom/Inventory.hpp>

```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.186 BomRoot.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir

```

```

00008 #include <stdair/basic/BasConst_General.hpp>
00009 #include <stdair/bom/BomManager.hpp>
00010 #include <stdair/bom/BomRoot.hpp>
00011 #include <stdair/bom/InventoryKey.hpp>
00012 #include <stdair/bom/Inventory.hpp>
00013
00014 namespace stdair {
00015
00016     // //////////////////////////////////////
00017     BomRoot::BomRoot() {
00018         assert (false);
00019     }
00020
00021     // //////////////////////////////////////
00022     BomRoot::BomRoot (const BomRoot& iBomRoot) {
00023         assert (false);
00024     }
00025
00026     // //////////////////////////////////////
00027     BomRoot::BomRoot (const Key_T& iKey) : _key (iKey) {
00028     }
00029
00030     // //////////////////////////////////////
00031     BomRoot::~BomRoot() {
00032     }
00033
00034     // //////////////////////////////////////
00035     std::string BomRoot::toString() const {
00036         std::ostringstream oStr;
00037         oStr << _key.toString();
00038         return oStr.str();
00039     }
00040
00041     // //////////////////////////////////////
00042     Inventory* BomRoot::getInventory (const std::string& iInventoryKeyStr) const {
00043         Inventory* oInventory_ptr =
00044             BomManager::getObjectPtr<Inventory> (*this, iInventoryKeyStr);
00045         return oInventory_ptr;
00046     }
00047
00048     // //////////////////////////////////////
00049     Inventory* BomRoot::getInventory (const InventoryKey& iInventoryKey) const {
00050         return getInventory (iInventoryKey.toString());
00051     }
00052
00053 }

```

35.187 stdair/bom/BomRoot.hpp File Reference

```

#include <iosfwd>

#include <string>

#include <stdair/bom/BomAbstract.hpp>

#include <stdair/bom/BomRootKey.hpp>

```

Classes

- class `stdair::BomRoot`

Class representing the actual attributes for the Bom root.

Namespaces

- namespace `boost`
Forward declarations.
- namespace `boost::serialization`
- namespace `stdair`

Handle on the StdAir library context.

35.188 BomRoot.hpp

```

00001 #ifndef __STDAIR_BOM_BOMROOT_HPP
00002 #define __STDAIR_BOM_BOMROOT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/bom/BomAbstract.hpp>
00012 #include <stdair/bom/BomRootKey.hpp>
00013
00015 namespace boost {
00016     namespace serialization {
00017         class access;
00018     }
00019 }
00020
00021 namespace stdair {
00022
00024     struct InventoryKey;
00025     class Inventory;
00026
00030     class BomRoot : public BomAbstract {
00031     public:
00032         template <typename BOM> friend class FacBom;
00033         friend class FacBomManager;
00034         friend class boost::serialization::access;
00035
00039         typedef BomRootKey Key_T;
00040
00041     public:
00042         // ////////////////////////////////// Getters //////////////////////////////////
00043         const Key_T& getKey() const {
00044             return _key;
00045         }
00046
00050         const HolderMap_T& getHolderMap() const {
00051             return _holderMap;

```

```

00052     }
00053
00064     Inventory* getInventory (const std::string& iInventoryKeyStr) const;
00065
00076     Inventory* getInventory (const InventoryKey&) const;
00077
00078
00079 public:
00080     // /////////// Display support methods ///////////
00086     void toStream (std::ostream& ioOut) const {
00087         ioOut << toString();
00088     }
00089
00095     void fromStream (std::istream& ioIn) {
00096     }
00097
00101     std::string toString() const;
00102
00106     const std::string describeKey() const {
00107         return _key.toString();
00108     }
00109
00110
00111 public:
00112     // /////////// (Boost) Serialisation support methods ///////////
00123     template<class Archive>
00124     void serialize (Archive& ar, const unsigned int iFileVersion);
00125
00126 private:
00134     void serialisationImplementationExport() const;
00135     void serialisationImplementationImport();
00136
00137
00138 protected:
00139     // /////////// Constructors and destructors ///////////
00143     BomRoot();
00144
00148     BomRoot (const BomRoot&);
00149
00153     BomRoot (const Key_T& iKey);
00154
00158     ~BomRoot();
00159
00160
00161 protected:
00162     // /////////// Attributes ///////////
00166     Key_T _key;
00167
00171     HolderMap_T _holderMap;
00172 };
00173
00174 }
00175 #endif // __STDAIR_BOM_BOMROOT_HPP

```

35.189 stdair/bom/BomRootKey.cpp File Reference

```

#include <cassert>
#include <sstream>

```

```
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/BomRootKey.hpp>
```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

Functions

- template void `stdair::BomRootKey::serialize< ba::text_oarchive >` (ba::text_oarchive &, unsigned int)
- template void `stdair::BomRootKey::serialize< ba::text_iarchive >` (ba::text_iarchive &, unsigned int)

35.190 BomRootKey.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/access.hpp>
00011 // StdAir
00012 #include <stdair/basic/BasConst_General.hpp>
00013 #include <stdair/bom/BomRootKey.hpp>
00014
00015 namespace stdair {
00016
00017 // //////////////////////////////////////
00018 BomRootKey::BomRootKey()
00019     : _id (DEFAULT_BOM_ROOT_KEY) {
00020 }
00021
00022 // //////////////////////////////////////
00023 BomRootKey::BomRootKey (const BomRootKey& iBomRootKey)
00024     : _id (iBomRootKey._id) {
00025 }
00026
00027 // //////////////////////////////////////
00028 BomRootKey::BomRootKey (const std::string& iIdentification)
00029     : _id (iIdentification) {
00030 }
00031
00032 // //////////////////////////////////////
```

```

00033 BomRootKey::~BomRootKey() {
00034 }
00035
00036 // //////////////////////////////////////
00037 void BomRootKey::toStream (std::ostream& ioOut) const {
00038     ioOut << "BomRootKey: " << toString() << std::endl;
00039 }
00040
00041 // //////////////////////////////////////
00042 void BomRootKey::fromStream (std::istream& ioIn) {
00043 }
00044
00045 // //////////////////////////////////////
00046 const std::string BomRootKey::toString() const {
00047     std::ostringstream oStr;
00048     oStr << _id;
00049     return oStr.str();
00050 }
00051
00052 // //////////////////////////////////////
00053 void BomRootKey::serialisationImplementationExport() const {
00054     std::ostringstream oStr;
00055     boost::archive::text_oarchive oa (oStr);
00056     oa << *this;
00057 }
00058
00059 // //////////////////////////////////////
00060 void BomRootKey::serialisationImplementationImport() {
00061     std::istringstream iStr;
00062     boost::archive::text_iarchive ia (iStr);
00063     ia >> *this;
00064 }
00065
00066 // //////////////////////////////////////
00067 template<class Archive>
00068 void BomRootKey::serialize (Archive& ioArchive,
00069                             const unsigned int iFileVersion) {
00070     ioArchive & _id;
00071 }
00072
00073 // //////////////////////////////////////
00074 // Explicit template instantiation
00075 namespace ba = boost::archive;
00076 template void BomRootKey::serialize<ba::text_oarchive> (ba::text_oarchive&,
00077                                                         unsigned int);
00078 template void BomRootKey::serialize<ba::text_iarchive> (ba::text_iarchive&,
00079                                                         unsigned int);
00080 // //////////////////////////////////////
00081
00082 }

```

35.191 stdair/bom/BomRootKey.hpp File Reference

```

#include <iosfwd>

#include <string>

#include <stdair/bom/KeyAbstract.hpp>

```

Classes

- struct `stdair::BomRootKey`
Key of the BOM structure root.

Namespaces

- namespace `boost`
Forward declarations.
- namespace `boost::serialization`
- namespace `stdair`
Handle on the StdAir library context.

35.192 BomRootKey.hpp

```

00001 #ifndef __STDAIR_BOM_BOMROOTKEY_HPP
00002 #define __STDAIR_BOM_BOMROOTKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/bom/KeyAbstract.hpp>
00012
00014 namespace boost {
00015     namespace serialization {
00016         class access;
00017     }
00018 }
00019
00020 namespace stdair {
00021
00025     struct BomRootKey : public KeyAbstract {
00026         friend class boost::serialization::access;
00027
00028         // ////////////////////////////////// Constructors and destructors //////////////////////////////////
00029     public:
00033         BomRootKey ();
00034
00038         BomRootKey (const std::string& iIdentification);
00039
00043         BomRootKey (const BomRootKey&);
00044
00048         ~BomRootKey ();
00049
00050
00051     public:
00052         // ////////////////////////////////// Getters //////////////////////////////////
00056         const std::string& getID() const {
00057             return _id;
00058         }
00059
00060

```

```

00061 public:
00062     // //////////// Display support methods ////////////
00068     void toStream (std::ostream& ioOut) const;
00069
00075     void fromStream (std::istream& ioIn);
00076
00086     const std::string toString() const;
00087
00088
00089 public:
00090     // //////////// (Boost) Serialisation support methods ////////////
00094     template<class Archive>
00095     void serialize (Archive& ar, const unsigned int iFileVersion);
00096
00097 private:
00102     void serialisationImplementationExport() const;
00103     void serialisationImplementationImport();
00104
00105
00106 private:
00107     // ////////////////////////////////// Attributes //////////////////////////////////
00111     std::string _id;
00112 };
00113
00114 }
00115 #endif // __STDAIR_BOM_BOMROOTKEY_HPP

```

35.193 stdair/bom/BookingClass.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/RandomGeneration.hpp>
#include <stdair/bom/BookingClass.hpp>

```

Namespaces

- namespace `stdair`
Handle on the *StdAir* library context.

35.194 BookingClass.cpp

```

00001 // ////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_General.hpp>

```

```

00009 #include <stdair/basic/BasConst_Inventory.hpp>
00010 #include <stdair/basic/RandomGeneration.hpp>
00011 #include <stdair/bom/BookingClass.hpp>
00012
00013 namespace stdair {
00014
00015     // //////////////////////////////////////
00016     BookingClass::BookingClass() : _key (DEFAULT_CLASS_CODE), _parent (NULL) {
00017         assert (false);
00018     }
00019
00020     // //////////////////////////////////////
00021     BookingClass::BookingClass (const BookingClass&)
00022         : _key (DEFAULT_CLASS_CODE), _parent (NULL) {
00023         assert (false);
00024     }
00025
00026     // //////////////////////////////////////
00027     BookingClass::BookingClass (const Key_T& iKey)
00028         : _key (iKey), _parent (NULL), _cumulatedProtection (0.0),
00029           _protection (0.0), _cumulatedBookingLimit (0.0), _au (0.0), _nego (0.0),
00030           _noShowPercentage (0.0), _cancellationPercentage (0.0),
00031           _nbOfBookings (0.0), _groupNbOfBookings (0.0),
00032           _groupPendingNbOfBookings (0.0), _staffNbOfBookings (0.0),
00033           _wlNbOfBookings (0.0), _nbOfCancellations (0.), _etb (0.0),
00034           _netClassAvailability (0.0), _segmentAvailability (0.0),
00035           _netRevenueAvailability (0.0), _yield (0.0), _mean (0.0), _stdDev (0.0) {
00036     }
00037
00038     // //////////////////////////////////////
00039     BookingClass::~BookingClass() {
00040     }
00041
00042     // //////////////////////////////////////
00043     std::string BookingClass::toString() const {
00044         std::ostringstream ostr;
00045         ostr << describeKey();
00046         return ostr.str();
00047     }
00048
00049     // //////////////////////////////////////
00050     void BookingClass::sell (const NbOfBookings_T& iNbOfBookings) {
00051         _nbOfBookings += iNbOfBookings;
00052     }
00053
00054     // //////////////////////////////////////
00055     void BookingClass::cancel (const NbOfBookings_T& iNbOfCancellations) {
00056         _nbOfBookings -= iNbOfCancellations;
00057         _nbOfCancellations += iNbOfCancellations;
00058     }
00059
00060     // //////////////////////////////////////
00061     void BookingClass::generateDemandSamples (const int& K) {
00062         _generatedDemandVector.clear();
00063         if (_stdDev > 0) {
00064             RandomGeneration lGenerator (DEFAULT_RANDOM_SEED);
00065             for (int i = 0; i < K; ++i) {
00066                 RealNumber_T lDemandSample = lGenerator.generateNormal (_mean, _stdDev);
00067                 _generatedDemandVector.push_back (lDemandSample);
00068             }
00069         }
00070     }

```

```

00071
00072 // //////////////////////////////////////
00073 void BookingClass::generateDemandSamples (const int& K,
00074                                           const RandomSeed_T& iSeed) {
00075     _generatedDemandVector.clear();
00076     if (_stdDev > 0) {
00077         RandomGeneration lGenerator (iSeed);
00078         for (int i = 0; i < K; ++i) {
00079             RealNumber_T lDemandSample = lGenerator.generateNormal (_mean, _stdDev);
00080             _generatedDemandVector.push_back (lDemandSample);
00081         }
00082     }
00083 }
00084
00085 }
00086

```

35.195 stdair/bom/BookingClass.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/stdair_rm_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/BookingClassKey.hpp>
#include <stdair/bom/BookingClassTypes.hpp>

```

Classes

- class [stdair::BookingClass](#)

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.196 BookingClass.hpp

```

00001 #ifndef __STDAIR_BOM_BOOKINGCLASS_HPP
00002 #define __STDAIR_BOM_BOOKINGCLASS_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>

```

```
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/stdair_maths_types.hpp>
00013 #include <stdair/stdair_rm_types.hpp>
00014 #include <stdair/bom/BomAbstract.hpp>
00015 #include <stdair/bom/BookingClassKey.hpp>
00016 #include <stdair/bom/BookingClassTypes.hpp>
00017
00018 namespace stdair {
00019
00020     class BookingClass : public BomAbstract {
00021     public:
00022         template <typename BOM> friend class FacBom;
00023         friend class FacBomManager;
00024
00025         // //////////// Type definitions ////////////
00026         typedef BookingClassKey Key_T;
00027
00028     public:
00029         // //////////// Getters ////////////
00030         const Key_T& getKey() const {
00031             return _key;
00032         }
00033
00034         const ClassCode_T& getClassCode() const {
00035             return _key.getClassCode();
00036         }
00037
00038         BomAbstract* const getParent() const {
00039             return _parent;
00040         }
00041
00042         const HolderMap_T& getHolderMap() const {
00043             return _holderMap;
00044         }
00045
00046         const SubclassCode_T& getSubclassCode() const {
00047             return _subclassCode;
00048         }
00049
00050         const AuthorizationLevel_T& getAuthorizationLevel() const {
00051             return _au;
00052         }
00053
00054         const ProtectionLevel_T& getProtection() const {
00055             return _protection;
00056         }
00057
00058         const ProtectionLevel_T& getCumulatedProtection() const {
00059             return _cumulatedProtection;
00060         }
00061
00062         const BookingLimit_T& getCumulatedBookingLimit() const {
00063             return _cumulatedBookingLimit;
00064         }
00065
00066         const NbOfSeats_T& getNegotiatedSpace() const {
00067             return _nego;
00068         }
00069
00070         const OverbookingRate_T& getNoShowPercentage() const {
00071             return _noShowPercentage;
00072         }
00073
00074     private:
00075         Key_T _key;
00076         BomAbstract* _parent;
00077         HolderMap_T _holderMap;
00078         SubclassCode_T _subclassCode;
00079         AuthorizationLevel_T _au;
00080         ProtectionLevel_T _protection;
00081         ProtectionLevel_T _cumulatedProtection;
00082         BookingLimit_T _cumulatedBookingLimit;
00083         NbOfSeats_T _nego;
00084         OverbookingRate_T _noShowPercentage;
00085     };
00086 }
```

```

00088     }
00089
00091     const OverbookingRate_T& getCancellationPercentage() const {
00092         return _cancellationPercentage;
00093     }
00094
00096     const NbOfBookings_T& getNbOfBookings() const {
00097         return _nbOfBookings;
00098     }
00099
00101     const NbOfBookings_T& getNbOfGroupBookings() const {
00102         return _groupNbOfBookings;
00103     }
00104
00106     const NbOfBookings_T& getNbOfPendingGroupBookings() const {
00107         return _groupPendingNbOfBookings;
00108     }
00109
00111     const NbOfBookings_T& getNbOfStaffBookings() const {
00112         return _staffNbOfBookings;
00113     }
00114
00116     const NbOfBookings_T& getNbOfWLBookings() const {
00117         return _wlNbOfBookings;
00118     }
00119
00121     const NbOfCancellations_T& getNbOfCancellations() const {
00122         return _nbOfCancellations;
00123     }
00124
00126     const NbOfBookings_T& getETB() const {
00127         return _etb;
00128     }
00129
00131     const Availability_T& getNetClassAvailability() const {
00132         return _netClassAvailability;
00133     }
00134
00136     const Availability_T& getSegmentAvailability() const {
00137         return _segmentAvailability;
00138     }
00139
00141     const Availability_T& getNetRevenueAvailability() const {
00142         return _netRevenueAvailability;
00143     }
00144
00146     const Yield_T& getYield () const { return _yield; }
00147
00149     const MeanValue_T& getMean () const { return _mean; }
00150     const StdDevValue_T& getStdDev () const {return _stdDev; }
00151
00153     const GeneratedDemandVector_T& getGeneratedDemandVector () const {
00154         return _generatedDemandVector;
00155     }
00156
00157 public:
00158     // /////////// Setters ///////////
00160     void setCumulatedProtection (const ProtectionLevel_T& iPL) {
00161         _cumulatedProtection = iPL;
00162     }
00163
00165     void setProtection (const ProtectionLevel_T& iPL) {

```

```

00166         _protection = iPL;
00167     }
00168
00170     void setCumulatedBookingLimit (const BookingLimit_T& iBL) {
00171         _cumulatedBookingLimit = iBL;
00172     }
00173
00175     void setAuthorizationLevel (const AuthorizationLevel_T& iAU) {
00176         _au = iAU;
00177     }
00178
00180     void setSegmentAvailability (const Availability_T& iAvl) {
00181         _segmentAvailability = iAvl;
00182     }
00183
00185     void setYield (const Yield_T& iYield) { _yield = iYield; }
00186
00188     void setMean (const MeanValue_T& iMean) { _mean = iMean; }
00189     void setStdDev (const StdDevValue_T& iStdDev) { _stdDev = iStdDev; }
00190
00191 public:
00192     // //////////// Display support methods ////////////
00195     void toStream (std::ostream& ioOut) const {
00196         ioOut << toString();
00197     }
00198
00201     void fromStream (std::istream& ioIn) {
00202     }
00203
00205     std::string toString() const;
00206
00208     const std::string describeKey() const {
00209         return _key.toString();
00210     }
00211
00212 public:
00213     // //////////// Business Methods ////////////
00215     void sell (const NbOfBookings_T&);
00216
00218     void cancel (const NbOfBookings_T&);
00219
00222     void generateDemandSamples (const int&);
00223
00226     void generateDemandSamples (const int&, const RandomSeed_T&);
00227
00228 protected:
00229     // //////////// Constructors and destructors ////////////
00231     BookingClass (const Key_T&);
00233     virtual ~BookingClass();
00234
00235 private:
00237     BookingClass();
00239     BookingClass (const BookingClass&);
00240
00241
00242 protected:
00243     // //////////// Attributes ////////////
00245     Key_T _key;
00246
00248     BomAbstract* _parent;
00249
00251     HolderMap_T _holderMap;

```

```

00252
00254     SubclassCode_T _subclassCode;
00255
00257     ProtectionLevel_T _cumulatedProtection;
00258
00260     ProtectionLevel_T _protection;
00261
00263     BookingLimit_T _cumulatedBookingLimit;
00264
00266     AuthorizationLevel_T _au;
00267
00269     NbOfSeats_T _nego;
00270
00272     OverbookingRate_T _noShowPercentage;
00273
00275     OverbookingRate_T _cancellationPercentage;
00276
00278     NbOfBookings_T _nbOfBookings;
00279
00281     NbOfBookings_T _groupNbOfBookings;
00282
00284     NbOfBookings_T _groupPendingNbOfBookings;
00285
00287     NbOfBookings_T _staffNbOfBookings;
00288
00290     NbOfBookings_T _wlNbOfBookings;
00291
00293     NbOfCancellations_T _nbOfCancellations;
00294
00296     NbOfBookings_T _etb;
00297
00299     Availability_T _netClassAvailability;
00300
00302     Availability_T _segmentAvailability;
00303
00305     Availability_T _netRevenueAvailability;
00306
00308     Yield_T _yield;
00309
00311     MeanValue_T _mean;
00312     StdDevValue_T _stdDev;
00313
00315     GeneratedDemandVector_T _generatedDemandVector;
00316 };
00317
00318 }
00319 #endif // __STDAIR_BOM_BOOKINGCLASS_HPP

```

35.197 stdair/bom/BookingClassKey.cpp File Reference

```

#include <cassert>

#include <sstream>

#include <stdair/basic/BasConst_Inventory.hpp>

#include <stdair/bom/BookingClassKey.hpp>

```

Namespaces

- namespace `stdair`

Handle on the StdAir library context.

35.198 BookingClassKey.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_Inventory.hpp>
00009 #include <stdair/bom/BookingClassKey.hpp>
00010
00011 namespace stdair {
00012
00013 // //////////////////////////////////////
00014 BookingClassKey::BookingClassKey() : _classCode (DEFAULT_CLASS_CODE) {
00015     assert (false);
00016 }
00017
00018 // //////////////////////////////////////
00019 BookingClassKey::BookingClassKey (const BookingClassKey& iKey)
00020     : _classCode (iKey._classCode) {
00021 }
00022
00023 // //////////////////////////////////////
00024 BookingClassKey::BookingClassKey (const ClassCode_T& iClassCode)
00025     : _classCode (iClassCode) {
00026 }
00027
00028 // //////////////////////////////////////
00029 BookingClassKey::~BookingClassKey () {
00030 }
00031
00032 // //////////////////////////////////////
00033 void BookingClassKey::toStream (std::ostream& ioOut) const {
00034     ioOut << "BookingClassKey: " << toString();
00035 }
00036
00037 // //////////////////////////////////////
00038 void BookingClassKey::fromStream (std::istream& ioIn) {
00039 }
00040
00041 // //////////////////////////////////////
00042 const std::string BookingClassKey::toString() const {
00043     std::ostringstream oStr;
00044     oStr << _classCode;
00045     return oStr.str();
00046 }
00047
00048 }

```

35.199 stdair/bom/BookingClassKey.hpp File Reference

```
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

Classes

- struct [stdair::BookingClassKey](#)

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.200 BookingClassKey.hpp

```
00001 #ifndef __STDAIR_BOM_BOOKINGCLASSKEY_HPP
00002 #define __STDAIR_BOM_BOOKINGCLASSKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009 #include <stdair/bom/KeyAbstract.hpp>
00010
00011 namespace stdair {
00012
00016     struct BookingClassKey : public KeyAbstract {
00017
00018         // ////////// Constructors and destructors //////////
00019     private:
00021         BookingClassKey();
00022
00023     public:
00025         BookingClassKey (const ClassCode_T& iClassCode);
00027         BookingClassKey (const BookingClassKey&);
00029         ~BookingClassKey();
00030
00031
00032         // ////////// Getters //////////
00034         const ClassCode_T& getClassCode () const {
00035             return _classCode;
00036         }
00037
00038
00039         // ////////// Display support methods //////////
00042         void toStream (std::ostream& ioOut) const;
00043
00046         void fromStream (std::istream& ioIn);
00047
00053         const std::string toString() const;
00054
00055     private:
00056
```

```

00057 // ////////////////////////////////// Attributes //////////////////////////////////
00059     ClassCode_T _classCode;
00060 };
00061
00062 }
00063 #endif // __STDAIR_BOM_BOOKINGCLASKEY_HPP

```

35.201 stdair/bom/BookingClassTypes.hpp File Reference

```

#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>

```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

Typedefs

- typedef `std::list< BookingClass * >` `stdair::BookingClassList_T`
- typedef `std::map< const MapKey_T, BookingClass * >` `stdair::BookingClassMap_T`

35.202 BookingClassTypes.hpp

```

00001 // ////////////////////////////////// //////////////////////////////////
00002 #ifndef __STDAIR_BOM_BOOKINGCLASSTYPES_HPP
00003 #define __STDAIR_BOM_BOOKINGCLASSTYPES_HPP
00004
00005 // ////////////////////////////////// //////////////////////////////////
00006 // Import section
00007 // ////////////////////////////////// //////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // StdAir
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016     // Forward declarations.
00017     class BookingClass;
00018
00020     typedef std::list<BookingClass*> BookingClassList_T;
00021
00023     typedef std::map<const MapKey_T, BookingClass*> BookingClassMap_T;
00024 }
00025 #endif // __STDAIR_BOM_BOOKINGCLASSTYPES_HPP
00026

```

35.203 stdair/bom/BookingRequestStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/date_time/gregorian/formatters.hpp>
#include <boost/date_time/posix_time/posix_time.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_Request.hpp>
#include <stdair/bom/BookingRequestStruct.hpp>
```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

Functions

- void [stdair::intDisplay](#) (std::ostream &oStream, const int &iInt)

35.204 BookingRequestStruct.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost
00008 #include <boost/date_time/gregorian/formatters.hpp>
00009 #include <boost/date_time/posix_time/posix_time.hpp>
00010 // StdAir
00011 #include <stdair/basic/BasConst_Inventory.hpp>
00012 #include <stdair/basic/BasConst_Request.hpp>
00013 #include <stdair/bom/BookingRequestStruct.hpp>
00014
00015 namespace stdair {
00016
00017 // //////////////////////////////////////
00018 BookingRequestStruct::BookingRequestStruct()
00019 : _origin (DEFAULT_ORIGIN), _destination (DEFAULT_DESTINATION),
00020   _pos (DEFAULT_POS),
00021   _preferredDepartureDate (DEFAULT_PREFERRED_DEPARTURE_DATE),
00022   _preferredDepartureTime (DEFAULT_PREFERRED_DEPARTURE_TIME),
00023   _requestDateTime (DEFAULT_REQUEST_DATE_TIME),
00024   _preferredCabin (DEFAULT_PREFERRED_CABIN),
00025   _partySize (DEFAULT_PARTY_SIZE),
00026   _channel (DEFAULT_CHANNEL),
00027   _tripType (TRIP_TYPE_ONE_WAY),
00028   _stayDuration (DEFAULT_STAY_DURATION),
00029   _frequentFlyerType (DEFAULT_FF_TIER),
```

```

00030     _wtp (DEFAULT_WTP),
00031     _valueOfTime (DEFAULT_VALUE_OF_TIME) {
00032     assert (false);
00033 }
00034
00035 // //////////////////////////////////////
00036 BookingRequestStruct::
00037 BookingRequestStruct (const BookingRequestStruct& iBookingRequest)
00038 : _generatorKey (iBookingRequest._generatorKey),
00039   _origin (iBookingRequest._origin),
00040   _destination (iBookingRequest._destination),
00041   _pos (iBookingRequest._pos),
00042   _preferredDepartureDate (iBookingRequest._preferredDepartureDate),
00043   _preferredDepartureTime (iBookingRequest._preferredDepartureTime),
00044   _requestDateTime (iBookingRequest._requestDateTime),
00045   _preferredCabin (iBookingRequest._preferredCabin),
00046   _partySize (iBookingRequest._partySize),
00047   _channel (iBookingRequest._channel),
00048   _tripType (iBookingRequest._tripType),
00049   _stayDuration (iBookingRequest._stayDuration),
00050   _frequentFlyerType (iBookingRequest._frequentFlyerType),
00051   _wtp (iBookingRequest._wtp),
00052   _valueOfTime (iBookingRequest._valueOfTime) {
00053 }
00054
00055 // //////////////////////////////////////
00056 BookingRequestStruct::
00057 BookingRequestStruct (const DemandGeneratorKey_T& iGeneratorKey,
00058                     const AirportCode_T& iOrigin,
00059                     const AirportCode_T& iDestination,
00060                     const CityCode_T& iPOS,
00061                     const Date_T& iDepartureDate,
00062                     const DateTime_T& iRequestDateTime,
00063                     const CabinCode_T& iPreferredCabin,
00064                     const NbOfSeats_T& iPartySize,
00065                     const ChannelLabel_T& iChannel,
00066                     const TripType_T& iTripType,
00067                     const DayDuration_T& iStayDuration,
00068                     const FrequentFlyer_T& iFrequentFlyerType,
00069                     const Duration_T& iPreferredDepartureTime,
00070                     const WTP_T& iWTP,
00071                     const PriceValue_T& iValueOfTime)
00072 : _generatorKey (iGeneratorKey), _origin (iOrigin),
00073   _destination (iDestination), _pos (iPOS),
00074   _preferredDepartureDate (iDepartureDate),
00075   _preferredDepartureTime (iPreferredDepartureTime),
00076   _requestDateTime (iRequestDateTime),
00077   _preferredCabin (iPreferredCabin), _partySize (iPartySize),
00078   _channel (iChannel), _tripType (iTripType),
00079   _stayDuration (iStayDuration), _frequentFlyerType (iFrequentFlyerType),
00080   _wtp (iWTP), _valueOfTime (iValueOfTime) {
00081 }
00082
00083 // //////////////////////////////////////
00084 BookingRequestStruct::
00085 BookingRequestStruct (const AirportCode_T& iOrigin,
00086                     const AirportCode_T& iDestination,
00087                     const CityCode_T& iPOS,
00088                     const Date_T& iDepartureDate,
00089                     const DateTime_T& iRequestDateTime,
00090                     const CabinCode_T& iPreferredCabin,
00091                     const NbOfSeats_T& iPartySize,

```

```

00092         const ChannelLabel_T& iChannel,
00093         const TripType_T& iTripType,
00094         const DayDuration_T& iStayDuration,
00095         const FrequentFlyer_T& iFrequentFlyerType,
00096         const Duration_T& iPreferredDepartureTime,
00097         const WTP_T& iWTP,
00098         const PriceValue_T& iValueOfTime)
00099     : _generatorKey (""), _origin (iOrigin),
00100       _destination (iDestination), _pos (iPOS),
00101       _preferredDepartureDate (iDepartureDate),
00102       _preferredDepartureTime (iPreferredDepartureTime),
00103       _requestDateTime (iRequestDateTime),
00104       _preferredCabin (iPreferredCabin), _partySize (iPartySize),
00105       _channel (iChannel), _tripType (iTripType),
00106       _stayDuration (iStayDuration), _frequentFlyerType (iFrequentFlyerType),
00107       _wtp (iWTP), _valueOfTime (iValueOfTime) {
00108 }
00109
00110 ///////////////////////////////////////////////////////////////////
00111 BookingRequestStruct::~BookingRequestStruct() {
00112 }
00113
00114 ///////////////////////////////////////////////////////////////////
00115 void BookingRequestStruct::toStream (std::ostream& ioOut) const {
00116     ioOut << describe();
00117 }
00118
00119 ///////////////////////////////////////////////////////////////////
00120 void BookingRequestStruct::fromStream (std::istream& ioIn) {
00121 }
00122
00123 ///////////////////////////////////////////////////////////////////
00124 const std::string BookingRequestStruct::describe() const {
00125     std::ostringstream ostr;
00126     ostr << "At " << _requestDateTime
00127           << ", for (" << _pos << ", " << _channel << ")"
00128           << " " << _origin << "-" << _destination << " (" << _tripType << ")"
00129           << " " << _preferredDepartureDate << " (" << _stayDuration << " days)"
00130           << " " << _preferredDepartureTime
00131           << " " << _preferredCabin << " " << _partySize
00132           << " " << _frequentFlyerType << " " << _wtp << " " << _valueOfTime;
00133     return ostr.str();
00134 }
00135
00136 ///////////////////////////////////////////////////////////////////
00137 void intDisplay (std::ostream& ostream, const int& iInt) {
00138     const int dInt = iInt - static_cast<int> (iInt / 100) * 100;
00139     if (dInt < 10) {
00140         ostream << "0" << dInt;
00141     } else {
00142         ostream << dInt;
00143     }
00144 }
00145
00146 ///////////////////////////////////////////////////////////////////
00147 const std::string BookingRequestStruct::display() const {
00148     std::ostringstream ostr;
00149
00150     // Request date and time
00151     const Date_T& lRequestDate = _requestDateTime.date();
00152     ostr << boost::gregorian::to_iso_extended_string (lRequestDate);
00153

```

```
00154     const Duration_T& lRequestTime = _requestDateTime.time_of_day();
00155     oStr << ", " << boost::posix_time::to_simple_string (lRequestTime);
00156
00157     // POS
00158     oStr << ", " << _pos;
00159
00160     // Channel
00161     oStr << ", " << _channel;
00162
00163     // Origin
00164     oStr << ", " << _origin;
00165
00166     // Destination
00167     oStr << ", " << _destination;
00168
00169     // Preferred departure date
00170     oStr << ", "
00171         << boost::gregorian::to_iso_extended_string (_preferredDepartureDate);
00172
00173     // Preferred departure time
00174     oStr << ", "
00175         << boost::posix_time::to_simple_string (_preferredDepartureTime);
00176
00177     // MIN & MAX preferred departure time (hardcode)
00178     oStr << ", " << "00:00-23:59";
00179
00180     // Preferred arrival date (hardcode to the preferred departure date)
00181     oStr << ", "
00182         << boost::gregorian::to_iso_extended_string (_preferredDepartureDate);
00183
00184     // Preferred arrival time (hard-coded to 23:55)
00185     oStr << ", " << "23:55";
00186
00187     // Preferred cabin
00188     oStr << ", " << _preferredCabin;
00189
00190     // Trip type
00191     oStr << ", " << _tripType;
00192
00193     // Duration of stay
00194     oStr << ", ";
00195     if (_tripType == TRIP_TYPE_ONE_WAY) {
00196         oStr << "0";
00197     } else {
00198         oStr << _stayDuration;
00199     }
00200
00201     // Frequent flyer tier
00202     oStr << ", " << _frequentFlyerType;
00203
00204     // Willingness-to-pay
00205     oStr << ", " << _wtp;
00206
00207     // Disutility per stop (hardcode to 100, expressed as a monetary
00208     // unit per hour)
00209     oStr << ", " << "100";
00210
00211     // Value of time
00212     oStr << ", " << _valueOfTime;
00213
00214     return oStr.str();
00215 }
```

```
00216
00217 }
```

35.205 stdair/bom/BookingRequestStruct.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/BookingRequestTypes.hpp>
```

Classes

- struct [stdair::BookingRequestStruct](#)
Structure holding the elements of a booking request.

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.206 BookingRequestStruct.hpp

```
00001 #ifndef __STDAIR_BOM_BOOKINGREQUESTSTRUCT_HPP
00002 #define __STDAIR_BOM_BOOKINGREQUESTSTRUCT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/stdair_demand_types.hpp>
00013 #include <stdair/basic/StructAbstract.hpp>
00014 #include <stdair/bom/BookingRequestTypes.hpp>
00015
00016 namespace stdair {
00017
00021     struct BookingRequestStruct : public StructAbstract {
00022     public:
00023         // ////////////////////////////////////// Getters //////////////////////////////////////
00025         const DemandGeneratorKey_T& getDemandGeneratorKey () const {
00026             return _generatorKey;
00027         }
00028
00030         const AirportCode_T& getOrigin() const {
```

```
00031         return _origin;
00032     }
00033
00035     const AirportCode_T& getDestination() const {
00036         return _destination;
00037     }
00038
00040     const CityCode_T& getPOS() const {
00041         return _pos;
00042     }
00043
00045     const Date_T& getPreferedDepartureDate() const {
00046         return _preferredDepartureDate;
00047     }
00048
00050     const Duration_T& getPreferredDepartureTime() const {
00051         return _preferredDepartureTime;
00052     }
00053
00055     const DateTime_T& getRequestDateTime() const {
00056         return _requestDateTime;
00057     }
00058
00060     const CabinCode_T& getPreferredCabin() const {
00061         return _preferredCabin;
00062     }
00063
00065     const NbOfSeats_T& getPartySize() const {
00066         return _partySize;
00067     }
00068
00070     const ChannelLabel_T& getBookingChannel() const {
00071         return _channel;
00072     }
00073
00075     const TripType_T& getTripType() const {
00076         return _tripType;
00077     }
00078
00080     const DayDuration_T& getStayDuration() const {
00081         return _stayDuration;
00082     }
00083
00085     const FrequentFlyer_T& getFrequentFlyerType() const {
00086         return _frequentFlyerType;
00087     }
00088
00090     const WTP_T& getWTP() const {
00091         return _wtp;
00092     }
00093
00095     const PriceValue_T& getValueOfTime () const {
00096         return _valueOfTime;
00097     }
00098
00099
00100 public:
00101     // //////////// Display support method ////////////
00106     void toStream (std::ostream& ioOut) const;
00107
00112     void fromStream (std::istream& ioIn);
00113
```

```

00117     const std::string describe() const;
00118
00169     const std::string display() const;
00170
00171
00172 public:
00173     // ////////////////////////////////// Constructors and Destructors //////////////////////////////////
00177     BookingRequestStruct (const DemandGeneratorKey_T& iGeneratorKey,
00178                          const AirportCode_T& iOrigin,
00179                          const AirportCode_T& iDestination,
00180                          const CityCode_T& iPOS,
00181                          const Date_T& iDepartureDate,
00182                          const DateTime_T& iRequestDateTime,
00183                          const CabinCode_T& iPreferredCabin,
00184                          const NbOfSeats_T& iPartySize,
00185                          const ChannelLabel_T& iChannel,
00186                          const TripType_T& iTripType,
00187                          const DayDuration_T& iStayDuration,
00188                          const FrequentFlyer_T& iFrequentFlyerType,
00189                          const Duration_T& iPreferredDepartureTime,
00190                          const WTP_T& iWTP,
00191                          const PriceValue_T& iValueOfTime);
00192
00196     BookingRequestStruct (const AirportCode_T& iOrigin,
00197                          const AirportCode_T& iDestination,
00198                          const CityCode_T& iPOS,
00199                          const Date_T& iDepartureDate,
00200                          const DateTime_T& iRequestDateTime,
00201                          const CabinCode_T& iPreferredCabin,
00202                          const NbOfSeats_T& iPartySize,
00203                          const ChannelLabel_T& iChannel,
00204                          const TripType_T& iTripType,
00205                          const DayDuration_T& iStayDuration,
00206                          const FrequentFlyer_T& iFrequentFlyerType,
00207                          const Duration_T& iPreferredDepartureTime,
00208                          const WTP_T& iWTP,
00209                          const PriceValue_T& iValueOfTime);
00213     BookingRequestStruct (const BookingRequestStruct&);
00214
00218     ~BookingRequestStruct ();
00219
00220
00221 private:
00227     BookingRequestStruct ();
00228
00229
00230 private:
00231     // ////////////////////////////////// Attributes //////////////////////////////////
00233     const DemandGeneratorKey_T _generatorKey;
00234
00236     const AirportCode_T _origin;
00237
00239     const AirportCode_T _destination;
00240
00242     const CityCode_T _pos;
00243
00245     const Date_T _preferredDepartureDate;
00246
00248     const Duration_T _preferredDepartureTime;
00249
00251     const DateTime_T _requestDateTime;
00252

```

```

00254     const CabinCode_T _preferredCabin;
00255
00257     const NbOfSeats_T _partySize;
00258
00260     const ChannelLabel_T _channel;
00261
00264     const TripType_T _tripType;
00265
00267     const DayDuration_T _stayDuration;
00268
00270     const FrequentFlyer_T _frequentFlyerType;
00271
00273     const WTP_T _wtp;
00274
00276     const PriceValue_T _valueOfTime;
00277 };
00278
00279 }
00280 #endif // __STDAIR_BOM_BOOKINGREQUESTSTRUCT_HPP

```

35.207 stdair/bom/BookingRequestTypes.hpp File Reference

```
#include <boost/shared_ptr.hpp>
```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

Typedefs

- typedef boost::shared_ptr< BookingRequestStruct > [stdair::BookingRequestPtr_T](#)
- typedef std::string [stdair::DemandGeneratorKey_T](#)

35.208 BookingRequestTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_BOOKINGREQUESTTYPES_HPP
00003 #define __STDAIR_BOM_BOOKINGREQUESTTYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // Boost
00009 #include <boost/shared_ptr.hpp>
00010
00011 namespace stdair {
00012
00013     // Forward declarations
00014     struct BookingRequestStruct;
00015
00016     // ////////////////////////////////// Type definitions //////////////////////////////////

```

```

00018     typedef boost::shared_ptr<BookingRequestStruct> BookingRequestPtr_T;
00019
00021     typedef std::string DemandGeneratorKey_T;
00022
00023
00024 }
00025 #endif // __STDAIR_BOM_BOOKINGREQUESTTYPES_HPP
00026

```

35.209 stdair/bom/Bucket.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/Bucket.hpp>

```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.210 Bucket.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/access.hpp>
00011 // StdAir
00012 #include <stdair/basic/BasConst_Inventory.hpp>
00013 #include <stdair/bom/Bucket.hpp>
00014
00015 namespace stdair {
00016
00017 // //////////////////////////////////////
00018     Bucket::Bucket ()
00019         : _key (DEFAULT_SEAT_INDEX), _parent (NULL) {
00020         assert (false);
00021     }
00022
00023 // //////////////////////////////////////
00024     Bucket::Bucket (const Key_T& iKey) : _key (iKey), _parent (NULL) {
00025     }

```

```

00026
00027 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00028 Bucket::~Bucket() {
00029 }
00030
00031 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00032 std::string Bucket::toString() const {
00033     std::ostringstream ostr;
00034     ostr << describeKey();
00035     return ostr.str();
00036 }
00037
00038 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00039 void Bucket::serialisationImplementationExport() const {
00040     std::ostringstream ostr;
00041     boost::archive::text_oarchive oa (ostr);
00042     oa << *this;
00043 }
00044
00045 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00046 void Bucket::serialisationImplementationImport() {
00047     std::istringstream istr;
00048     boost::archive::text_iarchive ia (istr);
00049     ia >> *this;
00050 }
00051
00052 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00053 template<class Archive>
00054 void Bucket::serialize (Archive& ioArchive, const unsigned int iFileVersion) {
00055     ioArchive & _key;
00056 }
00057
00058 }
00059

```

35.211 stdair/bom/Bucket.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/BucketKey.hpp>
#include <stdair/bom/BucketTypes.hpp>

```

Classes

- class [stdair::Bucket](#)
Class representing the actual attributes for an airline booking class.

Namespaces

- namespace [boost](#)

Forward declarations.

- namespace `boost::serialization`
- namespace `stdair`

Handle on the StdAir library context.

35.212 Bucket.hpp

```

00001 #ifndef __STDAIR_BOM_BUCKET_HPP
00002 #define __STDAIR_BOM_BUCKET_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/bom/BomAbstract.hpp>
00013 #include <stdair/bom/BucketKey.hpp>
00014 #include <stdair/bom/BucketTypes.hpp>
00015
00017 namespace boost {
00018     namespace serialization {
00019         class access;
00020     }
00021 }
00022
00023 namespace stdair {
00024
00029     class Bucket : public BomAbstract {
00030     public:
00031         template <typename BOM> friend class FacBom;
00032         friend class FacBomManager;
00033         friend class boost::serialization::access;
00034
00035         // ////////////////////////////////// Type definitions //////////////////////////////////
00039         typedef BucketKey Key_T;
00040
00041     public:
00042         // ////////////////////////////////// Getters //////////////////////////////////
00046         const Key_T& getKey() const {
00047             return _key;
00048         }
00049
00053         BomAbstract* const getParent() const {
00054             return _parent;
00055         }
00056
00058         const HolderMap_T& getHolderMap() const {
00059             return _holderMap;
00060         }
00061
00063         const SeatIndex_T& getSeatIndex() const {
00064             return _key.getSeatIndex();
00065         }
00066
00068         const Yield_T& getYieldRangeUpperValue() const {
00069             return _yieldRangeUpperValue;

```

```

00070     }
00071
00073     const CabinCapacity_T& getAvailability() const {
00074         return _availability;
00075     }
00076
00078     const NbOfSeats_T& getSoldSeats() const {
00079         return _soldSeats;
00080     }
00081
00082
00083     // //////////// Setters ////////////
00085     void setYieldRangeUpperValue (const Yield_T& iYield) {
00086         _yieldRangeUpperValue = iYield;
00087     }
00088
00090     void setAvailability (const CabinCapacity_T& iAvl) {
00091         _availability = iAvl;
00092     }
00093
00095     void setSoldSeats (const NbOfSeats_T& iSoldSeats) {
00096         _soldSeats = iSoldSeats;
00097     }
00098
00099
00100 public:
00101     // //////////// Display support methods ////////////
00107     void toStream (std::ostream& ioOut) const {
00108         ioOut << toString();
00109     }
00110
00116     void fromStream (std::istream& ioIn) {
00117     }
00118
00122     std::string toString() const;
00123
00127     const std::string describeKey() const {
00128         return _key.toString();
00129     }
00130
00131
00132 public:
00133     // //////////// (Boost) Serialisation support methods ////////////
00137     template<class Archive>
00138     void serialize (Archive& ar, const unsigned int iFileVersion);
00139
00140 private:
00145     void serialisationImplementationExport() const;
00146     void serialisationImplementationImport();
00147
00148
00149 protected:
00150     // //////////// Constructors and destructors ////////////
00154     Bucket (const Key_T&);
00155
00159     virtual ~Bucket();
00160
00161 private:
00165     Bucket();
00166
00170     Bucket (const Bucket&);
00171

```

```

00172
00173     protected:
00174         // ////////////////////////////////// Children //////////////////////////////////
00178         Key_T _key;
00179
00183         BomAbstract* _parent;
00184
00188         HolderMap_T _holderMap;
00189
00190
00191     protected:
00192         // ////////////////////////////////// Attributes //////////////////////////////////
00196         Yield_T _yieldRangeUpperValue;
00197
00201         CabinCapacity_T _availability;
00202
00206         NbOfSeats_T _soldSeats;
00207     };
00208
00209 }
00210 #endif // __STDAIR_BOM_BUCKET_HPP
00211

```

35.213 stdair/bom/BucketKey.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/bom/BucketKey.hpp>

```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

Functions

- template void `stdair::BucketKey::serialize< ba::text_oarchive >` (ba::text_oarchive &, unsigned int)
- template void `stdair::BucketKey::serialize< ba::text_iarchive >` (ba::text_iarchive &, unsigned int)

35.214 BucketKey.cpp

```

00001 // ////////////////////////////////// Import section //////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////// Import section //////////////////////////////////

```

```

00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/access.hpp>
00011 // StdAir
00012 #include <stdair/bom/BucketKey.hpp>
00013
00014 namespace stdair {
00015
00016 // //////////////////////////////////////
00017 BucketKey::BucketKey() {
00018     assert (false);
00019 }
00020
00021 // //////////////////////////////////////
00022 BucketKey::BucketKey (const SeatIndex_T& iSeatIndex)
00023 : _seatIndex (iSeatIndex) {
00024 }
00025
00026 // //////////////////////////////////////
00027 BucketKey::BucketKey (const BucketKey& iBucketKey)
00028 : _seatIndex (iBucketKey._seatIndex) {
00029 }
00030
00031 // //////////////////////////////////////
00032 BucketKey::~BucketKey() {
00033 }
00034
00035 // //////////////////////////////////////
00036 void BucketKey::toStream (std::ostream& ioOut) const {
00037     ioOut << "BucketKey: " << toString() << std::endl;
00038 }
00039
00040 // //////////////////////////////////////
00041 void BucketKey::fromStream (std::istream& ioIn) {
00042 }
00043
00044 // //////////////////////////////////////
00045 const std::string BucketKey::toString() const {
00046     std::ostringstream oStr;
00047     oStr << _seatIndex;
00048     return oStr.str();
00049 }
00050
00051 // //////////////////////////////////////
00052 void BucketKey::serialisationImplementationExport() const {
00053     std::ostringstream oStr;
00054     boost::archive::text_oarchive oa (oStr);
00055     oa << *this;
00056 }
00057
00058 // //////////////////////////////////////
00059 void BucketKey::serialisationImplementationImport() {
00060     std::istringstream iStr;
00061     boost::archive::text_iarchive ia (iStr);
00062     ia >> *this;
00063 }
00064
00065 // //////////////////////////////////////

```

```

00066     template<class Archive>
00067     void BucketKey::serialize (Archive& ioArchive,
00068                               const unsigned int iFileVersion) {
00069         ioArchive & _seatIndex;
00070     }
00071
00072     // //////////////////////////////////////
00073     // Explicit template instantiation
00074     namespace ba = boost::archive;
00075     template void BucketKey::serialize<ba::text_oarchive> (ba::text_oarchive&,
00076                                                           unsigned int);
00077     template void BucketKey::serialize<ba::text_iarchive> (ba::text_iarchive&,
00078                                                           unsigned int);
00079     // //////////////////////////////////////
00080
00081 }

```

35.215 stdair/bom/BucketKey.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>

```

Classes

- struct [stdair::BucketKey](#)
Key of booking-class.

Namespaces

- namespace [boost](#)
Forward declarations.
- namespace [boost::serialization](#)
- namespace [stdair](#)
Handle on the StdAir library context.

35.216 BucketKey.hpp

```

00001 #ifndef __STDAIR_BOM_BUCKETKEY_HPP
00002 #define __STDAIR_BOM_BUCKETKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>

```

```

00012 #include <stdair/bom/KeyAbstract.hpp>
00013
00015 namespace boost {
00016     namespace serialization {
00017         class access;
00018     }
00019 }
00020
00021 namespace stdair {
00022
00026     struct BucketKey : public KeyAbstract {
00027         friend class boost::serialization::access;
00028
00029         // ////////// Constructors and destructors //////////
00030     private:
00034         BucketKey();
00035
00036     public:
00040         BucketKey (const SeatIndex_T&);
00044         BucketKey (const BucketKey&);
00048         ~BucketKey();
00049
00050
00051     public:
00052         // ////////// Getters //////////
00054         const SeatIndex_T& getSeatIndex() const {
00055             return _seatIndex;
00056         }
00057
00058
00059     public:
00060         // ////////// Display support methods //////////
00066         void toStream (std::ostream& ioOut) const;
00067
00073         void fromStream (std::istream& ioIn);
00074
00084         const std::string toString() const;
00085
00086
00087     public:
00088         // ////////// (Boost) Serialisation support methods //////////
00092         template<class Archive>
00093         void serialize (Archive& ar, const unsigned int iFileVersion);
00094
00095     private:
00100         void serialisationImplementationExport() const;
00101         void serialisationImplementationImport();
00102
00103
00104     private:
00105         // ////////// Attributes //////////
00109         SeatIndex_T _seatIndex;
00110     };
00111
00112 }
00113 #endif // __STDAIR_BOM_BUCKETKEY_HPP

```

35.217 stdair/bom/BucketTypes.hpp File Reference

```
#include <map>
```

```
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

Typedefs

- typedef `std::list< Bucket * >` `stdair::BucketList_T`
- typedef `std::map< const MapKey_T, Bucket * >` `stdair::BucketMap_T`

35.218 BucketTypes.hpp

```
00001 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00002 #ifndef __STDAIR_BOM_BUCKETTYPES_HPP
00003 #define __STDAIR_BOM_BUCKETTYPES_HPP
00004
00005 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00006 // Import section
00007 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // StdAir
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016     // Forward declarations
00017     class Bucket;
00018
00020     typedef std::list<Bucket*> BucketList_T;
00021
00023     typedef std::map<const MapKey_T, Bucket*> BucketMap_T;
00024
00025 }
00026 #endif // __STDAIR_BOM_BUCKETTYPES_HPP
00027
```

35.219 stdair/bom/CancellationStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_BookingClass.hpp>
#include <stdair/bom/CancellationStruct.hpp>
```

Namespaces

- namespace `stdair`

Handle on the StdAir library context.

35.220 CancellationStruct.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_BookingClass.hpp>
00009 #include <stdair/bom/CancellationStruct.hpp>
00010
00011 namespace stdair {
00012 // //////////////////////////////////////
00013 CancellationStruct::CancellationStruct (const SegmentPath_T& iSegPath,
00014                                         const ClassList_String_T& iList,
00015                                         const PartySize_T& iSize,
00016                                         const DateTime_T& iDateTime)
00017     : _segmentPath (iSegPath), _classList (iList), _partySize (iSize),
00018       _datetime (iDateTime) {
00019 }
00020
00021 // //////////////////////////////////////
00022 CancellationStruct::~CancellationStruct () {
00023 }
00024
00025 // //////////////////////////////////////
00026 void CancellationStruct::toStream (std::ostream& ioOut) const {
00027     ioOut << describe();
00028 }
00029
00030 // //////////////////////////////////////
00031 void CancellationStruct::fromStream (std::istream& ioIn) {
00032 }
00033
00034 // //////////////////////////////////////
00035 const std::string CancellationStruct::describe() const {
00036     std::ostringstream oStr;
00037
00038     oStr << "Segment path: ";
00039     unsigned short idx = 0;
00040     for (SegmentPath_T::const_iterator lItSegmentPath = _segmentPath.begin();
00041          lItSegmentPath != _segmentPath.end(); ++lItSegmentPath, ++idx) {
00042         if (idx != 0) {
00043             oStr << "-";
00044         }
00045         const std::string& lSegmentKey = *lItSegmentPath;
00046         oStr << lSegmentKey;
00047     }
00048
00049     oStr << ";" << _classList << ";" << _partySize << ";" << _datetime;
00050     return oStr.str();
00051 }
00052
00053 // //////////////////////////////////////

```

```

00054     const std::string CancellationStruct::display() const {
00055         std::ostringstream oStr;
00056
00057         // List of segment keys (one per segment)
00058         unsigned short idx = 0;
00059         for (SegmentPath_T::const_iterator itSegPath = _segmentPath.begin();
00060              itSegPath != _segmentPath.end(); ++itSegPath, ++idx) {
00061             if (idx != 0) {
00062                 oStr << " ; ";
00063             }
00064             const std::string& lSegmentKey = *itSegPath;
00065             oStr << "[" << idx << "]" " << lSegmentKey;
00066         }
00067
00068         oStr << ";" << _classList << ";" << _partySize << ";" << _datetime;
00069         return oStr.str();
00070     }
00071 }

```

35.221 stdair/bom/CancellationStruct.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <vector>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/BookingClassTypes.hpp>
#include <stdair/bom/TravelSolutionTypes.hpp>

```

Classes

- struct [stdair::CancellationStruct](#)
Structure holding the elements of a travel solution.

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.222 CancellationStruct.hpp

```

00001 #ifndef __STDAIR_BOM_CANCELLATIONSTRUCT_HPP
00002 #define __STDAIR_BOM_CANCELLATIONSTRUCT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL

```

```

00008 #include <iosfwd>
00009 #include <string>
00010 #include <vector>
00011 // StdAir
00012 #include <stdair/stdair_basic_types.hpp>
00013 #include <stdair/basic/StructAbstract.hpp>
00014 #include <stdair/bom/BookingClassTypes.hpp>
00015 #include <stdair/bom/TravelSolutionTypes.hpp>
00016
00017 namespace stdair {
00018
00022     struct CancellationStruct : public StructAbstract {
00023     public:
00024         // //////////// Getters ////////////
00026         const SegmentPath_T& getSegmentPath() const {
00027             return _segmentPath;
00028         }
00029
00031         const ClassList_String_T& getClassList() const {
00032             return _classList;
00033         }
00034
00036         const PartySize_T& getPartySize() const {
00037             return _partySize;
00038         }
00039
00041         const DateTime_T& getCancellationDateTime() const {
00042             return _datetime;
00043         }
00044
00045     public:
00046         // //////////// Display support method ////////////
00052         void toStream (std::ostream& ioOut) const;
00053
00058         void fromStream (std::istream& ioIn);
00059
00063         const std::string describe() const;
00064
00068         const std::string display() const;
00069
00070
00071     public:
00072         // //////////// Constructors & Destructor ////////////
00076         CancellationStruct (const SegmentPath_T&, const ClassList_String_T&,
00077                             const PartySize_T&, const DateTime_T&);
00078
00082         ~CancellationStruct();
00083
00084
00085     private:
00086         // //////////// Attributes ////////////
00090         SegmentPath_T _segmentPath;
00091
00095         ClassList_String_T _classList;
00096
00100         PartySize_T _partySize;
00101
00105         DateTime_T _datetime;
00106     };
00107
00108 }
00109 #endif // __STDAIR_BOM_CANCELLATIONSTRUCT_HPP

```

35.223 stdair/bom/CancellationTypes.hpp File Reference

```
#include <boost/shared_ptr.hpp>
```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

Typedefs

- typedef boost::shared_ptr< CancellationStruct > [stdair::CancellationPtr_T](#)

35.224 CancellationTypes.hpp

```
00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_CANCELLATIONTYPES_HPP
00003 #define __STDAIR_BOM_CANCELLATIONTYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // Boost
00009 #include <boost/shared_ptr.hpp>
00010
00011 namespace stdair {
00012
00013     // Forward declarations
00014     struct CancellationStruct;
00015
00016     // ////////////////////////////////// Type definitions //////////////////////////////////
00017     typedef boost::shared_ptr<CancellationStruct> CancellationPtr_T;
00018
00019
00020 }
00021 #endif // __STDAIR_BOM_CANCELLATIONTYPES_HPP
00022
```

35.225 stdair/bom/DatePeriod.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_Period_BOM.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/DatePeriod.hpp>
```

Namespaces

- namespace [stdair](#)

Handle on the StdAir library context.

35.226 DatePeriod.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_Period_BOM.hpp>
00009 #include <stdair/service/Logger.hpp>
00010 #include <stdair/bom/DatePeriod.hpp>
00011
00012 namespace stdair {
00013
00014 // //////////////////////////////////////
00015 DatePeriod::DatePeriod()
00016 : _key (BOOST_DEFAULT_DATE_PERIOD),
00017   _parent (NULL) {
00018     // That constructor is used by the serialisation process
00019 }
00020
00021 // //////////////////////////////////////
00022 DatePeriod::DatePeriod (const DatePeriod& iDatePeriod)
00023 : _key (iDatePeriod.getKey()), _parent (NULL) {
00024     assert (false);
00025 }
00026
00027 // //////////////////////////////////////
00028 DatePeriod::DatePeriod (const Key_T& iKey)
00029 : _key (iKey), _parent (NULL) {
00030 }
00031
00032 // //////////////////////////////////////
00033 DatePeriod::~DatePeriod () {
00034 }
00035
00036 // //////////////////////////////////////
00037 std::string DatePeriod::toString() const {
00038     std::ostringstream oStr;
00039     oStr << describeKey();
00040     return oStr.str();
00041 }
00042
00043 // //////////////////////////////////////
00044 bool DatePeriod::
00045 isDepartureDateValid (const Date_T& iFlightDate) const {
00046
00047     // Check if the departure date is within the date range.
00048     const DatePeriod_T& lPeriod = getDatePeriod ();
00049     if (lPeriod.contains (iFlightDate) == false) {
00050         return false;
00051     }
00052
00053     return true;
00054 }
00055
00056 }

```

00057

35.227 stdair/bom/DatePeriod.hpp File Reference

```
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/DatePeriodKey.hpp>
#include <stdair/bom/DatePeriodTypes.hpp>
```

Classes

- class [stdair::DatePeriod](#)
Class representing the actual attributes for a fare date-period.

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.228 DatePeriod.hpp

```
00001 #ifndef __STDAIR_BOM_DATEPERIOD_HPP
00002 #define __STDAIR_BOM_DATEPERIOD_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STDAIR
00008 #include <stdair/bom/BomAbstract.hpp>
00009 #include <stdair/bom/DatePeriodKey.hpp>
00010 #include <stdair/bom/DatePeriodTypes.hpp>
00011
00012 // Forward declaration
00013 namespace stdair {
00014
00015     class DatePeriod : public BomAbstract {
00016     public:
00017         template <typename BOM> friend class FacBom;
00018         friend class FacBomManager;
00019
00020     public:
00021         // ////////////////////////////////// Type definitions //////////////////////////////////
00022         typedef DatePeriodKey Key_T;
00023
00024     public:
00025         // ////////////////////////////////// Display support methods //////////////////////////////////
00026         void toStream (std::ostream& ioOut) const {
00027             ioOut << toString();
00028         }
00029
00030         void fromStream (std::istream& ioIn) {
00031
00032         }
00033
00034         std::string toString() const;
```

```

00052
00056     const std::string describeKey() const {
00057         return _key.toString();
00058     }
00059
00060 public:
00061     // //////////// Getters ////////////
00065     const Key_T& getKey() const {
00066         return _key;
00067     }
00068
00072     BomAbstract* const getParent() const {
00073         return _parent;
00074     }
00075
00079     const HolderMap_T& getHolderMap() const {
00080         return _holderMap;
00081     }
00082
00086     const DatePeriod_T& getDatePeriod() const {
00087         return _key.getDatePeriod();
00088     }
00089
00090
00091 public:
00092     // //////////// Business methods ////////////
00097     bool isDepartureDateValid (const Date_T&) const;
00098
00099 protected:
00100     // //////////// Constructors and destructors ////////////
00104     DatePeriod (const Key_T&);
00108     virtual ~DatePeriod ();
00109
00110 private:
00114     DatePeriod ();
00118     DatePeriod (const DatePeriod&);
00119
00120 protected:
00121     // //////////// Attributes ////////////
00125     Key_T _key;
00126
00130     BomAbstract* _parent;
00131
00135     HolderMap_T _holderMap;
00136
00137 };
00138
00139 }
00140 #endif // __STDAIR_BOM_DATEPERIOD_HPP
00141

```

35.229 stdair/bom/DatePeriodKey.cpp File Reference

```

#include <ostream>

#include <sstream>

#include <boost/date_time/gregorian/formatters.hpp>

#include <stdair/basic/BasConst_Period_BOM.hpp>

```

```
#include <stdair/bom/DatePeriodKey.hpp>
```

Namespaces

- namespace `stdair`

Handle on the StdAir library context.

35.230 DatePeriodKey.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <ostream>
00006 #include <sstream>
00007 // Boost Date-Time
00008 #include <boost/date_time/gregorian/formatters.hpp>
00009 // STDAIR
00010 #include <stdair/basic/BasConst_Period_BOM.hpp>
00011 #include <stdair/bom/DatePeriodKey.hpp>
00012
00013 namespace stdair {
00014
00015 // //////////////////////////////////////
00016 DatePeriodKey::DatePeriodKey()
00017     : _datePeriod (BOOST_DEFAULT_DATE_PERIOD) {
00018     assert (false);
00019 }
00020
00021 // //////////////////////////////////////
00022 DatePeriodKey::DatePeriodKey (const stdair::DatePeriod_T& iDatePeriod)
00023     : _datePeriod (iDatePeriod) {
00024 }
00025
00026 // //////////////////////////////////////
00027 DatePeriodKey::DatePeriodKey (const DatePeriodKey& iKey)
00028     : _datePeriod (iKey._datePeriod) {
00029 }
00030
00031 // //////////////////////////////////////
00032 DatePeriodKey::~DatePeriodKey () {
00033 }
00034
00035 // //////////////////////////////////////
00036 void DatePeriodKey::toStream (std::ostream& ioOut) const {
00037     ioOut << "DatePeriodKey: " << toString() << std::endl;
00038 }
00039
00040 // //////////////////////////////////////
00041 void DatePeriodKey::fromStream (std::istream& ioIn) {
00042 }
00043
00044 // //////////////////////////////////////
00045 const std::string DatePeriodKey::toString() const {
00046     std::ostringstream oStr;
00047     const stdair::Date_T lStart = _datePeriod.begin();
00048     const stdair::Date_T lEnd = _datePeriod.end();
00049     oStr << "[" << boost::gregorian::to_simple_string(lStart)
```

```

00050         << "/" << boost::gregorian::to_simple_string(lEnd)
00051         << "];
00052     return oStr.str();
00053 }
00054
00055 }
```

35.231 stdair/bom/DatePeriodKey.hpp File Reference

```

#include <stdair/bom/KeyAbstract.hpp>
#include <stdair/stdair_date_time_types.hpp>
```

Classes

- struct [stdair::DatePeriodKey](#)
Key of date-period.

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.232 DatePeriodKey.hpp

```

00001 #ifndef __SIMFQT_BOM_DATEPERIODKEY_HPP
00002 #define __SIMFQT_BOM_DATEPERIODKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STDAIR
00008 #include <stdair/bom/KeyAbstract.hpp>
00009 #include <stdair/stdair_date_time_types.hpp>
00010
00011 namespace stdair {
00014     struct DatePeriodKey : public KeyAbstract {
00015
00016     public:
00017         // ////////////////////////////////// Construction //////////////////////////////////
00019         DatePeriodKey (const DatePeriod_T&);
00021         DatePeriodKey (const DatePeriodKey&);
00023         ~DatePeriodKey ();
00024
00025     private:
00027         DatePeriodKey();
00028
00029     public:
00030         // ////////////////////////////////// Getters //////////////////////////////////
00032         const DatePeriod_T& getDatePeriod() const {
00033             return _datePeriod;
00034         }
00035 }
```

```

00036     public:
00037
00038         // //////////// Display support methods ////////////
00044         void toStream (std::ostream& ioOut) const;
00045
00051         void fromStream (std::istream& ioIn);
00052
00058         const std::string toString() const;
00059
00060     private:
00061         // //////////// Attributes ////////////
00065         DatePeriod_T _datePeriod;
00066
00067     };
00068
00069 }
00070 #endif // __SIMFQT_BOM_DATEPERIODKEY_HPP

```

35.233 stdair/bom/DatePeriodTypes.hpp File Reference

```

#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>

```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

Typedefs

- typedef std::list< DatePeriod * > [stdair::DatePeriodList_T](#)
- typedef std::map< const MapKey_T, DatePeriod * > [stdair::DatePeriodMap_T](#)
- typedef std::pair< MapKey_T, DatePeriod * > [stdair::DatePeriodWithKey_T](#)
- typedef std::list< DatePeriodWithKey_T > [stdair::DatePeriodDetailedList_T](#)

35.234 DatePeriodTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_DATEPERIODTYPES_HPP
00003 #define __STDAIR_BOM_DATEPERIODTYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // STDAIR
00012 #include <stdair/bom/key_types.hpp>
00013

```

```

00014 namespace stdair {
00015
00016     // Forward declarations.
00017     class DatePeriod;
00018
00020     typedef std::list<DatePeriod*> DatePeriodList_T;
00021
00023     typedef std::map<const MapKey_T, DatePeriod*> DatePeriodMap_T;
00024
00026     typedef std::pair<MapKey_T, DatePeriod*> DatePeriodWithKey_T;
00027     typedef std::list<DatePeriodWithKey_T> DatePeriodDetailedList_T;
00028 }
00029 #endif // __STDAIR_BOM_DATEPERIODTYPES_HPP
00030

```

35.235 stdair/bom/DoWStruct.cpp File Reference

```

#include <sstream>

#include <cassert>

#include <stdair/basic/BasConst_Period_BOM.hpp>

#include <stdair/bom/DoWStruct.hpp>

```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.236 DoWStruct.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <sstream>
00006 #include <cassert>
00007 // STDAIR
00008 #include <stdair/basic/BasConst_Period_BOM.hpp>
00009 #include <stdair/bom/DoWStruct.hpp>
00010
00011 namespace stdair {
00012
00013     // //////////////////////////////////////
00014     DoWStruct::DoWStruct () {
00015         for (unsigned short i = 0; i < 7; ++i) {
00016             _dowList.push_back (false);
00017         }
00018     }
00019
00020     // //////////////////////////////////////
00021     DoWStruct::DoWStruct (const std::string& iDowString) {
00022         const unsigned short lDowStringSize = iDowString.size();
00023         assert (lDowStringSize == 7);
00024

```

```

00025     _dowList.reserve (lDowStringSize);
00026     for (std::string::const_iterator itChar = iDowString.begin();
00027          itChar != iDowString.end(); ++itChar) {
00028         const bool isDoWSet = (*itChar == '1')?true:false;
00029         _dowList.push_back (isDoWSet);
00030     }
00031 }
00032
00033 // //////////////////////////////////////
00034 DoWStruct::DoWStruct (const DoWStruct& iDowStruct) :
00035     _dowList (iDowStruct._dowList) {
00036
00037 }
00038
00039 // //////////////////////////////////////
00040 const std::string DoWStruct::describeShort() const {
00041     std::ostream ostr;
00042     short i = 0;
00043     for (BooleanList_T::const_iterator itDoW = _dowList.begin();
00044          itDoW != _dowList.end(); ++itDoW, ++i) {
00045         const char lDoW = (*itDoW == true)?'1':'0';
00046         ostr << lDoW;
00047     }
00048     return ostr.str();
00049 }
00050
00051 // //////////////////////////////////////
00052 const std::string DoWStruct::describe() const {
00053     std::ostream ostr;
00054     short i = 0;
00055     for (BooleanList_T::const_iterator itDoW = _dowList.begin();
00056          itDoW != _dowList.end(); ++itDoW, ++i) {
00057         const bool lDoW = *itDoW;
00058         if (lDoW == true) {
00059             ostr << DOW_STR[i] << ".";
00060         }
00061     }
00062     return ostr.str();
00063 }
00064
00065 // //////////////////////////////////////
00066 bool DoWStruct::getDayOfWeek (const unsigned short i) const {
00067     return _dowList.at (i);
00068 }
00069
00070 // //////////////////////////////////////
00071 bool DoWStruct::getStandardDayOfWeek (const unsigned short i) const {
00072     unsigned short iStd = i;
00073     if (iStd == 0) {
00074         iStd = 6;
00075     } else {
00076         --iStd;
00077     }
00078     return _dowList.at (iStd);
00079 }
00080
00081 // //////////////////////////////////////
00082 void DoWStruct::setDayOfWeek (const unsigned short i, const bool iBool) {
00083     assert (i < 7);
00084     _dowList.at (i) = iBool;
00085 }
00086

```

```

00087 ///////////////////////////////////////////////////////////////////
00088 DoWStruct DoWStruct::shift (const long& iNbOfDays) const {
00089     DoWStruct oDoW (DEFAULT_DOW_STRING);
00090
00091     for (short i = 0; i < 7; ++i) {
00092         const bool lDoWBool = _dowList.at (i);
00093         short lIndex = (i + iNbOfDays) % 7;
00094         if (lIndex < 0) {
00095             lIndex += 7;
00096         }
00097         oDoW.setDayOfWeek (lIndex, lDoWBool);
00098     }
00099
00100     return oDoW;
00101 }
00102
00103 ///////////////////////////////////////////////////////////////////
00104 DoWStruct DoWStruct::intersection (const DoWStruct& iDoW) const {
00105     DoWStruct oDoW (DEFAULT_DOW_STRING);
00106     for (unsigned short i = 0; i < 7; ++i) {
00107         if (getDayOfWeek(i) && iDoW.getDayOfWeek(i)) {
00108             oDoW.setDayOfWeek (i, true);
00109         } else {
00110             oDoW.setDayOfWeek (i, false);
00111         }
00112     }
00113     return oDoW;
00114 }
00115
00116 ///////////////////////////////////////////////////////////////////
00117 const bool DoWStruct::isValid () const {
00118     for (unsigned short i = 0; i < 7; ++i) {
00119         if (getDayOfWeek(i)) {
00120             return true;
00121         }
00122     }
00123     return false;
00124 }
00125
00126 }

```

35.237 stdair/bom/DoWStruct.hpp File Reference

```

#include <string>

#include <vector>

#include <stdair/basic/StructAbstract.hpp>

```

Classes

- struct [stdair::DoWStruct](#)

Namespaces

- namespace [stdair](#)

Handle on the StdAir library context.

35.238 DoWStruct.hpp

```

00001 #ifndef __STDAIR_BOM_DOWSTRUCT_HPP
00002 #define __STDAIR_BOM_DOWSTRUCT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010 // STDAIR
00011 #include <stdair/basic/StructAbstract.hpp>
00012
00013 namespace stdair {
00014
00015     struct DoWStruct : public StructAbstract {
00016     public:
00021         typedef std::vector<bool> BooleanList_T;
00022
00023     public:
00024         // ////////////////////////////////// Getters //////////////////////////////////
00026         bool getDayOfWeek (const unsigned short i) const;
00027
00029         bool getStandardDayOfWeek (const unsigned short i) const;
00030
00031     public:
00032         // ////////////////////////////////// Setters //////////////////////////////////
00034         void setDayOfWeek (const unsigned short, const bool);
00035
00036     public:
00037         // ////////////////////////////////// Display methods //////////////////////////////////
00039         const std::string describe() const;
00040
00042         const std::string describeShort() const;
00043
00044     public:
00045         // ////////////////////////////////// Business Methods //////////////////////////////////
00047         DoWStruct shift (const long&) const;
00048
00050         DoWStruct intersection (const DoWStruct&) const;
00051
00053         const bool isValid () const;
00054
00055     public:
00058         DoWStruct (const std::string& iDowString);
00060         DoWStruct ();
00061         DoWStruct (const DoWStruct&);
00063         ~DoWStruct () { }
00064
00065     private:
00067         BooleanList_T _dowList;
00068     };
00069
00070 }
00071 #endif // __STDAIR_BOM_DOWSTRUCT_HPP

```

35.239 stdair/bom/EventQueue.cpp File Reference

```
#include <cassert>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/BasConst_Event.hpp>
#include <stdair/bom/EventStruct.hpp>
#include <stdair/bom/EventQueue.hpp>
#include <stdair/service/Logger.hpp>
```

Namespaces

- namespace [stdair](#)

Handle on the StdAir library context.

35.240 EventQueue.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // StdAir
00007 #include <stdair/stdair_exceptions.hpp>
00008 #include <stdair/basic/BasConst_Event.hpp>
00009 #include <stdair/bom/EventStruct.hpp>
00010 #include <stdair/bom/EventQueue.hpp>
00011 #include <stdair/service/Logger.hpp>
00012
00013 namespace stdair {
00014
00015 // //////////////////////////////////////
00016 EventQueue::EventQueue()
00017 : _key (DEFAULT_EVENT_QUEUE_ID), _parent (NULL),
00018   _progressStatus (stdair::DEFAULT_PROGRESS_STATUS,
00019                   stdair::DEFAULT_PROGRESS_STATUS) {
00020 }
00021
00022 // //////////////////////////////////////
00023 EventQueue::EventQueue (const Key_T& iKey)
00024 : _key (iKey), _parent (NULL),
00025   _progressStatus (stdair::DEFAULT_PROGRESS_STATUS,
00026                   stdair::DEFAULT_PROGRESS_STATUS) {
00027 }
00028
00029 // //////////////////////////////////////
00030 EventQueue::EventQueue (const EventQueue& iEventQueue)
00031 : _key (DEFAULT_EVENT_QUEUE_ID), _parent (NULL),
00032   _progressStatus (stdair::DEFAULT_PROGRESS_STATUS,
00033                   stdair::DEFAULT_PROGRESS_STATUS) {
00034   assert (false);
00035 }
00036
00037 // //////////////////////////////////////
```

```

00038 EventQueue::~EventQueue() {
00039     _eventList.clear();
00040 }
00041
00042 // //////////////////////////////////////
00043 std::string EventQueue::toString() const {
00044     std::ostringstream ostr;
00045     ostr << "(" << _eventList.size() << " "
00046         << _progressStatus.getCurrentNb() << "/" << "{"
00047         << _progressStatus.getExpectedNb() << ", "
00048         << _progressStatus.getActualNb() << "}";
00049     return ostr.str();
00050 }
00051
00052 // //////////////////////////////////////
00053 std::string EventQueue::display() const {
00054     std::ostringstream ostr;
00055
00056     ostr << toString();
00057
00058     return ostr.str();
00059 }
00060
00061 // //////////////////////////////////////
00062 Count_T EventQueue::getQueueSize() const {
00063     return _eventList.size();
00064 }
00065
00066 // //////////////////////////////////////
00067 bool EventQueue::isEmpty() const {
00068     return _eventList.empty();
00069 }
00070
00071 // //////////////////////////////////////
00072 bool EventQueue::isQueueDone() const {
00073     const bool isEmpty = _eventList.empty();
00074     return isEmpty;
00075 }
00076
00077 // //////////////////////////////////////
00078 void EventQueue::reset() {
00079     // Reset only the current number of events, not the expected one
00080     _progressStatus.reset();
00081
00082     // Empty the list of events
00083     _eventList.clear();
00084
00085     // Reset the progress statuses for all the event types
00086     for (ProgressStatusMap_T::iterator itProgressStatus =
00087         _progressStatusMap.begin();
00088         itProgressStatus != _progressStatusMap.end(); ++itProgressStatus) {
00089         ProgressStatus& lProgressStatus = *itProgressStatus->second;
00090         lProgressStatus.reset();
00091     }
00092 }
00093
00094 // //////////////////////////////////////
00095 const Count_T& EventQueue::
00096 getCurrentNbOfEvents (const EventType::EN_EventType& iType) const {
00097
00098     // Retrieve the ProgressStatus structure corresponding to the

```

```

00100 // given event type
00101 ProgressStatusMap_T::const_iterator itProgressStatus =
00102     _progressStatusMap.find (iType);
00103 if (itProgressStatus == _progressStatusMap.end()) {
00104     //
00105     STDAIR_LOG_ERROR ("No ProgressStatus structure can be retrieved in the "
00106         << "EventQueue: " << display());
00107     assert (false);
00108 }
00109
00110 const ProgressStatus& lProgressStatus = itProgressStatus->second;
00111 return lProgressStatus.getCurrentNb();
00112 }
00113
00114 // //////////////////////////////////////
00115 const Count_T& EventQueue::
00116 getExpectedTotalNbOfEvents (const EventType::EN_EventType& iType) const {
00117
00118     // Retrieve the ProgressStatus structure corresponding to the
00119     // given event type
00120     ProgressStatusMap_T::const_iterator itProgressStatus =
00121         _progressStatusMap.find (iType);
00122     if (itProgressStatus == _progressStatusMap.end()) {
00123         std::ostringstream ostr;
00124         ostr << "No ProgressStatus structure can be retrieved in the EventQueue '"
00125             << display() << "'. The EventQueue should be initialised, e.g., by "
00126             << "calling a buildSampleBom() method.";
00127         //
00128         STDAIR_LOG_ERROR (ostr.str());
00129         throw EventQueueException (ostr.str());
00130     }
00131
00132     const ProgressStatus& lProgressStatus = itProgressStatus->second;
00133     return lProgressStatus.getExpectedNb();
00134 }
00135
00136 // //////////////////////////////////////
00137 const Count_T& EventQueue::
00138 getActualTotalNbOfEvents (const EventType::EN_EventType& iType) const {
00139
00140     // Retrieve the ProgressStatus structure corresponding to the
00141     // given event type
00142     ProgressStatusMap_T::const_iterator itProgressStatus =
00143         _progressStatusMap.find (iType);
00144     if (itProgressStatus == _progressStatusMap.end()) {
00145         //
00146         STDAIR_LOG_ERROR ("No ProgressStatus structure can be retrieved in the "
00147             << "EventQueue: " << display());
00148         assert (false);
00149     }
00150
00151     const ProgressStatus& lProgressStatus = itProgressStatus->second;
00152     return lProgressStatus.getActualNb();
00153 }
00154
00155 // //////////////////////////////////////
00156 void EventQueue::updateStatus (const EventType::EN_EventType& iType,
00157     const ProgressStatus& iProgressStatus) {
00158
00159     // Retrieve, if existing, the ProgressStatus structure
00160     // corresponding to the given event type
00161     ProgressStatusMap_T::iterator itProgressStatus =

```

```

00162     _progressStatusMap.find (iType);
00163     if (itProgressStatus == _progressStatusMap.end()) {
00164         const bool hasInsertBeenSuccessful =
00165             _progressStatusMap.insert (ProgressStatusMap_T::
00166                 value_type (iType, iProgressStatus)).second;
00167
00168         if (hasInsertBeenSuccessful == false) {
00169             STDAIR_LOG_ERROR ("No progress_status can be inserted "
00170                 << "for the following event type: "
00171                 << EventType::getLabel(iType)
00172                 << ". EventQueue: " << toString());
00173             throw EventException ("No progress_status can be inserted for the "
00174                 "following event type: "
00175                 + EventType::getLabel(iType)
00176                 + ". EventQueue: " + toString());
00177         }
00178         return;
00179     }
00180
00181     ProgressStatus& lProgressStatus = itProgressStatus->second;
00182
00183     // Update the progress status
00184     const Count_T& lCurrentNb = iProgressStatus.getCurrentNb();
00185     lProgressStatus.setCurrentNb (lCurrentNb);
00186
00187     const Count_T& lExpectedNb = iProgressStatus.getExpectedNb();
00188     lProgressStatus.setExpectedNb(lProgressStatus.getExpectedNb() + lExpectedNb);
00189
00190
00191     const Count_T& lActualNb = iProgressStatus.getActualNb();
00192     lProgressStatus.setActualNb (lProgressStatus.getActualNb() + lActualNb);
00193 }
00194
00195 // //////////////////////////////////////
00196 void EventQueue::
00197 addStatus (const EventType::EN_EventType& iType,
00198     const NbOfEvents_T& iExpectedTotalNbOfEvents) {
00199
00200     // Initialise the progress status object
00201     const Count_T lExpectedTotalNbOfEventsInt =
00202         static_cast<const Count_T> (std::floor (iExpectedTotalNbOfEvents));
00203     const ProgressStatus lProgressStatus (lExpectedTotalNbOfEventsInt);
00204
00205     // Update the progress status for the given event type
00206     updateStatus (iType, lProgressStatus);
00207
00208     // Update the overall progress status
00209     const Count_T lExpectedNb =
00210         static_cast<const Count_T> (_progressStatus.getExpectedNb()
00211             + iExpectedTotalNbOfEvents);
00212     _progressStatus.setExpectedNb (lExpectedNb);
00213
00214     const Count_T lActualNb =
00215         static_cast<const Count_T> (_progressStatus.getActualNb()
00216             + iExpectedTotalNbOfEvents);
00217     _progressStatus.setActualNb (lActualNb);
00218 }
00219
00220 // //////////////////////////////////////
00221 void EventQueue::updateStatus (const EventType::EN_EventType& iType,
00222     const NbOfEvents_T& iActualNbOfEvents) {

```

```

00223
00224     // Initialise the progress status object for the type key
00225     Count_T lActualNbOfEventsInt =
00226         static_cast<const Count_T> (std::floor (iActualNbOfEvents));
00227
00228     // Update the progress status for the corresponding content type key
00229     ProgressStatusMap_T::iterator itProgressStatus =
00230         _progressStatusMap.find (iType);
00231     if (itProgressStatus != _progressStatusMap.end()) {
00232         ProgressStatus& lProgressStatus = itProgressStatus->second;
00233         //
00234         lActualNbOfEventsInt += lProgressStatus.getActualNb();
00235         //
00236         lProgressStatus.setActualNb (lActualNbOfEventsInt);
00237     }
00238 }
00239
00240 // //////////////////////////////////////
00241 void EventQueue::setStatus (const EventType::EN_EventType& iType,
00242                             const ProgressStatus& iProgressStatus) {
00243
00244     // Retrieve the ProgressStatus structure corresponding to the
00245     // given event type
00246     ProgressStatusMap_T::iterator itProgressStatus =
00247         _progressStatusMap.find (iType);
00248     // assert (itProgressStatus != _progressStatusMap.end());
00249     if (itProgressStatus != _progressStatusMap.end()) {
00250         // Update the ProgressStatus structure
00251         itProgressStatus->second = iProgressStatus;
00252     }
00253 }
00254
00255 // //////////////////////////////////////
00256 ProgressStatus EventQueue::
00257 getStatus (const EventType::EN_EventType& iType) const {
00258
00259     // Retrieve the ProgressStatus structure corresponding to the
00260     // given event type
00261     ProgressStatusMap_T::const_iterator itProgressStatus =
00262         _progressStatusMap.find (iType);
00263     if (itProgressStatus != _progressStatusMap.end()) {
00264         const ProgressStatus& oProgressStatus = itProgressStatus->second;
00265         return oProgressStatus;
00266     }
00267
00268     return ProgressStatus();
00269 }
00270
00271 // //////////////////////////////////////
00272 ProgressPercentage_T EventQueue::
00273 calculateProgress (const EventType::EN_EventType& iType) const {
00274
00275     // Retrieve the ProgressStatus structure corresponding to the
00276     // given event type
00277     ProgressStatusMap_T::const_iterator itProgressStatus =
00278         _progressStatusMap.find (iType);
00279     if (itProgressStatus == _progressStatusMap.end()) {
00280         //
00281         STDAIR_LOG_ERROR ("No ProgressStatus structure can be retrieved in the "
00282             << "EventQueue: " << display());
00283         assert (false);
00284     }

```

```

00285
00286     const ProgressStatus& lProgressStatus = itProgressStatus->second;
00287     return lProgressStatus.progress();
00288 }
00289
00290 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00291 ProgressStatusSet EventQueue::popEvent (EventStruct& ioEventStruct) {
00292     assert (_eventList.empty() == false);
00293
00294     // Get an iterator on the first event (sorted by date-time stamps)
00295     EventList_T::iterator itEvent = _eventList.begin();
00296
00297     ioEventStruct = itEvent->second;
00298     // Retrieve the event type
00299     const EventType::EN_EventType& lEventType = ioEventStruct.getEventType();
00300     ProgressStatusSet oProgressStatusSet (lEventType);
00301
00302     // Update the (current number part of the) overall progress status,
00303     // to account for the event that is being popped out of the event
00304     // queue.
00305     ++_progressStatus;
00306
00307     // Remove the event, which has just been retrieved
00308     _eventList.erase (itEvent);
00309
00310     // Retrieve the progress status specific to that event type
00311     ProgressStatus lEventTypeProgressStatus = getStatus (lEventType);
00312
00313     // Increase the current number of events
00314     ++lEventTypeProgressStatus;
00315
00316     // Store back the progress status
00317     setStatus (lEventType, lEventTypeProgressStatus);
00318
00319     // Update the progress status of the progress status set, specific to
00320     // the event type.
00321     oProgressStatusSet.setTypeSpecificStatus (lEventTypeProgressStatus);
00322
00323     // Update the overall progress status of the progress status set.
00324     oProgressStatusSet.setOverallStatus (_progressStatus);
00325
00326     //
00327     return oProgressStatusSet;
00328 }
00329
00330 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00331 bool EventQueue::addEvent (EventStruct& ioEventStruct) {
00332     bool insertionSucceeded =
00333         _eventList.insert (EventListElement_T (ioEventStruct._eventTimeStamp,
00334                                                 ioEventStruct)).second;
00335
00336     const unsigned int idx = 0;
00337     while (insertionSucceeded == false && idx != 1e3) {
00338         // Retrieve the date-time stamp (expressed in milliseconds)
00339         LongDuration_T& lEventTimeStamp (ioEventStruct._eventTimeStamp);
00340         ++lEventTimeStamp;
00341
00342         // Retry to insert into the event queue
00343         insertionSucceeded =
00344             _eventList.insert (EventListElement_T (ioEventStruct._eventTimeStamp,
00345                                                     ioEventStruct)).second;
00346     }
00347 }

```

```

00378     }
00379     assert (idx != 1e3);
00380
00381     return insertionSucceeded;
00382 }
00383
00384 }
```

35.241 stdair/bom/EventQueue.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/basic/ProgressStatusSet.hpp>
#include <stdair/basic/EventType.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/EventQueueKey.hpp>
#include <stdair/bom/EventQueueTypes.hpp>
#include <stdair/bom/EventTypes.hpp>
```

Classes

- class [stdair::EventQueue](#)
Class holding event structures.

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.242 EventQueue.hpp

```

00001 #ifndef __STDAIR_BOM_EVENTQUEUE_HPP
00002 #define __STDAIR_BOM_EVENTQUEUE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_basic_types.hpp>
00012 #include <stdair/basic/ProgressStatusSet.hpp>
00013 #include <stdair/basic/EventType.hpp>
00014 #include <stdair/bom/BomAbstract.hpp>
```

```

00015 #include <stdair/bom/EventQueueKey.hpp>
00016 #include <stdair/bom/EventQueueTypes.hpp>
00017 #include <stdair/bom/EventTypes.hpp>
00018
00019 namespace stdair {
00020
00059     class EventQueue : public BomAbstract {
00060     template <typename BOM> friend class FacBom;
00061     friend class FacBomManager;
00062
00063     public:
00064         // //////////// Type definitions ////////////
00068         typedef EventQueueKey Key_T;
00069
00075         typedef std::map<EventType::EN_EventType,
00076             ProgressStatus> ProgressStatusMap_T;
00077
00078
00079     public:
00080         // //////////// Getters ////////////
00082         const Key_T& getKey() const {
00083             return _key;
00084         }
00085
00087         BomAbstract* const getParent() const {
00088             return _parent;
00089         }
00090
00092         const HolderMap_T& getHolderMap() const {
00093             return _holderMap;
00094         }
00095
00097         const ProgressStatus& getStatus() const {
00098             return _progressStatus;
00099         }
00101         const Count_T& getCurrentNbOfEvents() const {
00102             return _progressStatus.getCurrentNb();
00103         }
00105         const Count_T& getExpectedTotalNbOfEvents() const {
00106             return _progressStatus.getExpectedNb();
00107         }
00109         const Count_T& getActualTotalNbOfEvents() const {
00110             return _progressStatus.getActualNb();
00111         }
00112
00117         ProgressStatus getStatus (const EventType::EN_EventType&) const;
00118
00120         const Count_T& getCurrentNbOfEvents (const EventType::EN_EventType&) const;
00121
00123         const Count_T& getExpectedTotalNbOfEvents (const EventType::EN_EventType&) co
nst;
00124
00126         const Count_T& getActualTotalNbOfEvents (const EventType::EN_EventType&) cons
t;
00127
00128     public:
00129         // //////////// Setters ////////////
00131         void setStatus (const ProgressStatus& iProgressStatus) {
00132             _progressStatus = iProgressStatus;
00133         }
00135         void setStatus (const Count_T& iCurrentNbOfEvents,
00136             const Count_T& iExpectedTotalNbOfEvents,

```

```

00137         const Count_T& iActualTotalNbOfEvents) {
00138     _progressStatus.setCurrentNb (iCurrentNbOfEvents);
00139     _progressStatus.setExpectedNb (iExpectedTotalNbOfEvents);
00140     _progressStatus.setActualNb (iActualTotalNbOfEvents);
00141 }
00142 void setStatus (const Count_T& iCurrentNbOfEvents,
00143                const Count_T& iActualTotalNbOfEvents) {
00144     _progressStatus.setCurrentNb (iCurrentNbOfEvents);
00145     _progressStatus.setActualNb (iActualTotalNbOfEvents);
00146 }
00147 void setCurrentNbOfEvents (const Count_T& iCurrentNbOfEvents) {
00148     _progressStatus.setCurrentNb (iCurrentNbOfEvents);
00149 }
00150 void setExpectedTotalNbOfEvents (const Count_T& iExpectedTotalNbOfEvents) {
00151     _progressStatus.setExpectedNb (iExpectedTotalNbOfEvents);
00152 }
00153 void setActualTotalNbOfEvents (const Count_T& iActualTotalNbOfEvents) {
00154     _progressStatus.setActualNb (iActualTotalNbOfEvents);
00155 }
00156 void setStatus (const EventType::EN_EventType& iType,
00157                const ProgressStatus& iProgressStatus);
00158
00159 public:
00160 // //////////// Display support methods ////////////
00161 void toStream (std::ostream& ioOut) const {
00162     ioOut << toString();
00163 }
00164
00165 void fromStream (std::istream& ioIn) {
00166 }
00167
00168 std::string toString() const;
00169
00170 const std::string describeKey() const {
00171     return _key.toString();
00172 }
00173
00174 /*
00175  * Display the full content of the event queue, with all its
00176  * demand streams.
00177  *
00178  * That method can be very consuming (in time, CPU and memory)
00179  * when there are a lot of demand streams (e.g., several hundreds
00180  * of thousands). Call it only for debug purposes.
00181  */
00182 std::string display() const;
00183
00184 public:
00185 // //////////// Business methods ////////////
00186 void reset();
00187
00188 ProgressStatusSet popEvent (EventStruct&);
00189
00190 bool addEvent (EventStruct&);
00191
00192 bool isQueueDone() const;
00193
00194 void addStatus (const EventType::EN_EventType&,
00195                const NbOfRequests_T& iExpectedTotalNbOfEvents);

```

```

00278
00287     void updateStatus (const EventType::EN_EventType&,
00288                       const ProgressStatus& iProgressStatus);
00289
00303     void updateStatus (const EventType::EN_EventType&,
00304                       const NbOfEvents_T& iActualTotalNbOfEvents);
00305
00316     ProgressPercentage_T calculateProgress() const {
00317         return _progressStatus.progress();
00318     }
00319
00330     ProgressPercentage_T calculateProgress(const EventType::EN_EventType&) const;
00331
00332
00333 public:
00334     // //////////// Debug methods ////////////
00336     Count_T getQueueSize() const;
00337
00339     bool isEmpty() const;
00340
00341
00342 protected:
00343     // //////////// Constructors and destructors ////////////
00345     EventQueue (const Key_T&);
00347     EventQueue (const EventQueue&);
00349     ~EventQueue();
00350 private:
00352     EventQueue();
00353
00354
00355 protected:
00356     // //////////// Attributes ////////////
00360     Key_T _key;
00361
00365     BomAbstract* _parent;
00366
00372     HolderMap_T _holderMap;
00373
00377     EventList_T _eventList;
00378
00382     ProgressStatus _progressStatus;
00383
00389     ProgressStatusMap_T _progressStatusMap;
00390 };
00391
00392 }
00393 #endif // __STDAIR_BOM_EVENTQUEUE_HPP

```

35.243 stdair/bom/EventQueueKey.cpp File Reference

```

#include <sstream>

#include <stdair/bom/EventQueueKey.hpp>

```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

35.244 EventQueueKey.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <sstream>
00006 // StdAir
00007 #include <stdair/bom/EventQueueKey.hpp>
00008
00009 namespace stdair {
00010
00011 // //////////////////////////////////////
00012 EventQueueKey::EventQueueKey (const EventQueueID_T& iEventQueueID)
00013     : _eventQueueID (iEventQueueID) {
00014 }
00015 // //////////////////////////////////////
00016 EventQueueKey::EventQueueKey (const EventQueueKey& iKey)
00017     : _eventQueueID (iKey._eventQueueID) {
00018 }
00019
00020 // //////////////////////////////////////
00021 EventQueueKey::~EventQueueKey () {
00022 }
00023
00024 // //////////////////////////////////////
00025 void EventQueueKey::toStream (std::ostream& ioOut) const {
00026     ioOut << "EventQueueKey: " << toString() << std::endl;
00027 }
00028
00029 // //////////////////////////////////////
00030 void EventQueueKey::fromStream (std::istream& ioIn) {
00031 }
00032
00033 // //////////////////////////////////////
00034 const std::string EventQueueKey::toString() const {
00035     std::ostringstream oStr;
00036     oStr << _eventQueueID;
00037     return oStr.str();
00038 }
00039
00040 }

```

35.245 stdair/bom/EventQueueKey.hpp File Reference

```

#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_event_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>

```

Classes

- struct [stdair::EventQueueKey](#)

Namespaces

- namespace `stdair`

Handle on the StdAir library context.

35.246 EventQueueKey.hpp

```

00001 #ifndef __STDAIR_BOM_EVENTQUEUEKEY_HPP
00002 #define __STDAIR_BOM_EVENTQUEUEKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009 #include <stdair/stdair_event_types.hpp>
00010 #include <stdair/bom/KeyAbstract.hpp>
00011
00012 namespace stdair {
00013
00014     struct EventQueueKey : public KeyAbstract {
00015
00016     private:
00017         // /////////// Default constructor ///////////
00018         EventQueueKey () { }
00019
00020     public:
00021         // /////////// Construction ///////////
00022         EventQueueKey (const EventQueueID_T&);
00023         EventQueueKey (const EventQueueKey&);
00024         ~EventQueueKey ();
00025
00026         // /////////// Getters ///////////
00027         const EventQueueID_T& getEventQueueID() const {
00028             return _eventQueueID;
00029         }
00030
00031         // /////////// Display support methods ///////////
00032         void toStream (std::ostream& ioOut) const;
00033
00034         void fromStream (std::istream& ioIn);
00035
00036         const std::string toString() const;
00037
00038     private:
00039         // /////////// Attributes ///////////
00040         EventQueueID_T _eventQueueID;
00041     };
00042 }
00043 #endif // __STDAIR_BOM_EVENTQUEUEKEY_HPP

```

35.247 stdair/bom/EventQueueTypes.hpp File Reference

```

#include <map>
#include <list>

```

```
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace `stdair`
Handle on the `StdAir` library context.

Typedefs

- typedef `std::list< EventQueue * >` `stdair::EventQueueList_T`
- typedef `std::map< const MapKey_T, EventQueue * >` `stdair::EventQueueMap_T`

35.248 EventQueueTypes.hpp

```
00001 ///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00002 #ifndef __STDAIR_BOM_EVENTQUEUEUETYPES_HPP
00003 #define __STDAIR_BOM_EVENTQUEUEUETYPES_HPP
00004
00005 ///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00006 // Import section
00007 ///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // Stdair
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016     // Forward declarations.
00017     class EventQueue;
00018
00020     typedef std::list<EventQueue*> EventQueueList_T;
00021
00023     typedef std::map<const MapKey_T, EventQueue*> EventQueueMap_T;
00024
00025 }
00026 #endif // __STDAIR_BOM_EVENTQUEUEUETYPES_HPP
```

35.249 stdair/bom/EventStruct.cpp File Reference

```
#include <cassert>
#include <boost/shared_ptr.hpp>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/BasConst_Event.hpp>
#include <stdair/bom/BookingRequestStruct.hpp>
#include <stdair/bom/OptimisationNotificationStruct.hpp>
#include <stdair/bom/SnapshotStruct.hpp>
```

```
#include <stdair/bom/CancellationStruct.hpp>
#include <stdair/bom/RMEventStruct.hpp>
#include <stdair/bom/EventStruct.hpp>
```

Namespaces

- namespace `stdair`
Handle on the `StdAir` library context.

35.250 EventStruct.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // Boost
00007 #if BOOST_VERSION >= 103900
00008 #include <boost/make_shared.hpp>
00009 #else // BOOST_VERSION >= 103900
00010 #include <boost/shared_ptr.hpp>
00011 #endif // BOOST_VERSION >= 103900
00012 // StdAir
00013 #include <stdair/basic/BasConst_General.hpp>
00014 #include <stdair/basic/BasConst_Event.hpp>
00015 #include <stdair/bom/BookingRequestStruct.hpp>
00016 #include <stdair/bom/OptimisationNotificationStruct.hpp>
00017 #include <stdair/bom/SnapshotStruct.hpp>
00018 #include <stdair/bom/CancellationStruct.hpp>
00019 #include <stdair/bom/RMEventStruct.hpp>
00020 #include <stdair/bom/EventStruct.hpp>
00021
00022 namespace stdair {
00023
00024 // //////////////////////////////////////
00025 EventStruct::EventStruct ()
00026 : _eventType (EventType::BKG_REQ), _eventTimeStamp (0) {
00027 }
00028
00029 // //////////////////////////////////////
00030 EventStruct::EventStruct (const EventType::EN_EventType& iEventType,
00031                          BookingRequestPtr_T ioRequestPtr)
00032 : _eventType (iEventType) {
00033
00034 //
00035 assert (ioRequestPtr != NULL);
00036 #if BOOST_VERSION >= 103900
00037 _bookingRequest = boost::make_shared<BookingRequestStruct> (*ioRequestPtr);
00038 #else // BOOST_VERSION >= 103900
00039 _bookingRequest = ioRequestPtr;
00040 #endif // BOOST_VERSION >= 103900
00041 assert (_bookingRequest != NULL);
00042
00048 const Duration_T lDuration =
00049 _bookingRequest->getRequestDateTime() - DEFAULT_EVENT_OLDEST_DATETIME;
00050 _eventTimeStamp = lDuration.total_milliseconds();
```

```

00051     }
00052
00053     // //////////////////////////////////////
00054     EventStruct::EventStruct (const EventType::EN_EventType& iEventType,
00055                               CancellationPtr_T ioCancellationPtr)
00056         : _eventType (iEventType) {
00057
00058         //
00059         assert (ioCancellationPtr != NULL);
00060         #if BOOST_VERSION >= 103900
00061             _cancellation = boost::make_shared<CancellationStruct> (*ioCancellationPtr);
00062         #else // BOOST_VERSION >= 103900
00063             _cancellation = ioCancellationPtr;
00064         #endif // BOOST_VERSION >= 103900
00065         assert (_cancellation != NULL);
00066
00072         const Duration_T lDuration =
00073             _cancellation->getCancellationDateTime() - DEFAULT_EVENT_OLDEST_DATETIME;
00074         _eventTimeStamp = lDuration.total_milliseconds();
00075     }
00076
00077     // //////////////////////////////////////
00078     EventStruct::
00079     EventStruct (const EventType::EN_EventType& iEventType,
00080                 const DateTime_T& iDCPDate,
00081                 OptimisationNotificationPtr_T ioOptimisationNotificationPtr)
00082         : _eventType (iEventType) {
00083
00084         //
00085         assert (ioOptimisationNotificationPtr != NULL);
00086         #if BOOST_VERSION >= 103900
00087             _optimisationNotification =
00088                 boost::make_shared<OptimisationNotificationStruct> (*ioOptimisationNotifica
00089 tionPtr);
00089         #else // BOOST_VERSION >= 103900
00090             _optimisationNotification = ioOptimisationNotificationPtr;
00091         #endif // BOOST_VERSION >= 103900
00092         assert (_optimisationNotification != NULL);
00093
00099         const Duration_T lDuration = iDCPDate - DEFAULT_EVENT_OLDEST_DATETIME;
00100         _eventTimeStamp = lDuration.total_milliseconds();
00101     }
00102
00103     // //////////////////////////////////////
00104     EventStruct::EventStruct (const EventType::EN_EventType& iEventType,
00105                               SnapshotPtr_T ioSnapshotPtr)
00106         : _eventType (iEventType) {
00107
00108         //
00109         assert (ioSnapshotPtr != NULL);
00110
00111         #if BOOST_VERSION >= 103900
00112             _snapshot = boost::make_shared<SnapshotStruct> (*ioSnapshotPtr);
00113         #else // BOOST_VERSION >= 103900
00114             _snapshot = ioSnapshotPtr;
00115         #endif // BOOST_VERSION >= 103900
00116         assert (_snapshot != NULL);
00117
00123         const Duration_T lDuration =
00124             _snapshot->getSnapshotTime() - DEFAULT_EVENT_OLDEST_DATETIME;
00125         _eventTimeStamp = lDuration.total_milliseconds();
00126     }

```

```

00127
00128 // //////////////////////////////////////
00129 EventStruct::EventStruct (const EventType::EN_EventType& iEventType,
00130                          RMEventPtr_T ioRMEventPtr)
00131 : _eventType (iEventType) {
00132
00133     //
00134     assert (ioRMEventPtr != NULL);
00135
00136     #if BOOST_VERSION >= 103900
00137         _rmEvent = boost::make_shared<RMEventStruct> (*ioRMEventPtr);
00138     #else // BOOST_VERSION >= 103900
00139         _rmEvent = ioRMEventPtr;
00140     #endif // BOOST_VERSION >= 103900
00141     assert (_rmEvent != NULL);
00142
00143     const Duration_T lDuration =
00144         _rmEvent->getRMEventTime() - DEFAULT_EVENT_OLDEST_DATETIME;
00145     _eventTimeStamp = lDuration.total_milliseconds();
00146 }
00147
00148 // //////////////////////////////////////
00149 EventStruct::EventStruct (const EventStruct& iEventStruct)
00150 : _eventType (iEventStruct._eventType),
00151   _eventTimeStamp (iEventStruct._eventTimeStamp) {
00152
00153     //
00154     if (iEventStruct._bookingRequest != NULL) {
00155     #if BOOST_VERSION >= 103900
00156         _bookingRequest =
00157             boost::make_shared<BookingRequestStruct> (*iEventStruct._bookingRequest);
00158     #else // BOOST_VERSION >= 103900
00159         _bookingRequest = iEventStruct._bookingRequest;
00160     #endif // BOOST_VERSION >= 103900
00161     }
00162
00163     //
00164     if (iEventStruct._cancellation != NULL) {
00165     #if BOOST_VERSION >= 103900
00166         _cancellation =
00167             boost::make_shared<CancellationStruct> (*iEventStruct._cancellation);
00168     #else // BOOST_VERSION >= 103900
00169         _cancellation = iEventStruct._cancellation;
00170     #endif // BOOST_VERSION >= 103900
00171     }
00172
00173     //
00174     if (iEventStruct._optimisationNotification != NULL) {
00175     #if BOOST_VERSION >= 103900
00176         _optimisationNotification =
00177             boost::make_shared<OptimisationNotificationStruct> (*iEventStruct._optimi
00178             sationNotification);
00179     #else // BOOST_VERSION >= 103900
00180         _optimisationNotification = iEventStruct._optimisationNotification;
00181     #endif // BOOST_VERSION >= 103900
00182     }
00183
00184     //
00185     if (iEventStruct._snapshot != NULL) {
00186     #if BOOST_VERSION >= 103900
00187         _snapshot = boost::make_shared<SnapshotStruct> (*iEventStruct._snapshot);
00188     #else // BOOST_VERSION >= 103900

```

```

00193         _snapshot = iEventStruct._snapshot;
00194 #endif // BOOST_VERSION >= 103900
00195     }
00196
00197     //
00198     if (iEventStruct._rmEvent != NULL) {
00199 #if BOOST_VERSION >= 103900
00200         _rmEvent = boost::make_shared<RMEventStruct> (*iEventStruct._rmEvent);
00201 #else // BOOST_VERSION >= 103900
00202         _rmEvent = iEventStruct._rmEvent;
00203 #endif // BOOST_VERSION >= 103900
00204     }
00205 }
00206
00207 // //////////////////////////////////////
00208 EventStruct::~EventStruct() {
00209 }
00210
00211 // //////////////////////////////////////
00212 void EventStruct::fromStream (std::istream& ioIn) {
00213 }
00214
00215 // //////////////////////////////////////
00216 const std::string EventStruct::describe() const {
00217     std::ostringstream oStr;
00218
00219     //
00220     const Duration_T lEventDateTimeDelta =
00221         boost::posix_time::milliseconds (_eventTimeStamp);
00222     const DateTime_T lEventDateTime (DEFAULT_EVENT_OLDEST_DATETIME
00223                                     + lEventDateTimeDelta);
00224     oStr << lEventDateTime;
00225
00226     //
00227     switch (_eventType) {
00228     case EventType::BKG_REQ: {
00229         assert (_bookingRequest != NULL);
00230         oStr << ", " << _eventType << ", " << _bookingRequest->describe();
00231         break;
00232     }
00233     case EventType::CX: {
00234         assert (_cancellation != NULL);
00235         oStr << ", " << _eventType << ", " << _cancellation->describe();
00236         break;
00237     }
00238     case EventType::OPT_NOT_4_FD: {
00239         assert (_optimisationNotification != NULL);
00240         oStr << ", " << _eventType
00241             << ", " << _optimisationNotification->describe();
00242         break;
00243     }
00244     case EventType::SNAPSHOT: {
00245         assert (_snapshot != NULL);
00246         oStr << ", " << _eventType
00247             << ", " << _snapshot->describe();
00248         break;
00249     }
00250     case EventType::RM: {
00251         assert (_rmEvent != NULL);
00252         oStr << ", " << _eventType
00253             << ", " << _rmEvent->describe();
00254         break;

```

```

00255     }
00256     default: {
00257         ostr << ", " << _eventType << " (not yet recognised)";
00258         break;
00259     }
00260 }
00261
00262     return ostr.str();
00263 }
00264
00265 }
```

35.251 stdair/bom/EventStruct.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/stdair_event_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/basic/EventType.hpp>
#include <stdair/bom/EventTypes.hpp>
#include <stdair/bom/BookingRequestTypes.hpp>
#include <stdair/bom/OptimisationNotificationTypes.hpp>
#include <stdair/bom/SnapshotTypes.hpp>
#include <stdair/bom/CancellationTypes.hpp>
#include <stdair/bom/RMEventTypes.hpp>
```

Classes

- struct [stdair::EventStruct](#)

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.252 EventStruct.hpp

```

00001 #ifndef __STDAIR_BAS_EVENTSTRUCT_HPP
00002 #define __STDAIR_BAS_EVENTSTRUCT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
```

```

00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_basic_types.hpp>
00012 #include <stdair/stdair_date_time_types.hpp>
00013 #include <stdair/stdair_event_types.hpp>
00014 #include <stdair/basic/StructAbstract.hpp>
00015 #include <stdair/basic/EventType.hpp>
00016 #include <stdair/bom/EventTypes.hpp>
00017 #include <stdair/bom/BookingRequestTypes.hpp>
00018 #include <stdair/bom/OptimisationNotificationTypes.hpp>
00019 #include <stdair/bom/SnapshotTypes.hpp>
00020 #include <stdair/bom/CancellationTypes.hpp>
00021 #include <stdair/bom/RMEventTypes.hpp>
00022
00023 namespace stdair {
00024
00035     struct EventStruct : public StructAbstract {
00036         // Friend classes and structures
00037         friend struct EventQueue;
00038
00039         // ////////////////////////////////// Getters //////////////////////////////////
00040     public:
00042         const EventType::EN_EventType& getEventType() const {
00043             return _eventType;
00044         }
00045
00052         const BookingRequestStruct& getBookingRequest() const {
00053             assert (_bookingRequest != NULL);
00054             return *_bookingRequest;
00055         }
00056
00063         const CancellationStruct& getCancellation() const {
00064             assert (_cancellation != NULL);
00065             return *_cancellation;
00066         }
00067
00075         const OptimisationNotificationStruct&
00076         getOptimisationNotificationStruct() const {
00077             assert (_optimisationNotification != NULL);
00078             return *_optimisationNotification;
00079         }
00080
00088         const SnapshotStruct& getSnapshotStruct() const {
00089             assert (_snapshot != NULL);
00090             return *_snapshot;
00091         }
00092
00100         const RMEventStruct& getRMEvent() const {
00101             assert (_rmEvent != NULL);
00102             return *_rmEvent;
00103         }
00104
00105         // ////////////////////////////////// Display methods //////////////////////////////////
00106     public:
00109         void fromStream (std::istream& ioIn);
00110
00112         const std::string describe() const;
00113
00114

```

```

00115 // ////////// Constructors and destructors //////////
00116 public:
00117     EventStruct();
00120     EventStruct(const EventType::EN_EventType&, BookingRequestPtr_T);
00122     EventStruct(const EventType::EN_EventType&, CancellationPtr_T);
00124     EventStruct(const EventType::EN_EventType&, const DateTime_T& iDCPDate,
00125                 OptimisationNotificationPtr_T);
00127     EventStruct(const EventType::EN_EventType&, SnapshotPtr_T);
00129     EventStruct(const EventType::EN_EventType&, RMEventPtr_T);
00131     EventStruct(const EventStruct&);
00132
00133     ~EventStruct();
00134
00135 // ////////// Attributes //////////
00136 private:
00137     EventType::EN_EventType _eventType;
00142     LongDuration_T _eventTimeStamp;
00143
00149     BookingRequestPtr_T _bookingRequest;
00150
00154     CancellationPtr_T _cancellation;
00155
00159     OptimisationNotificationPtr_T _optimisationNotification;
00160
00164     SnapshotPtr_T _snapshot;
00165
00169     RMEventPtr_T _rmEvent;
00174
00175 };
00176
00177 }
00178 #endif // __STDAIR_BAS_EVENTSTRUCT_HPP

```

35.253 stdair/bom/EventTypes.hpp File Reference

```

#include <map>
#include <boost/shared_ptr.hpp>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/stdair_event_types.hpp>
#include <stdair/basic/ProgressStatus.hpp>
#include <stdair/bom/key_types.hpp>

```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

Typedefs

- typedef std::pair< const LongDuration_T, EventStruct > [stdair::EventListElement_T](#)
- typedef std::map< const LongDuration_T, EventStruct > [stdair::EventList_T](#)

35.254 EventTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_EVENTTYPES_HPP
00003 #define __STDAIR_BOM_EVENTTYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <map>
00010 // Boost Smart Pointers
00011 #include <boost/shared_ptr.hpp>
00012 // StdAir
00013 #include <stdair/stdair_basic_types.hpp>
00014 #include <stdair/stdair_date_time_types.hpp>
00015 #include <stdair/stdair_event_types.hpp>
00016 #include <stdair/basic/ProgressStatus.hpp>
00017 #include <stdair/bom/key_types.hpp>
00018
00019 namespace stdair {
00020
00022     struct EventStruct;
00023
00027     typedef std::pair<const LongDuration_T, EventStruct> EventListElement_T;
00028
00032     typedef std::map<const LongDuration_T, EventStruct> EventList_T;
00033 }
00034 #endif // __STDAIR_BOM_EVENTTYPES_HPP
00035

```

35.255 stdair/bom/FareFamily.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/FareFamily.hpp>

```

Namespaces

- namespace [stdair](#)

Handle on the StdAir library context.

35.256 FareFamily.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/access.hpp>
00011 // StdAir
00012 #include <stdair/basic/BasConst_Inventory.hpp>
00013 #include <stdair/bom/FareFamily.hpp>
00014
00015 namespace stdair {
00016
00017 // //////////////////////////////////////
00018 FareFamily::FareFamily() : _key (DEFAULT_FARE_FAMILY_CODE), _parent (NULL) {
00019     assert (false);
00020 }
00021
00022 // //////////////////////////////////////
00023 FareFamily::FareFamily (const FareFamily&)
00024 : _key (DEFAULT_FARE_FAMILY_CODE), _parent (NULL) {
00025     assert (false);
00026 }
00027
00028 // //////////////////////////////////////
00029 FareFamily::FareFamily (const Key_T& iKey) : _key (iKey), _parent (NULL) {
00030 }
00031
00032 // //////////////////////////////////////
00033 FareFamily::~FareFamily() {
00034 }
00035
00036 // //////////////////////////////////////
00037 std::string FareFamily::toString() const {
00038     std::ostringstream oStr;
00039     oStr << describeKey();
00040     return oStr.str();
00041 }
00042
00043 // //////////////////////////////////////
00044 void FareFamily::serialisationImplementationExport() const {
00045     std::ostringstream oStr;
00046     boost::archive::text_oarchive oa (oStr);
00047     oa << *this;
00048 }
00049
00050 // //////////////////////////////////////
00051 void FareFamily::serialisationImplementationImport() {
00052     std::istringstream iStr;
00053     boost::archive::text_iarchive ia (iStr);
00054     ia >> *this;
00055 }
00056

```

```

00057 // //////////////////////////////////////
00058 template<class Archive>
00059 void FareFamily::serialize (Archive& ioArchive,
00060                             const unsigned int iFileVersion) {
00061     ioArchive & _key;
00062 }
00063
00064 }
00065
00066

```

35.257 stdair/bom/FareFamily.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/FareFamilyKey.hpp>
#include <stdair/bom/FareFamilyTypes.hpp>

```

Classes

- class [stdair::FareFamily](#)
Class representing the actual attributes for a family fare.

Namespaces

- namespace [boost](#)
Forward declarations.
- namespace [boost::serialization](#)
- namespace [stdair](#)
Handle on the StdAir library context.

35.258 FareFamily.hpp

```

00001 #ifndef __STDAIR_BOM_FAREFAMILY_HPP
00002 #define __STDAIR_BOM_FAREFAMILY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/bom/BomAbstract.hpp>
00012 #include <stdair/bom/FareFamilyKey.hpp>
00013 #include <stdair/bom/FareFamilyTypes.hpp>
00014
00016 namespace boost {

```

```

00017 namespace serialization {
00018     class access;
00019 }
00020 }
00021
00022 namespace stdair {
00023
00027 class FareFamily : public BomAbstract {
00028     template <typename BOM> friend class FacBom;
00029     friend class FacBomManager;
00030     friend class boost::serialization::access;
00031
00032 public:
00033     // //////////// Type definitions ////////////
00037     typedef FareFamilyKey Key_T;
00038
00039 public:
00041     // //////////// Getters ////////////
00043     const Key_T& getKey() const {
00044         return _key;
00045     }
00046
00048     BomAbstract* const getParent() const {
00049         return _parent;
00050     }
00051
00053     const FamilyCode_T& getFamilyCode() const {
00054         return _key.getFamilyCode();
00055     }
00056
00058     const HolderMap_T& getHolderMap() const {
00059         return _holderMap;
00060     }
00061
00062 public:
00064     // //////////// Display support methods ////////////
00070     void toStream (std::ostream& ioOut) const {
00071         ioOut << toString();
00072     }
00073
00079     void fromStream (std::istream& ioIn) {
00080     }
00081
00085     std::string toString() const;
00086
00090     const std::string describeKey() const {
00091         return _key.toString();
00092     }
00093
00094 public:
00096     // //////////// (Boost) Serialisation support methods ////////////
00100     template<class Archive>
00101     void serialize (Archive& ar, const unsigned int iFileVersion);
00102
00103 private:
00108     void serialisationImplementationExport() const;
00109     void serialisationImplementationImport();
00110
00111

```

```

00112     protected:
00113         // ////////// Constructors and destructors //////////
00117         FareFamily (const Key_T&);
00118
00122         virtual ~FareFamily();
00123
00124     private:
00128         FareFamily();
00129
00133         FareFamily (const FareFamily&);
00134
00135
00136     public:
00137         // ////////// Attributes //////////
00141         Key_T _key;
00142
00146         BomAbstract* _parent;
00147
00151         HolderMap_T _holderMap;
00152     };
00153
00154 }
00155 #endif // __STDAIR_BOM_FAREFAMILY_HPP
00156

```

35.259 stdair/bom/FareFamilyKey.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/FareFamilyKey.hpp>

```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

Functions

- template void `stdair::FareFamilyKey::serialize< ba::text_oarchive >` (ba::text_oarchive &, unsigned int)
- template void `stdair::FareFamilyKey::serialize< ba::text_iarchive >` (ba::text_iarchive &, unsigned int)

35.260 FareFamilyKey.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/access.hpp>
00011 // StdAir
00012 #include <stdair/basic/BasConst_Inventory.hpp>
00013 #include <stdair/bom/FareFamilyKey.hpp>
00014
00015 namespace stdair {
00016
00017 // //////////////////////////////////////
00018 FareFamilyKey::FareFamilyKey() : _familyCode (DEFAULT_FARE_FAMILY_CODE) {
00019     assert (false);
00020 }
00021
00022 // //////////////////////////////////////
00023 FareFamilyKey::FareFamilyKey (const FareFamilyKey& iFareFamilyKey)
00024 : _familyCode (iFareFamilyKey._familyCode) {
00025 }
00026
00027 // //////////////////////////////////////
00028 FareFamilyKey::FareFamilyKey (const FamilyCode_T& iFamilyCode)
00029 : _familyCode (iFamilyCode) {
00030 }
00031
00032 // //////////////////////////////////////
00033 FareFamilyKey::~FareFamilyKey() {
00034 }
00035
00036 // //////////////////////////////////////
00037 void FareFamilyKey::toStream (std::ostream& ioOut) const {
00038     ioOut << "FareFamilyKey: " << toString();
00039 }
00040
00041 // //////////////////////////////////////
00042 void FareFamilyKey::fromStream (std::istream& ioIn) {
00043 }
00044
00045 // //////////////////////////////////////
00046 const std::string FareFamilyKey::toString() const {
00047     std::ostringstream oStr;
00048     oStr << _familyCode;
00049     return oStr.str();
00050 }
00051
00052 // //////////////////////////////////////
00053 void FareFamilyKey::serialisationImplementationExport() const {
00054     std::ostringstream oStr;
00055     boost::archive::text_oarchive oa (oStr);
00056     oa << *this;
00057 }
00058
00059 // //////////////////////////////////////
00060 void FareFamilyKey::serialisationImplementationImport() {

```

```

00061     std::istringstream iStr;
00062     boost::archive::text_iarchive ia (iStr);
00063     ia >> *this;
00064 }
00065
00066 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00067 template<class Archive>
00068 void FareFamilyKey::serialize (Archive& ioArchive,
00069                               const unsigned int iFileVersion) {
00074     ioArchive & _familyCode;
00075 }
00076
00077 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00078 // Explicit template instantiation
00079 namespace ba = boost::archive;
00080 template void FareFamilyKey::serialize<ba::text_oarchive> (ba::text_oarchive&,
00081                                                         unsigned int);
00082 template void FareFamilyKey::serialize<ba::text_iarchive> (ba::text_iarchive&,
00083                                                         unsigned int);
00084 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00085
00086 }

```

35.261 stdair/bom/FareFamilyKey.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>

```

Classes

- struct [stdair::FareFamilyKey](#)
Key of a given fare family, made of a fare family code.

Namespaces

- namespace [boost](#)
Forward declarations.
- namespace [boost::serialization](#)
- namespace [stdair](#)
Handle on the StdAir library context.

35.262 FareFamilyKey.hpp

```

00001 #ifndef __STDAIR_BOM_FAREFAMILYKEY_HPP
00002 #define __STDAIR_BOM_FAREFAMILYKEY_HPP
00003
00004 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00005 // Import section

```

```

00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/bom/KeyAbstract.hpp>
00013
00015 namespace boost {
00016     namespace serialization {
00017         class access;
00018     }
00019 }
00020
00021 namespace stdair {
00022
00026     struct FareFamilyKey : public KeyAbstract {
00027         friend class boost::serialization::access;
00028
00029         // ////////////////////////////////// Constructors and destructors //////////////////////////////////
00030     private:
00031         FareFamilyKey();
00032
00033     public:
00034         FareFamilyKey (const FamilyCode_T& iFamilyCode);
00035
00036         FareFamilyKey (const FareFamilyKey&);
00037
00038         ~FareFamilyKey();
00039
00040     public:
00041         // ////////////////////////////////// Getters //////////////////////////////////
00042         const FamilyCode_T& getFamilyCode () const {
00043             return _familyCode;
00044         }
00045
00046     public:
00047         // ////////////////////////////////// Display support methods //////////////////////////////////
00048         void toStream (std::ostream& ioOut) const;
00049
00050         void fromStream (std::istream& ioIn);
00051
00052         const std::string toString() const;
00053
00054     public:
00055         // ////////////////////////////////// (Boost) Serialisation support methods //////////////////////////////////
00056         template<class Archive>
00057         void serialize (Archive& ar, const unsigned int iFileVersion);
00058
00059     private:
00060         void serialisationImplementationExport() const;
00061         void serialisationImplementationImport();
00062
00063     private:
00064         // ////////////////////////////////// Attributes //////////////////////////////////
00065         FamilyCode_T _familyCode;
00066     };
00067
00068
00069

```

```
00114 }
00115 #endif // __STDAIR_BOM_FAREFAMILYKEY_HPP
```

35.263 stdair/bom/FareFamilyTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

Typedefs

- typedef std::list< FareFamily * > [stdair::FareFamilyList_T](#)
- typedef std::map< const MapKey_T, FareFamily * > [stdair::FareFamilyMap_T](#)

35.264 FareFamilyTypes.hpp

```
00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_FAREFAMILYTYPES_HPP
00003 #define __STDAIR_BOM_FAREFAMILYTYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // StdAir
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016     // Forward declarations.
00017     class FareFamily;
00018
00020     typedef std::list<FareFamily*> FareFamilyList_T;
00021
00023     typedef std::map<const MapKey_T, FareFamily*> FareFamilyMap_T;
00024 }
00025 #endif // __STDAIR_BOM_FAREFAMILYTYPES_HPP
```

35.265 stdair/bom/FareFeatures.cpp File Reference

```
#include <cassert>
#include <sstream>
```

```
#include <stdair/basic/BasConst_DefaultObject.hpp>
#include <stdair/basic/BasConst_Request.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/FareFeatures.hpp>
```

Namespaces

- namespace `stdair`

Handle on the StdAir library context.

35.266 FareFeatures.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_DefaultObject.hpp>
00009 #include <stdair/basic/BasConst_Request.hpp>
00010 #include <stdair/service/Logger.hpp>
00011 #include <stdair/bom/FareFeatures.hpp>
00012
00013 namespace stdair {
00014
00015     // //////////////////////////////////////
00016     FareFeatures::FareFeatures()
00017         : _key (TRIP_TYPE_ONE_WAY,
00018               NO_ADVANCE_PURCHASE,
00019               SATURDAY_STAY,
00020               CHANGE_FEES,
00021               NON_REFUNDABLE,
00022               NO_STAY_DURATION),
00023         _parent (NULL) {
00024         // That constructor is used by the serialisation process
00025     }
00026
00027     // //////////////////////////////////////
00028     FareFeatures::FareFeatures (const FareFeatures& iFeatures)
00029         : _key (iFeatures.getKey()), _parent (NULL) {
00030         assert (false);
00031     }
00032
00033     // //////////////////////////////////////
00034     FareFeatures::FareFeatures (const Key_T& iKey)
00035         : _key (iKey), _parent (NULL) {
00036     }
00037
00038     // //////////////////////////////////////
00039     FareFeatures::~FareFeatures () {
00040     }
00041
00042     // //////////////////////////////////////
00043     std::string FareFeatures::toString() const {
00044         std::ostringstream oStr;
```

```

00045     oStr << describeKey();
00046     return oStr.str();
00047 }
00048
00049 ///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00050 bool FareFeatures::
00051 isTripTypeValid (const TripType_T& iBookingRequestTripType) const {
00052     bool oIsTripTypeValidFlag = true;
00053
00054     const TripType_T lFareTripType = getTripType();
00055     // Check whether the fare trip type is the same as the booking request
00056     // trip type
00057     if (iBookingRequestTripType == lFareTripType) {
00058         // One way case
00059         return oIsTripTypeValidFlag;
00060     }
00061
00062     if (iBookingRequestTripType == TRIP_TYPE_INBOUND
00063         || iBookingRequestTripType == TRIP_TYPE_OUTBOUND) {
00064         // Round trip case
00065         if (lFareTripType == TRIP_TYPE_ROUND_TRIP) {
00066             return oIsTripTypeValidFlag;
00067         }
00068     }
00069
00070     oIsTripTypeValidFlag = false;
00071     return oIsTripTypeValidFlag;
00072 }
00073
00074 ///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00075 bool FareFeatures::
00076 isStayDurationValid (const DayDuration_T& iStayDuration) const {
00077
00078     // Check if the stay duration is lower or equal to the minimum one.
00079     const DayDuration_T lMinimumDayDuration = getMinimumStay();
00080     if (lMinimumDayDuration > iStayDuration) {
00081         return false;
00082     }
00083
00084     return true;
00085 }
00086
00087 ///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00088 bool FareFeatures::
00089 isAdvancePurchaseValid (const DateTime_T& iBookingRequestDateTime,
00090                        const DateTime_T& iFlightDateTime) const {
00091     bool oIsAdvancePurchaseValidFlag = true;
00092
00093     // Check whether the departure date is within the date range.
00094     const DayDuration_T lAdvancedPurchase = getAdvancePurchase();
00095     const DateOffset_T lMinimumAdvancedPurchase (lAdvancedPurchase);
00096     const DateTime_T lCriticalDate = iFlightDateTime - lMinimumAdvancedPurchase;
00097
00098     if (lCriticalDate < iBookingRequestDateTime) {
00099         oIsAdvancePurchaseValidFlag = false;
00100         return oIsAdvancePurchaseValidFlag;
00101     }
00102
00103     return true;
00104 }
00105
00106 }

```

00107

35.267 stdair/bom/FareFeatures.hpp File Reference

```
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/FareFeaturesKey.hpp>
#include <stdair/bom/FareFeaturesTypes.hpp>
```

Classes

- class [stdair::FareFeatures](#)
Class representing the actual attributes for a fare date-period.

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.268 FareFeatures.hpp

```
00001 #ifndef __STDAIR_BOM_FAREFEATURES_HPP
00002 #define __STDAIR_BOM_FAREFEATURES_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/bom/BomAbstract.hpp>
00009 #include <stdair/bom/FareFeaturesKey.hpp>
00010 #include <stdair/bom/FareFeaturesTypes.hpp>
00011
00012 // Forward declaration
00013 namespace stdair {
00014
00018     class FareFeatures : public BomAbstract {
00019         template <typename BOM> friend class FacBom;
00020         friend class FacBomManager;
00021
00022     public:
00023         // ////////////////////////////////// Type definitions //////////////////////////////////
00027         typedef FareFeaturesKey Key_T;
00028
00029     public:
00030         // ////////////////////////////////// Display support methods //////////////////////////////////
00036         void toStream (std::ostream& ioOut) const {
00037             ioOut << toString();
00038         }
00039
00045         void fromStream (std::istream& ioIn) {
00046         }
00047
00051         std::string toString() const;
```

```
00052
00056     const std::string describeKey() const {
00057         return _key.toString();
00058     }
00059
00060
00061 public:
00062     // //////////// Getters ////////////
00066     const Key_T& getKey() const {
00067         return _key;
00068     }
00069
00073     BomAbstract* const getParent() const {
00074         return _parent;
00075     }
00076
00080     const HolderMap_T& getHolderMap() const {
00081         return _holderMap;
00082     }
00083
00087     const TripType_T& getTripType() const {
00088         return _key.getTripType();
00089     }
00090
00094     const DayDuration_T& getAdvancePurchase() const {
00095         return _key.getAdvancePurchase();
00096     }
00097
00101     const SaturdayStay_T& getSaturdayStay() const {
00102         return _key.getSaturdayStay();
00103     }
00104
00108     const ChangeFees_T& getChangeFees() const {
00109         return _key.getChangeFees();
00110     }
00111
00115     const NonRefundable_T& getRefundableOption() const {
00116         return _key.getRefundableOption();
00117     }
00118
00122     const DayDuration_T& getMinimumStay() const {
00123         return _key.getMinimumStay();
00124     }
00125
00126
00127 public:
00128     // //////////// Business methods ////////////
00133     bool isTripTypeValid (const TripType_T&) const;
00134
00139     bool isStayDurationValid (const DayDuration_T&) const;
00140
00145     bool isAdvancePurchaseValid (const DateTime_T& iBookingRequestDateTime,
00146                                 const DateTime_T& iFlightDateTime) const;
00147
00148
00149 protected:
00150     // //////////// Constructors and destructors ////////////
00154     FareFeatures (const Key_T&);
00158     virtual ~FareFeatures ();
00159
00160 private:
00164     FareFeatures ();
```

```

00168     FareFeatures (const FareFeatures&);
00169
00170 protected:
00171     // //////////// Attributes ////////////
00175     Key_T _key;
00176
00180     BomAbstract* _parent;
00181
00185     HolderMap_T _holderMap;
00186 };
00187
00188 }
00189 #endif // __STDAIR_BOM_FAREFEATURES_HPP
00190

```

35.269 stdair/bom/FareFeaturesKey.cpp File Reference

```

#include <ostream>

#include <sstream>

#include <stdair/basic/BasConst_DefaultObject.hpp>
#include <stdair/basic/BasConst_Request.hpp>
#include <stdair/bom/FareFeaturesKey.hpp>

```

Namespaces

- namespace `stdair`
Handle on the *StdAir* library context.

35.270 FareFeaturesKey.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <ostream>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_DefaultObject.hpp>
00009 #include <stdair/basic/BasConst_Request.hpp>
00010 #include <stdair/bom/FareFeaturesKey.hpp>
00011
00012 namespace stdair {
00013
00014     // //////////////////////////////////////
00015     FareFeaturesKey::FareFeaturesKey()
00016     : _tripType (TRIP_TYPE_ONE_WAY),
00017       _advancePurchase (NO_ADVANCE_PURCHASE),
00018       _saturdayStay (SATURDAY_STAY),
00019       _changeFees (CHANGE_FEES),
00020       _nonRefundable (NON_REFUNDABLE),
00021       _minimumStay (NO_STAY_DURATION) {
00022     assert (false);

```

```

00023     }
00024
00025     // //////////////////////////////////////
00026     FareFeaturesKey::FareFeaturesKey (const TripType_T& iTripType,
00027                                       const DayDuration_T& iAdvancePurchase,
00028                                       const SaturdayStay_T& iSaturdayStay,
00029                                       const ChangeFees_T& iChangeFees,
00030                                       const NonRefundable_T& iNonRefundable,
00031                                       const DayDuration_T& iMinimumStay)
00032     : _tripType (iTripType), _advancePurchase (iAdvancePurchase),
00033       _saturdayStay (iSaturdayStay), _changeFees (iChangeFees),
00034       _nonRefundable (iNonRefundable), _minimumStay (iMinimumStay) {
00035     }
00036
00037     // //////////////////////////////////////
00038     FareFeaturesKey::FareFeaturesKey (const FareFeaturesKey& iKey)
00039     : _tripType (iKey.getTripType()),
00040       _advancePurchase (iKey.getAdvancePurchase()),
00041       _saturdayStay (iKey.getSaturdayStay()),
00042       _changeFees (iKey.getChangeFees()),
00043       _nonRefundable (iKey.getRefundableOption()),
00044       _minimumStay (iKey.getMinimumStay()) {
00045     }
00046
00047     // //////////////////////////////////////
00048     FareFeaturesKey::~FareFeaturesKey() {
00049     }
00050
00051     // //////////////////////////////////////
00052     void FareFeaturesKey::toStream (std::ostream& ioOut) const {
00053         ioOut << "FareFeaturesKey: " << toString() << std::endl;
00054     }
00055
00056     // //////////////////////////////////////
00057     void FareFeaturesKey::fromStream (std::istream& ioIn) {
00058     }
00059
00060     // //////////////////////////////////////
00061     const std::string FareFeaturesKey::toString() const {
00062         std::ostringstream oStr;
00063         oStr << _tripType << " -- " << _advancePurchase << "-"
00064             << _saturdayStay << "-" << _changeFees << "-"
00065             << _nonRefundable << "-" << _minimumStay;
00066         return oStr.str();
00067     }
00068
00069 }

```

35.271 stdair/bom/FareFeaturesKey.hpp File Reference

```

#include <stdair/bom/KeyAbstract.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/stdair_inventory_types.hpp>

```

Classes

- struct `stdair::FareFeaturesKey`

Key of date-period.

Namespaces

- namespace `stdair`

Handle on the StdAir library context.

35.272 FareFeaturesKey.hpp

```

00001 #ifndef __STDAIR_BOM_FAREFEATURESKEY_HPP
00002 #define __STDAIR_BOM_FAREFEATURESKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/bom/KeyAbstract.hpp>
00009 #include <stdair/stdair_date_time_types.hpp>
00010 #include <stdair/stdair_demand_types.hpp>
00011 #include <stdair/stdair_inventory_types.hpp>
00012
00013 namespace stdair {
00014
00015     struct FareFeaturesKey : public KeyAbstract {
00016     public:
00017         // ////////////////////////////////// Construction //////////////////////////////////
00018         FareFeaturesKey (const TripType_T&, const DayDuration_T&,
00019                         const SaturdayStay_T&, const ChangeFees_T&,
00020                         const NonRefundable_T&, const DayDuration_T&);
00021         FareFeaturesKey (const FareFeaturesKey&);
00022         ~FareFeaturesKey ();
00023     private:
00024         FareFeaturesKey ();
00025
00026     public:
00027         // ////////////////////////////////// Getters //////////////////////////////////
00028         const TripType_T& getTripType() const {
00029             return _tripType;
00030         }
00031
00032         const DayDuration_T& getAdvancePurchase() const {
00033             return _advancePurchase;
00034         }
00035
00036         const SaturdayStay_T& getSaturdayStay() const {
00037             return _saturdayStay;
00038         }
00039
00040         const ChangeFees_T& getChangeFees() const {
00041             return _changeFees;
00042         }
00043
00044         const NonRefundable_T& getRefundableOption() const {

```

```

00068         return _nonRefundable;
00069     }
00070
00074     const DayDuration_T& getMinimumStay() const {
00075         return _minimumStay;
00076     }
00077
00078
00079 public:
00080     // //////////// Display support methods ////////////
00086     void toStream (std::ostream& ioOut) const;
00087
00093     void fromStream (std::istream& ioIn);
00094
00100     const std::string toString() const;
00101
00102
00103 private:
00104     // ////////////////////////////////// Attributes //////////////////////////////////
00108     TripType_T _tripType;
00109
00113     DayDuration_T _advancePurchase;
00114
00118     SaturdayStay_T _saturdayStay;
00119
00123     ChangeFees_T _changeFees;
00124
00128     NonRefundable_T _nonRefundable;
00129
00133     DayDuration_T _minimumStay;
00134 };
00135
00136 }
00137 #endif // __STDAIR_BOM_FAREFEATURESKEY_HPP

```

35.273 stdair/bom/FareFeaturesTypes.hpp File Reference

```

#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>

```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

Typedefs

- typedef std::list< FareFeatures * > [stdair::FareFeaturesList_T](#)
- typedef std::map< const MapKey_T, FareFeatures * > [stdair::FareFeaturesMap_T](#)
- typedef std::pair< MapKey_T, FareFeatures * > [stdair::FareFeaturesWithKey_T](#)
- typedef std::list< FareFeaturesWithKey_T > [stdair::FareFeaturesDetailedList_T](#)

35.274 FareFeaturesTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_FAREFEATURESTYPES_HPP
00003 #define __STDAIR_BOM_FAREFEATURESTYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // STDAIR
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016     // Forward declarations.
00017     class FareFeatures;
00018
00019     typedef std::list<FareFeatures*> FareFeaturesList_T;
00020
00021     typedef std::map<const MapKey_T, FareFeatures*> FareFeaturesMap_T;
00022
00023     typedef std::pair<MapKey_T, FareFeatures*> FareFeaturesWithKey_T;
00024     typedef std::list<FareFeaturesWithKey_T> FareFeaturesDetailedList_T;
00025 }
00026
00027 #endif // __STDAIR_BOM_FAREFEATURESTYPES_HPP
00028
00029
00030

```

35.275 stdair/bom/FareOptionStruct.cpp File Reference

```

#include <cassert>

#include <sstream>

#include <stdair/basic/BasConst_BookingClass.hpp>

#include <stdair/bom/FareOptionStruct.hpp>

```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

35.276 FareOptionStruct.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_BookingClass.hpp>
00009 #include <stdair/bom/FareOptionStruct.hpp>

```

```

00010
00011 namespace stdair {
00012
00013 // //////////////////////////////////////
00014 FareOptionStruct::FareOptionStruct ()
00015 : _fare (DEFAULT_FARE_VALUE), _avl (DEFAULT_AVAILABILITY) {
00016 }
00017
00018 // //////////////////////////////////////
00019 FareOptionStruct::FareOptionStruct (const FareOptionStruct& iFO)
00020 : _classPath (iFO._classPath),
00021   _fare (iFO._fare), _avl (iFO._avl), _changeFee (iFO._changeFee),
00022   _nonRefundable (iFO._nonRefundable), _saturdayStay (iFO._saturdayStay) {
00023 }
00024
00025 // //////////////////////////////////////
00026 FareOptionStruct::FareOptionStruct (const std::string& iClassPath,
00027                                     const Fare_T& iFare,
00028                                     const ChangeFees_T& iChangeFee,
00029                                     const NonRefundable_T& iNonRefundable,
00030                                     const SaturdayStay_T& iSaturdayNightStay)
00031 : _fare (iFare), _avl (DEFAULT_AVAILABILITY),
00032   _changeFee (iChangeFee), _nonRefundable (iNonRefundable),
00033   _saturdayStay (iSaturdayNightStay) {
00034   _classPath.push_back (iClassPath);
00035 }
00036
00037 // //////////////////////////////////////
00038 FareOptionStruct::~FareOptionStruct () {
00039 }
00040
00041 // //////////////////////////////////////
00042 void FareOptionStruct::toStream (std::ostream& ioOut) const {
00043   ioOut << describe();
00044 }
00045
00046 // //////////////////////////////////////
00047 void FareOptionStruct::fromStream (std::istream& ioIn) {
00048 }
00049
00050 // //////////////////////////////////////
00051 const std::string FareOptionStruct::describe() const {
00052   std::ostringstream oStr;
00053
00054   oStr << "Class path: ";
00055   unsigned short idx = 0;
00056   for (ClassList_StringList_T::const_iterator itClassPath =
00057         _classPath.begin(); itClassPath != _classPath.end();
00058        ++itClassPath, ++idx) {
00059     if (idx != 0) {
00060       oStr << "-";
00061     }
00062     const std::string& lClassPath = *itClassPath;
00063     oStr << lClassPath;
00064   }
00065
00066   oStr << "; " << _fare << " EUR";
00067   oStr << "; conditions: " << _changeFee << " " << _nonRefundable
00068     << " " << _saturdayStay;
00069   return oStr.str();
00070 }
00071

```

```

00072 // //////////////////////////////////////
00073 const std::string FareOptionStruct::display() const {
00074     std::ostringstream ostr;
00075
00076     unsigned short idx = 0;
00077     for (ClassList_StringList_T::const_iterator itClassPath =
00078         _classPath.begin(); itClassPath != _classPath.end();
00079         ++itClassPath, ++idx) {
00080         if (idx != 0) {
00081             ostr << "-";
00082         }
00083         const std::string& lClassPath = *itClassPath;
00084         ostr << lClassPath;
00085     }
00086
00087     ostr << ", " << _fare << ", " << _changeFee << " " << _nonRefundable
00088         << " " << _saturdayStay;
00089     return ostr.str();
00090 }
00091
00092 // //////////////////////////////////////
00093 void FareOptionStruct::addClassList (const std::string iClassCodeList) {
00094     _classPath.push_back (iClassCodeList);
00095 }
00096
00097 // //////////////////////////////////////
00098 void FareOptionStruct::emptyClassList () {
00099     _classPath.clear();
00100 }
00101
00102 }

```

35.277 stdair/bom/FareOptionStruct.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/stdair_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/BookingClassTypes.hpp>

```

Classes

- struct [stdair::FareOptionStruct](#)
Structure holding the elements of a fare option.

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.278 FareOptionStruct.hpp

```

00001 #ifndef __STDAIR_BOM_FAREOPTIONSTRUCT_HPP
00002 #define __STDAIR_BOM_FAREOPTIONSTRUCT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_types.hpp>
00012 #include <stdair/basic/StructAbstract.hpp>
00013 #include <stdair/bom/BookingClassTypes.hpp>
00014
00015 namespace stdair {
00016
00020     struct FareOptionStruct : public StructAbstract {
00021     public:
00022         // //////////////////////////////////
00024         const ClassList_StringList_T& getClassPath() const {
00025             return _classPath;
00026         }
00027
00029         const Fare_T& getFare() const {
00030             return _fare;
00031         }
00032
00034         const Availability_T& getAvailability() const {
00035             return _avl;
00036         }
00037
00039         const ChangeFees_T getChangeFees() const {
00040             return _changeFee;
00041         }
00042
00044         const NonRefundable_T getNonRefundable() const {
00045             return _nonRefundable;
00046         }
00047
00049         const SaturdayStay_T getSaturdayStay() const {
00050             return _saturdayStay;
00051         }
00052
00053     public:
00055         // //////////////////////////////////
00057         void addClassList (const std::string);
00058
00060         void emptyClassList ();
00061
00063         void setFare (const Fare_T& iFare) {
00064             _fare = iFare;
00065         }
00066
00068         void setAvailability (const Availability_T& iAvl) {
00069             _avl = iAvl;
00070         }
00071
00073         void setChangeFees (const ChangeFees_T iRes) {
00074             _changeFee = iRes;

```

```

00075     }
00076
00077     void setNonRefundable (const NonRefundable_T iRes) {
00078         _nonRefundable = iRes;
00079     }
00080
00081     void setSaturdayStay (const SaturdayStay_T iRes) {
00082         _saturdayStay = iRes;
00083     }
00084
00085 public:
00086     // //////////// Display support method ////////////
00087     void toStream (std::ostream& ioOut) const;
00088
00089     void fromStream (std::istream& ioIn);
00090
00091     const std::string describe() const;
00092
00093     const std::string display() const;
00094
00095 public:
00096     // //////////// Constructors & Destructor ////////////
00097     FareOptionStruct ();
00098
00099     FareOptionStruct (const std::string& iClassPath,
00100                     const Fare_T&, const ChangeFees_T&,
00101                     const NonRefundable_T&, const SaturdayStay_T&);
00102
00103     FareOptionStruct (const FareOptionStruct&);
00104
00105     ~FareOptionStruct ();
00106
00107 private:
00108     // //////////// Attributes ////////////
00109     ClassList_StringList_T _classPath;
00110
00111     Fare_T _fare;
00112
00113     Availability_T _avl;
00114
00115     ChangeFees_T _changeFee;
00116
00117     NonRefundable_T _nonRefundable;
00118
00119     SaturdayStay_T _saturdayStay;
00120 };
00121
00122 #endif // __STDAIR_BOM_FAREOPTIONSTRUCT_HPP

```

35.279 stdair/bom/FareOptionTypes.hpp File Reference

```

#include <list>

#include <map>

#include <stdair/stdair_types.hpp>

```

```
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

Typedefs

- typedef `std::list< FareOptionStruct >` `stdair::FareOptionList_T`

35.280 FareOptionTypes.hpp

```
00001 ///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00002 #ifndef __STDAIR_BOM_FAREOPTIONTYPES_HPP
00003 #define __STDAIR_BOM_FAREOPTIONTYPES_HPP
00004
00005 ///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00006 // Import section
00007 ///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00008 // STL
00009 #include <list>
00010 #include <map>
00011 // STDAIR
00012 #include <stdair/stdair_types.hpp>
00013 #include <stdair/bom/key_types.hpp>
00014
00015 namespace stdair {
00016
00017     // Forward declarations.
00018     struct FareOptionStruct;
00019
00021     typedef std::list<FareOptionStruct> FareOptionList_T;
00022
00023 }
00024 #endif // __STDAIR_BOM_FAREOPTIONTYPES_HPP
00025
```

35.281 stdair/bom/FlightDate.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/SegmentDate.hpp>
```

Namespaces

- namespace `stdair`

Handle on the StdAir library context.

35.282 FlightDate.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_Inventory.hpp>
00009 #include <stdair/bom/BomManager.hpp>
00010 #include <stdair/bom/Inventory.hpp>
00011 #include <stdair/bom/FlightDate.hpp>
00012 #include <stdair/bom/LegDate.hpp>
00013 #include <stdair/bom/SegmentDate.hpp>
00014
00015 namespace stdair {
00016
00017     // //////////////////////////////////////
00018     FlightDate::FlightDate()
00019         : _key (DEFAULT_FLIGHT_NUMBER, DEFAULT_DEPARTURE_DATE), _parent (NULL) {
00020         // That constructor is used by the serialisation process
00021     }
00022
00023     // //////////////////////////////////////
00024     FlightDate::FlightDate (const FlightDate&)
00025         : _key (DEFAULT_FLIGHT_NUMBER, DEFAULT_DEPARTURE_DATE), _parent (NULL) {
00026         assert (false);
00027     }
00028
00029     // //////////////////////////////////////
00030     FlightDate::FlightDate (const Key_T& iKey) : _key (iKey), _parent (NULL) {
00031     }
00032
00033     // //////////////////////////////////////
00034     FlightDate::~FlightDate() {
00035     }
00036
00037     // //////////////////////////////////////
00038     const AirlineCode_T& FlightDate::getAirlineCode() const {
00039         const Inventory* lInventory_ptr =
00040             static_cast<const Inventory*> (getParent());
00041         assert (lInventory_ptr != NULL);
00042         return lInventory_ptr->getAirlineCode();
00043     }
00044
00045     // //////////////////////////////////////
00046     std::string FlightDate::toString() const {
00047         std::ostringstream oStr;
00048         oStr << describeKey();
00049         return oStr.str();
00050     }
00051
00052     // //////////////////////////////////////
00053     LegDate* FlightDate::getLegDate (const std::string& iLegDateKeyStr) const {

```

```

00054     LegDate* oLegDate_ptr =
00055         BomManager::getObjectPtr<LegDate> (*this, iLegDateKeyStr);
00056     return oLegDate_ptr;
00057 }
00058
00059 // //////////////////////////////////////
00060 LegDate* FlightDate::getLegDate (const LegDateKey& iLegDateKey) const {
00061     return getLegDate (iLegDateKey.toString());
00062 }
00063
00064 // //////////////////////////////////////
00065 SegmentDate* FlightDate::
00066 getSegmentDate (const std::string& iSegmentDateKeyStr) const {
00067     SegmentDate* oSegmentDate_ptr =
00068         BomManager::getObjectPtr<SegmentDate> (*this, iSegmentDateKeyStr);
00069     return oSegmentDate_ptr;
00070 }
00071
00072 // //////////////////////////////////////
00073 SegmentDate* FlightDate::
00074 getSegmentDate (const SegmentDateKey& iSegmentDateKey) const {
00075     return getSegmentDate (iSegmentDateKey.toString());
00076 }
00077
00078 }
00079

```

35.283 stdair/bom/FlightDate.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/FlightDateKey.hpp>
#include <stdair/bom/FlightDateTypes.hpp>

```

Classes

- class [stdair::FlightDate](#)
Class representing the actual attributes for an airline flight-date.

Namespaces

- namespace [boost](#)
Forward declarations.
- namespace [boost::serialization](#)
- namespace [stdair](#)
Handle on the StdAir library context.

35.284 FlightDate.hpp

```

00001 #ifndef __STDAIR_BOM_FLIGHTDATE_HPP
00002 #define __STDAIR_BOM_FLIGHTDATE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/bom/BomAbstract.hpp>
00013 #include <stdair/bom/FlightDateKey.hpp>
00014 #include <stdair/bom/FlightDateTypes.hpp>
00015
00016 namespace boost {
00017     namespace serialization {
00018         class access;
00019     }
00020 }
00021
00022 namespace stdair {
00023     struct LegDateKey;
00024     class LegDate;
00025     struct SegmentDateKey;
00026     class SegmentDate;
00027
00028     class FlightDate : public BomAbstract {
00029     public:
00030         template <typename BOM> friend class FacBom;
00031         friend class FacBomManager;
00032         friend class boost::serialization::access;
00033
00034         // ////////////////////////////////// Type definitions //////////////////////////////////
00035         typedef FlightDateKey Key_T;
00036
00037     public:
00038         // ////////////////////////////////// Getters //////////////////////////////////
00039         const Key_T& getKey() const {
00040             return _key;
00041         }
00042
00043         BomAbstract* const getParent() const {
00044             return _parent;
00045         }
00046
00047         const FlightNumber_T& getFlightNumber() const {
00048             return _key.getFlightNumber();
00049         }
00050
00051         const Date_T& getDepartureDate() const {
00052             return _key.getDepartureDate();
00053         }
00054
00055         const AirlineCode_T& getAirlineCode() const;
00056
00057         const HolderMap_T& getHolderMap() const {
00058             return _holderMap;
00059         }
00060     };
00061 }

```

```

00084     }
00085
00096     LegDate* getLegDate (const std::string& iLegDateKeyStr) const;
00097
00108     LegDate* getLegDate (const LegDateKey&) const;
00109
00120     SegmentDate* getSegmentDate (const std::string& iSegmentDateKeyStr) const;
00121
00132     SegmentDate* getSegmentDate (const SegmentDateKey&) const;
00133
00134 public:
00135     // //////////// Display support methods ////////////
00141     void toStream (std::ostream& ioOut) const {
00142         ioOut << toString();
00143     }
00144
00150     void fromStream (std::istream& ioIn) {
00151     }
00152
00156     std::string toString() const;
00157
00161     const std::string describeKey() const {
00162         return _key.toString();
00163     }
00164
00165
00166 public:
00167     // //////////// (Boost) Serialisation support methods ////////////
00171     template<class Archive>
00172     void serialize (Archive& ar, const unsigned int iFileVersion);
00173
00174 private:
00182     void serialisationImplementationExport() const;
00183     void serialisationImplementationImport();
00184
00185
00186 protected:
00187     // //////////// Constructors and destructors ////////////
00191     FlightDate (const Key_T&);
00192
00196     virtual ~FlightDate();
00197
00198 private:
00202     FlightDate();
00203
00207     FlightDate (const FlightDate&);
00208
00209
00210 protected:
00211     // //////////// Attributes ////////////
00215     Key_T _key;
00216
00220     BomAbstract* _parent;
00221
00225     HolderMap_T _holderMap;
00226 };
00227
00228 }
00229 #endif // __STDAIR_BOM_FLIGHTDATE_HPP
00230

```

35.285 stdair/bom/FlightDateKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/date_time/gregorian/formatters.hpp>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/FlightDateKey.hpp>
```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

Functions

- template void [stdair::FlightDateKey::serialize< ba::text_oarchive >](#) (ba::text_oarchive &, unsigned int)
- template void [stdair::FlightDateKey::serialize< ba::text_iarchive >](#) (ba::text_iarchive &, unsigned int)

35.286 FlightDateKey.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost Date-Time
00008 #include <boost/date_time/gregorian/formatters.hpp>
00009 // Boost.Serialization
00010 #include <boost/archive/text_iarchive.hpp>
00011 #include <boost/archive/text_oarchive.hpp>
00012 #include <boost/serialization/access.hpp>
00013 // StdAir
00014 #include <stdair/basic/BasConst_Inventory.hpp>
00015 #include <stdair/basic/BasConst_BomDisplay.hpp>
00016 #include <stdair/bom/FlightDateKey.hpp>
00017
00018 namespace stdair {
00019
00020 // //////////////////////////////////////
00021 FlightDateKey::FlightDateKey()
00022     : _flightNumber (DEFAULT_FLIGHT_NUMBER),
```

```

00023     _departureDate (DEFAULT_DEPARTURE_DATE) {
00024         assert (false);
00025     }
00026
00027 ///////////////////////////////////////////////////////////////////
00028 FlightDateKey::FlightDateKey (const FlightNumber_T& iFlightNumber,
00029                               const Date_T& iFlightDate)
00030     : _flightNumber (iFlightNumber), _departureDate (iFlightDate) {
00031     }
00032
00033 ///////////////////////////////////////////////////////////////////
00034 FlightDateKey::FlightDateKey (const FlightDateKey& iKey)
00035     : _flightNumber (iKey._flightNumber), _departureDate (iKey._departureDate) {
00036     }
00037
00038 ///////////////////////////////////////////////////////////////////
00039 FlightDateKey::~FlightDateKey() {
00040 }
00041
00042 ///////////////////////////////////////////////////////////////////
00043 void FlightDateKey::toStream (std::ostream& ioOut) const {
00044     ioOut << "FlightDateKey: " << toString();
00045 }
00046
00047 ///////////////////////////////////////////////////////////////////
00048 void FlightDateKey::fromStream (std::istream& ioIn) {
00049 }
00050
00051 ///////////////////////////////////////////////////////////////////
00052 const std::string FlightDateKey::toString() const {
00053     std::ostringstream oStr;
00054     const std::string& lDepartureDateStr =
00055         boost::gregorian::to_simple_string (_departureDate);
00056     oStr << _flightNumber
00057         << DEFAULT_KEY_SUB_FLD_DELIMITER << " " << lDepartureDateStr;
00058     return oStr.str();
00059 }
00060
00061 ///////////////////////////////////////////////////////////////////
00062 void FlightDateKey::serialisationImplementationExport() const {
00063     std::ostringstream oStr;
00064     boost::archive::text_oarchive oa (oStr);
00065     oa << *this;
00066 }
00067
00068 ///////////////////////////////////////////////////////////////////
00069 void FlightDateKey::serialisationImplementationImport() {
00070     std::istringstream iStr;
00071     boost::archive::text_iarchive ia (iStr);
00072     ia >> *this;
00073 }
00074
00075 ///////////////////////////////////////////////////////////////////
00076 template<class Archive>
00077 void FlightDateKey::serialize (Archive& ioArchive,
00078                               const unsigned int iFileVersion) {
00079     std::string lDepartureDateStr =
00080         boost::gregorian::to_simple_string (_departureDate);
00081     ioArchive & _flightNumber & lDepartureDateStr;
00082 }
00083
00084 ///////////////////////////////////////////////////////////////////

```

```

00089 // Explicit template instantiation
00090 namespace ba = boost::archive;
00091 template void FlightDateKey::serialize<ba::text_oarchive> (ba::text_oarchive&,
00092                                                         unsigned int);
00093 template void FlightDateKey::serialize<ba::text_iarchive> (ba::text_iarchive&,
00094                                                         unsigned int);
00095 // //////////////////////////////////////
00096
00097 }

```

35.287 stdair/bom/FlightDateKey.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>

```

Classes

- struct [stdair::FlightDateKey](#)
Key of a given flight-date, made of a flight number and a departure date.

Namespaces

- namespace [boost](#)
Forward declarations.
- namespace [boost::serialization](#)
- namespace [stdair](#)
Handle on the StdAir library context.

35.288 FlightDateKey.hpp

```

00001 #ifndef __STDAIR_BOM_FLIGHTDATEKEY_HPP
00002 #define __STDAIR_BOM_FLIGHTDATEKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_basic_types.hpp>
00012 #include <stdair/stdair_date_time_types.hpp>
00013 #include <stdair/bom/KeyAbstract.hpp>
00014
00016 namespace boost {
00017     namespace serialization {

```

```

00018     class access;
00019 }
00020 }
00021
00022 namespace stdair {
00023
00028     struct FlightDateKey : public KeyAbstract {
00029         friend class boost::serialization::access;
00030
00031         // ////////// Constructors and destructors //////////
00032     private:
00036         FlightDateKey();
00037
00038     public:
00042         FlightDateKey (const FlightNumber_T&, const Date_T&);
00043
00047         FlightDateKey (const FlightDateKey&);
00048
00052         ~FlightDateKey();
00053
00054
00055     public:
00056         // ////////// Getters //////////
00058         const FlightNumber_T& getFlightNumber() const {
00059             return _flightNumber;
00060         }
00061
00063         const Date_T& getDepartureDate() const {
00064             return _departureDate;
00065         }
00066
00067
00068     public:
00069         // ////////// Display support methods //////////
00075         void toStream (std::ostream& ioOut) const;
00076
00082         void fromStream (std::istream& ioIn);
00083
00093         const std::string toString() const;
00094
00095
00096     public:
00097         // ////////// (Boost) Serialisation support methods //////////
00101         template<class Archive>
00102         void serialize (Archive& ar, const unsigned int iFileVersion);
00103
00104     private:
00109         void serialisationImplementationExport() const;
00110         void serialisationImplementationImport();
00111
00112
00113     private:
00114         // ////////// Attributes //////////
00118         FlightNumber_T _flightNumber;
00119
00123         Date_T _departureDate;
00124     };
00125 }
00126 }
00127 #endif // __STDAIR_BOM_FLIGHTDATEKEY_HPP

```

35.289 stdair/bom/FlightDateTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

Typedefs

- typedef std::list< FlightDate * > [stdair::FlightDateList_T](#)
- typedef std::map< const MapKey_T, FlightDate * > [stdair::FlightDateMap_T](#)

35.290 FlightDateTypes.hpp

```
00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_FLIGHTDATETYPES_HPP
00003 #define __STDAIR_BOM_FLIGHTDATETYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // StdAir
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016     // Forward declarations
00017     class FlightDate;
00018
00019     // ////////////////////////////////// Type definitions //////////////////////////////////
00021     typedef std::list<FlightDate*> FlightDateList_T;
00022
00024     typedef std::map<const MapKey_T, FlightDate*> FlightDateMap_T;
00025
00026 }
00027 #endif // __STDAIR_BOM_FLIGHTDATETYPES_HPP
00028
```

35.291 stdair/bom/FlightPeriod.cpp File Reference

```
#include <cassert>
#include <stdair/bom/FlightPeriod.hpp>
```

Namespaces

- namespace `stdair`

Handle on the StdAir library context.

35.292 FlightPeriod.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // STDAIR
00007 #include <stdair/bom/FlightPeriod.hpp>
00008
00009 namespace stdair {
00010
00011 // //////////////////////////////////////
00012 FlightPeriod::FlightPeriod (const Key_T& iKey)
00013     : _key (iKey), _parent (NULL) {
00014 }
00015
00016 // //////////////////////////////////////
00017 FlightPeriod::~FlightPeriod () {
00018 }
00019
00020 // //////////////////////////////////////
00021 std::string FlightPeriod::toString() const {
00022     std::ostringstream ostr;
00023     ostr << describeKey();
00024     return ostr.str();
00025 }
00026
00027 }
00028

```

35.293 stdair/bom/FlightPeriod.hpp File Reference

```

#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/FlightPeriodKey.hpp>
#include <stdair/bom/FlightPeriodTypes.hpp>

```

Classes

- class `stdair::FlightPeriod`

Namespaces

- namespace `stdair`

Handle on the StdAir library context.

35.294 FlightPeriod.hpp

```

00001 #ifndef __STDAIR_BOM_FLIGHTPERIOD_HPP
00002 #define __STDAIR_BOM_FLIGHTPERIOD_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STDAIR
00008 #include <stdair/bom/BomAbstract.hpp>
00009 #include <stdair/bom/FlightPeriodKey.hpp>
00010 #include <stdair/bom/FlightPeriodTypes.hpp>
00011
00012 namespace stdair {
00013
00014     class FlightPeriod : public BomAbstract {
00015     public:
00016         template <typename BOM> friend class FacBom;
00017         friend class FacBomManager;
00018
00019         // Type definitions.
00020         typedef FlightPeriodKey Key_T;
00021
00022     public:
00023         // ////////////////////////////////// Getters //////////////////////////////////
00024         const Key_T& getKey () const { return _key; }
00025
00026         BomAbstract* const getParent () const { return _parent; }
00027
00028         const FlightNumber_T& getFlightNumber () const {
00029             return _key.getFlightNumber();
00030         }
00031
00032         const PeriodStruct& getPeriod () const { return _key.getPeriod(); }
00033
00034         const HolderMap_T& getHolderMap () const { return _holderMap; }
00035
00036     public:
00037         // ////////////////////////////////// Display support methods //////////////////////////////////
00038         void toStream (std::ostream& ioOut) const { ioOut << toString(); }
00039
00040         void fromStream (std::istream& ioIn) { }
00041
00042         std::string toString () const;
00043
00044         const std::string describeKey () const { return _key.toString(); }
00045
00046     protected:
00047         FlightPeriod (const Key_T&);
00048         FlightPeriod (const FlightPeriod&);
00049         ~FlightPeriod ();
00050
00051     protected:
00052         // Attributes
00053         Key_T _key;
00054         BomAbstract* _parent;
00055         HolderMap_T _holderMap;
00056     };
00057
00058 }
00059
00060 #endif // __STDAIR_BOM_FLIGHTPERIOD_HPP

```

00076

35.295 stdair/bom/FlightPeriodKey.cpp File Reference

```
#include <stdair/bom/FlightPeriodKey.hpp>
```

Namespaces

- namespace `stdair`

*Handle on the StdAir library context.***35.296 FlightPeriodKey.cpp**

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STDAIR
00005 #include <stdair/bom/FlightPeriodKey.hpp>
00006
00007 namespace stdair {
00008
00009 // //////////////////////////////////////
00010 FlightPeriodKey::FlightPeriodKey (const FlightNumber_T& iFlightNumber,
00011                                   const PeriodStruct& iPeriod)
00012     : _flightNumber (iFlightNumber), _period (iPeriod) {
00013 }
00014
00015 // //////////////////////////////////////
00016 FlightPeriodKey::FlightPeriodKey (const FlightPeriodKey& iKey)
00017     : _flightNumber (iKey._flightNumber), _period (iKey._period) {
00018 }
00019
00020 // //////////////////////////////////////
00021 FlightPeriodKey::~FlightPeriodKey () {
00022 }
00023
00024 // //////////////////////////////////////
00025 void FlightPeriodKey::toStream (std::ostream& ioOut) const {
00026     ioOut << "FlightPeriodKey: " << toString() << std::endl;
00027 }
00028
00029 // //////////////////////////////////////
00030 void FlightPeriodKey::fromStream (std::istream& ioIn) {
00031 }
00032
00033 // //////////////////////////////////////
00034 const std::string FlightPeriodKey::toString() const {
00035     std::ostringstream oStr;
00036     oStr << _flightNumber << ", " << _period.describeShort();
00037     return oStr.str();
00038 }
00039
00040 }

```

35.297 stdair/bom/FlightPeriodKey.hpp File Reference

```
#include <stdair/bom/KeyAbstract.hpp>
#include <stdair/bom/PeriodStruct.hpp>
```

Classes

- struct [stdair::FlightPeriodKey](#)

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.298 FlightPeriodKey.hpp

```
00001 #ifndef __STDAIR_BOM_FLIGHTPERIODKEY_HPP
00002 #define __STDAIR_BOM_FLIGHTPERIODKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STDAIR
00008 #include <stdair/bom/KeyAbstract.hpp>
00009 #include <stdair/bom/PeriodStruct.hpp>
00010
00011 namespace stdair {
00013     struct FlightPeriodKey : public KeyAbstract {
00014
00015     private:
00016         // ////////////////////////////////// Default constructor //////////////////////////////////
00017         FlightPeriodKey ();
00018     public:
00019         // ////////////////////////////////// Construction //////////////////////////////////
00021         FlightPeriodKey (const FlightNumber_T&, const PeriodStruct&);
00022         FlightPeriodKey (const FlightPeriodKey&);
00024         ~FlightPeriodKey ();
00025
00026         // ////////////////////////////////// Getters //////////////////////////////////
00028         const FlightNumber_T& getFlightNumber() const {
00029             return _flightNumber;
00030         }
00031
00033         const PeriodStruct& getPeriod () const {
00034             return _period;
00035         }
00036
00037         // ////////////////////////////////// Display support methods //////////////////////////////////
00040         void toStream (std::ostream& ioOut) const;
00041
00044         void fromStream (std::istream& ioIn);
00045
00051         const std::string toString() const;
00052
00053     private:
```

```

00054     // Attributes
00055     FlightNumber_T _flightNumber;
00056
00057     PeriodStruct _period;
00058
00059 };
00060
00061 };
00062
00063 }
00064 #endif // __STDAIR_BOM_FLIGHTPERIODKEY_HPP

```

35.299 stdair/bom/FlightPeriodTypes.hpp File Reference

```

#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>

```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

Typedefs

- typedef `std::list< FlightPeriod * >` `stdair::FlightPeriodList_T`
- typedef `std::map< const MapKey_T, FlightPeriod * >` `stdair::FlightPeriodMap_T`

35.300 FlightPeriodTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_FLIGHTPERIODTYPES_HPP
00003 #define __STDAIR_BOM_FLIGHTPERIODTYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // StdAir
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016     // Forward declarations.
00017     class FlightPeriod;
00018
00019     typedef std::list<FlightPeriod*> FlightPeriodList_T;
00020
00021     typedef std::map<const MapKey_T, FlightPeriod*> FlightPeriodMap_T;
00022 }
00023 #endif // __STDAIR_BOM_FLIGHTPERIODTYPES_HPP
00024
00025
00026

```

35.301 stdair/bom/GuillotineBlock.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <boost/multi_array.hpp>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/GuillotineBlock.hpp>

```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

35.302 GuillotineBlock.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost
00008 #include <boost/multi_array.hpp>
00009 // Boost.Serialization
00010 #include <boost/archive/text_iarchive.hpp>
00011 #include <boost/archive/text_oarchive.hpp>
00012 #include <boost/serialization/access.hpp>
00013 // StdAir
00014 #include <stdair/basic/BasConst_Inventory.hpp>
00015 #include <stdair/bom/BomManager.hpp>
00016 #include <stdair/bom/GuillotineBlock.hpp>
00017
00018 namespace stdair {
00019
00020 // //////////////////////////////////////
00021 GuillotineBlock::GuillotineBlock()
00022     : _key (DEFAULT_GUILLOTINE_NUMBER), _parent (NULL) {
00023     assert (false);
00024 }
00025
00026 // //////////////////////////////////////
00027 GuillotineBlock::GuillotineBlock (const GuillotineBlock&)
00028     : _key (DEFAULT_GUILLOTINE_NUMBER), _parent (NULL) {
00029     assert (false);
00030 }
00031
00032 // //////////////////////////////////////

```

```

00033 GuillotineBlock::
00034 GuillotineBlock (const Key_T& iKey) : _key (iKey), _parent (NULL) {
00035 }
00036
00037 // //////////////////////////////////////
00038 GuillotineBlock::~GuillotineBlock() {
00039 }
00040
00041 // //////////////////////////////////////
00042 std::string GuillotineBlock::toString() const {
00043     std::ostringstream ostr;
00044     ostr << describeKey();
00045     return ostr.str();
00046 }
00047
00048 // //////////////////////////////////////
00049 void GuillotineBlock::
00050 initSnapshotBlocks (const SegmentCabinIndexMap_T& iSegmentCabinIndexMap,
00051                    const ValueTypeIndexMap_T& iValueTypeIndexMap) {
00052     _segmentCabinIndexMap = iSegmentCabinIndexMap;
00053     _valueTypesIndexMap = iValueTypeIndexMap;
00054
00055     unsigned int lNumberOfSegmentCabins = _segmentCabinIndexMap.size();
00056     unsigned int lNumberOfValueTypes = _valueTypesIndexMap.size();
00057
00058     // Initialise the snapshot blocks
00059     // Normally, the block includes snapshots from DTD MAX to DTD 0, thus
00060     // DEFAULT_MAX_DTD + 1 values. However, we would like to add the day
00061     // before DTD MAX (this value will be initialised to zero).
00062     _bookingSnapshotBlock.
00063         resize (boost::extents[lNumberOfSegmentCabins*lNumberOfValueTypes]
00064                [DEFAULT_MAX_DTD + 2]);
00065     _cancellationSnapshotBlock.
00066         resize (boost::extents[lNumberOfSegmentCabins*lNumberOfValueTypes]
00067                [DEFAULT_MAX_DTD + 2]);
00068     _productAndPriceOrientedBookingSnapshotBlock.
00069         resize (boost::extents[lNumberOfSegmentCabins*lNumberOfValueTypes]
00070                [DEFAULT_MAX_DTD + 2]);
00071     _availabilitySnapshotBlock.
00072         resize (boost::extents[lNumberOfSegmentCabins*lNumberOfValueTypes]
00073                [DEFAULT_MAX_DTD + 2]);
00074
00075 }
00076
00077 // //////////////////////////////////////
00078 const BlockIndex_T& GuillotineBlock::
00079 getBlockIndex (const MapKey_T& iKey) const {
00080     ValueTypeIndexMap_T::const_iterator itVTIdx =
00081         _valueTypesIndexMap.find (iKey);
00082     assert (itVTIdx != _valueTypesIndexMap.end());
00083     return itVTIdx->second;
00084 }
00085
00086 // //////////////////////////////////////
00087 const BlockNumber_T& GuillotineBlock::
00088 getBlockNumber (const SegmentCabin& iSegmentCabin) const {
00089     SegmentCabinIndexMap_T::const_iterator itSCIdx =
00090         _segmentCabinIndexMap.find (&iSegmentCabin);
00091     assert (itSCIdx != _segmentCabinIndexMap.end());
00092     return itSCIdx->second;
00093 }
00094

```

```

00095 // //////////////////////////////////////
00096 ConstSegmentCabinDTDSnapshotView_T GuillotineBlock::
00097 getConstSegmentCabinDTDBookingSnapshotView (const BlockNumber_T iSCIdxBegin,
00098                                             const BlockNumber_T iSCIdxEnd,
00099                                             const DTD_T iDTD) const {
00100     const unsigned int lNbOfValueTypes = _valueTypesIndexMap.size();
00101     const unsigned int lValueIdxBegin = iSCIdxBegin * lNbOfValueTypes;
00102     const unsigned int lValueIdxEnd = (iSCIdxEnd + 1) * lNbOfValueTypes;
00103
00104     return _bookingSnapshotBlock [ boost::indices[SnapshotBlockRange_T(lValueIdxB
egin, lValueIdxEnd)][iDTD] ];
00105 }
00106
00107 // //////////////////////////////////////
00108 ConstSegmentCabinDTDRangeSnapshotView_T GuillotineBlock::
00109 getConstSegmentCabinDTDRangeBookingSnapshotView
00110 (const BlockNumber_T iSCIdxBegin, const BlockNumber_T iSCIdxEnd,
00111  const DTD_T iDTDBegin, const DTD_T iDTDEnd) const {
00112     const unsigned int lNbOfValueTypes = _valueTypesIndexMap.size();
00113     const unsigned int lValueIdxBegin = iSCIdxBegin * lNbOfValueTypes;
00114     const unsigned int lValueIdxEnd = (iSCIdxEnd + 1) * lNbOfValueTypes;
00115
00116     return _bookingSnapshotBlock [ boost::indices[SnapshotBlockRange_T(lValueIdxB
egin, lValueIdxEnd)][SnapshotBlockRange_T(iDTDBegin, iDTDEnd + 1)] ];
00117 }
00118
00119 // //////////////////////////////////////
00120 SegmentCabinDTDSnapshotView_T GuillotineBlock::
00121 getSegmentCabinDTDBookingSnapshotView (const BlockNumber_T iSCIdxBegin,
00122                                         const BlockNumber_T iSCIdxEnd,
00123                                         const DTD_T iDTD) {
00124     const unsigned int lNbOfValueTypes = _valueTypesIndexMap.size();
00125     const unsigned int lValueIdxBegin = iSCIdxBegin * lNbOfValueTypes;
00126     const unsigned int lValueIdxEnd = (iSCIdxEnd + 1) * lNbOfValueTypes;
00127
00128     return _bookingSnapshotBlock [ boost::indices[SnapshotBlockRange_T(lValueIdxB
egin, lValueIdxEnd)][iDTD] ];
00129 }
00130
00131 // //////////////////////////////////////
00132 SegmentCabinDTDRangeSnapshotView_T GuillotineBlock::
00133 getSegmentCabinDTDRangeBookingSnapshotView (const BlockNumber_T iSCIdxBegin,
00134                                             const BlockNumber_T iSCIdxEnd,
00135                                             const DTD_T iDTDBegin,
00136                                             const DTD_T iDTDEnd) {
00137     const unsigned int lNbOfValueTypes = _valueTypesIndexMap.size();
00138     const unsigned int lValueIdxBegin = iSCIdxBegin * lNbOfValueTypes;
00139     const unsigned int lValueIdxEnd = (iSCIdxEnd + 1) * lNbOfValueTypes;
00140
00141     return _bookingSnapshotBlock [ boost::indices[SnapshotBlockRange_T(lValueIdxB
egin, lValueIdxEnd)][SnapshotBlockRange_T(iDTDBegin, iDTDEnd + 1)] ];
00142 }
00143
00144 // //////////////////////////////////////
00145 ConstSegmentCabinDTDSnapshotView_T GuillotineBlock::
00146 getConstSegmentCabinDTDCancellationSnapshotView (const BlockNumber_T iSCIdxBegi
n,
00147                                                    const BlockNumber_T iSCIdxEnd,
00148                                                    const DTD_T iDTD) const {
00149     const unsigned int lNbOfValueTypes = _valueTypesIndexMap.size();
00150     const unsigned int lValueIdxBegin = iSCIdxBegin * lNbOfValueTypes;
00151     const unsigned int lValueIdxEnd = (iSCIdxEnd + 1) * lNbOfValueTypes;

```

```

00152
00153     return _cancellationSnapshotBlock [ boost::indices[SnapshotBlockRange_T(lValu
00154 eIdxBegin, lValueIdxEnd)][iDTD] ];
00155 }
00156 // //////////////////////////////////////
00157 ConstSegmentCabinDTRangeSnapshotView_T GuillotineBlock::
00158 getConstSegmentCabinDTRangeCancellationSnapshotView
00159 (const BlockNumber_T iSCIdxBegin, const BlockNumber_T iSCIdxEnd,
00160  const DTD_T iDTDBegin, const DTD_T iDTDEnd) const {
00161     const unsigned int lNbOfValueTypes = _valueTypesIndexMap.size();
00162     const unsigned int lValueIdxBegin = iSCIdxBegin * lNbOfValueTypes;
00163     const unsigned int lValueIdxEnd = (iSCIdxEnd + 1) * lNbOfValueTypes;
00164
00165     return _cancellationSnapshotBlock [ boost::indices[SnapshotBlockRange_T(lValu
00166 eIdxBegin, lValueIdxEnd)][SnapshotBlockRange_T(iDTDBegin, iDTDEnd + 1)] ];
00167 }
00168 // //////////////////////////////////////
00169 SegmentCabinDTRangeSnapshotView_T GuillotineBlock::
00170 getSegmentCabinDTRangeCancellationSnapshotView (const BlockNumber_T iSCIdxBegin,
00171                                                  const BlockNumber_T iSCIdxEnd,
00172                                                  const DTD_T iDTD) {
00173     const unsigned int lNbOfValueTypes = _valueTypesIndexMap.size();
00174     const unsigned int lValueIdxBegin = iSCIdxBegin * lNbOfValueTypes;
00175     const unsigned int lValueIdxEnd = (iSCIdxEnd + 1) * lNbOfValueTypes;
00176
00177     return _cancellationSnapshotBlock [ boost::indices[SnapshotBlockRange_T(lValu
00178 eIdxBegin, lValueIdxEnd)][iDTD] ];
00179 }
00180 // //////////////////////////////////////
00181 SegmentCabinDTRangeSnapshotView_T GuillotineBlock::
00182 getSegmentCabinDTRangeCancellationSnapshotView(const BlockNumber_T iSCIdxBegin
00183 ,
00184                                                  const BlockNumber_T iSCIdxEnd,
00185                                                  const DTD_T iDTDBegin,
00186                                                  const DTD_T iDTDEnd) {
00187     const unsigned int lNbOfValueTypes = _valueTypesIndexMap.size();
00188     const unsigned int lValueIdxBegin = iSCIdxBegin * lNbOfValueTypes;
00189     const unsigned int lValueIdxEnd = (iSCIdxEnd + 1) * lNbOfValueTypes;
00190
00191     return _cancellationSnapshotBlock [ boost::indices[SnapshotBlockRange_T(lValu
00192 eIdxBegin, lValueIdxEnd)][SnapshotBlockRange_T(iDTDBegin, iDTDEnd + 1)] ];
00193 }
00194 // //////////////////////////////////////
00195 ConstSegmentCabinDTRangeSnapshotView_T GuillotineBlock::
00196 getConstSegmentCabinDTRangeProductAndPriceOrientedBookingSnapshotView (const
00197 BlockNumber_T iSCIdxBegin,
00198                                                  const BlockNumber_T iSCIdxEnd,
00199                                                  const DTD_T iDTD) const {
00200     const unsigned int lNbOfValueTypes = _valueTypesIndexMap.size();
00201     const unsigned int lValueIdxBegin = iSCIdxBegin * lNbOfValueTypes;
00202     const unsigned int lValueIdxEnd = (iSCIdxEnd + 1) * lNbOfValueTypes;
00203
00204     return _productAndPriceOrientedBookingSnapshotBlock [ boost::indices[
00205 SnapshotBlockRange_T(lValueIdxBegin, lValueIdxEnd)][iDTD] ];
00206 }
00207 // //////////////////////////////////////
00208 ConstSegmentCabinDTRangeSnapshotView_T GuillotineBlock::

```

```

00207     getConstSegmentCabinDTDRangeProductAndPriceOrientedBookingSnapshotView
00208     (const BlockNumber_T iSCIdxBegin, const BlockNumber_T iSCIdxEnd,
00209      const DTD_T iDTDBegin, const DTD_T iDTDEnd) const {
00210         const unsigned int lNbOfValueTypes = _valueTypesIndexMap.size();
00211         const unsigned int lValueIdxBegin = iSCIdxBegin * lNbOfValueTypes;
00212         const unsigned int lValueIdxEnd = (iSCIdxEnd + 1) * lNbOfValueTypes;
00213
00214         return _productAndPriceOrientedBookingSnapshotBlock [ boost::indices[
SnapshotBlockRange_T(lValueIdxBegin, lValueIdxEnd)][SnapshotBlockRange_T(iDTDBegi
n, iDTDEnd + 1)] ];
00215     }
00216
00217     // //////////////////////////////////////
00218     SegmentCabinDTDSnapshotView_T GuillotineBlock::
00219     getSegmentCabinDTDProductAndPriceOrientedBookingSnapshotView (const
BlockNumber_T iSCIdxBegin,
00220                                                                    const BlockNumber_T iSCIdxEnd,
00221                                                                    const DTD_T iDTD) {
00222         const unsigned int lNbOfValueTypes = _valueTypesIndexMap.size();
00223         const unsigned int lValueIdxBegin = iSCIdxBegin * lNbOfValueTypes;
00224         const unsigned int lValueIdxEnd = (iSCIdxEnd + 1) * lNbOfValueTypes;
00225
00226         return _productAndPriceOrientedBookingSnapshotBlock [ boost::indices[
SnapshotBlockRange_T(lValueIdxBegin, lValueIdxEnd)][iDTD] ];
00227     }
00228
00229     // //////////////////////////////////////
00230     SegmentCabinDTDRangeSnapshotView_T GuillotineBlock::
00231     getSegmentCabinDTDRangeProductAndPriceOrientedBookingSnapshotView (const
BlockNumber_T iSCIdxBegin,
00232                                                                    const BlockNumber_T iSCIdxEnd,
00233                                                                    const DTD_T iDTDBegin,
00234                                                                    const DTD_T iDTDEnd) {
00235         const unsigned int lNbOfValueTypes = _valueTypesIndexMap.size();
00236         const unsigned int lValueIdxBegin = iSCIdxBegin * lNbOfValueTypes;
00237         const unsigned int lValueIdxEnd = (iSCIdxEnd + 1) * lNbOfValueTypes;
00238
00239         return _productAndPriceOrientedBookingSnapshotBlock [ boost::indices[
SnapshotBlockRange_T(lValueIdxBegin, lValueIdxEnd)][SnapshotBlockRange_T(iDTDBegi
n, iDTDEnd + 1)] ];
00240     }
00241
00242     // //////////////////////////////////////
00243     ConstSegmentCabinDTDSnapshotView_T GuillotineBlock::
00244     getConstSegmentCabinDTDAvailabilitySnapshotView (const BlockNumber_T iSCIdxBegi
n,
00245                                                                    const BlockNumber_T iSCIdxEnd,
00246                                                                    const DTD_T iDTD) const {
00247         const unsigned int lNbOfValueTypes = _valueTypesIndexMap.size();
00248         const unsigned int lValueIdxBegin = iSCIdxBegin * lNbOfValueTypes;
00249         const unsigned int lValueIdxEnd = (iSCIdxEnd + 1) * lNbOfValueTypes;
00250
00251         return _availabilitySnapshotBlock [ boost::indices[SnapshotBlockRange_T(lValu
eIdxBegin, lValueIdxEnd)][iDTD] ];
00252     }
00253
00254     // //////////////////////////////////////
00255     ConstSegmentCabinDTDRangeSnapshotView_T GuillotineBlock::
00256     getConstSegmentCabinDTDRangeAvailabilitySnapshotView
00257     (const BlockNumber_T iSCIdxBegin, const BlockNumber_T iSCIdxEnd,
00258      const DTD_T iDTDBegin, const DTD_T iDTDEnd) const {
00259         const unsigned int lNbOfValueTypes = _valueTypesIndexMap.size();

```

```

00260     const unsigned int lValueIdxBegin = iSCIdxBegin * lNbOfValueTypes;
00261     const unsigned int lValueIdxEnd = (iSCIdxEnd + 1) * lNbOfValueTypes;
00262
00263     return _availabilitySnapshotBlock [ boost::indices[SnapshotBlockRange_T(lValueIdxBegin, lValueIdxEnd)][SnapshotBlockRange_T(iDTDBegin, iDTDEnd + 1)] ];
00264 }
00265
00266 // //////////////////////////////////////
00267 SegmentCabinDTDSnapshotView_T GuillotineBlock::
00268 getSegmentCabinDTDAvailabilitySnapshotView (const BlockNumber_T iSCIdxBegin,
00269                                             const BlockNumber_T iSCIdxEnd,
00270                                             const DTD_T iDTD) {
00271     const unsigned int lNbOfValueTypes = _valueTypesIndexMap.size();
00272     const unsigned int lValueIdxBegin = iSCIdxBegin * lNbOfValueTypes;
00273     const unsigned int lValueIdxEnd = (iSCIdxEnd + 1) * lNbOfValueTypes;
00274
00275     return _availabilitySnapshotBlock [ boost::indices[SnapshotBlockRange_T(lValueIdxBegin, lValueIdxEnd)][iDTD] ];
00276 }
00277
00278 // //////////////////////////////////////
00279 SegmentCabinDTDRangeSnapshotView_T GuillotineBlock::
00280 getSegmentCabinDTDRangeAvailabilitySnapshotView(const BlockNumber_T iSCIdxBegin
00281 ,
00282                                             const BlockNumber_T iSCIdxEnd,
00283                                             const DTD_T iDTDBegin,
00284                                             const DTD_T iDTDEnd) {
00285     const unsigned int lNbOfValueTypes = _valueTypesIndexMap.size();
00286     const unsigned int lValueIdxBegin = iSCIdxBegin * lNbOfValueTypes;
00287     const unsigned int lValueIdxEnd = (iSCIdxEnd + 1) * lNbOfValueTypes;
00288
00289     return _availabilitySnapshotBlock [ boost::indices[SnapshotBlockRange_T(lValueIdxBegin, lValueIdxEnd)][SnapshotBlockRange_T(iDTDBegin, iDTDEnd + 1)] ];
00290 }
00291
00292 // //////////////////////////////////////
00293 void GuillotineBlock::serialisationImplementationExport() const {
00294     std::ostringstream oStr;
00295     boost::archive::text_oarchive oa (oStr);
00296     oa << *this;
00297 }
00298
00299 // //////////////////////////////////////
00300 void GuillotineBlock::serialisationImplementationImport() {
00301     std::istream iStr;
00302     boost::archive::text_iarchive ia (iStr);
00303     ia >> *this;
00304 }
00305
00306 // //////////////////////////////////////
00307 template<class Archive>
00308 void GuillotineBlock::serialize (Archive& ioArchive,
00309                                 const unsigned int iFileVersion) {
00310     ioArchive & _key;
00311 }
00312 }
00313

```

35.303 stdair/bom/GuillotineBlock.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/GuillotineBlockKey.hpp>
#include <stdair/bom/GuillotineBlockTypes.hpp>
```

Classes

- class [stdair::GuillotineBlock](#)
Class representing the actual attributes for an airline guillotine-block.

Namespaces

- namespace [boost](#)
Forward declarations.
- namespace [boost::serialization](#)
- namespace [stdair](#)
Handle on the StdAir library context.

35.304 GuillotineBlock.hpp

```
00001 #ifndef __STDAIR_BOM_GUILLOTINEBLOCK_HPP
00002 #define __STDAIR_BOM_GUILLOTINEBLOCK_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/bom/BomAbstract.hpp>
00013 #include <stdair/bom/GuillotineBlockKey.hpp>
00014 #include <stdair/bom/GuillotineBlockTypes.hpp>
00015
00016 namespace boost {
00017     namespace serialization {
00018         class access;
00019     }
00020 }
00021
00022 namespace stdair {
00023     // Forward declarations
00024     class SegmentCabin;
00025 }
00026
```

```

00031 class GuillotineBlock : public BomAbstract {
00032     template <typename BOM> friend class FacBom;
00033     friend class FacBomManager;
00034     friend class boost::serialization::access;
00035
00036 public:
00037     // //////////// Type definitions ////////////
00041     typedef GuillotineBlockKey Key_T;
00042
00043
00044 public:
00045     // //////////// Getters ////////////
00047     const Key_T& getKey() const {
00048         return _key;
00049     }
00050
00052     BomAbstract* const getParent() const {
00053         return _parent;
00054     }
00055
00057     const GuillotineNumber_T& getGuillotineNumber() const {
00058         return _key.getGuillotineNumber();
00059     }
00060
00064     const HolderMap_T& getHolderMap() const {
00065         return _holderMap;
00066     }
00067
00069     const SegmentCabinIndexMap_T& getSegmentCabinIndexMap() const {
00070         return _segmentCabinIndexMap;
00071     }
00072
00074     const ValueTypeIndexMap_T& getValueTypeIndexMap() const {
00075         return _valueTypesIndexMap;
00076     }
00077
00079     const BlockIndex_T& getBlockIndex (const MapKey_T&) const;
00080
00082     const BlockNumber_T& getBlockNumber (const SegmentCabin&) const;
00083
00086     ConstSegmentCabinDTDSnapshotView_T
00087     getConstSegmentCabinDTDBookingSnapshotView (const BlockNumber_T,
00088                                                 const BlockNumber_T,
00089                                                 const DTD_T) const;
00090
00093     ConstSegmentCabinDTDRangeSnapshotView_T
00094     getConstSegmentCabinDTDRangeBookingSnapshotView (const BlockNumber_T,
00095                                                       const BlockNumber_T,
00096                                                       const DTD_T,
00097                                                       const DTD_T) const;
00098
00101     SegmentCabinDTDSnapshotView_T
00102     getSegmentCabinDTDBookingSnapshotView (const BlockNumber_T,
00103                                             const BlockNumber_T, const DTD_T);
00104
00107     SegmentCabinDTDRangeSnapshotView_T
00108     getSegmentCabinDTDRangeBookingSnapshotView (const BlockNumber_T,
00109                                                  const BlockNumber_T,
00110                                                  const DTD_T, const DTD_T);
00111
00114     ConstSegmentCabinDTDSnapshotView_T
00115     getConstSegmentCabinDTDCancellationSnapshotView (const BlockNumber_T,

```

```

00116                                     const BlockNumber_T,
00117                                     const DTD_T) const;
00118
00121     ConstSegmentCabinDTDRangeSnapshotView_T
00122     getConstSegmentCabinDTDRangeCancellationSnapshotView (const BlockNumber_T,
00123                                                         const BlockNumber_T,
00124                                                         const DTD_T,
00125                                                         const DTD_T) const;
00126
00129     SegmentCabinDTDSnapshotView_T
00130     getSegmentCabinDTDCancellationSnapshotView (const BlockNumber_T,
00131                                                 const BlockNumber_T,
00132                                                 const DTD_T);
00133
00136     SegmentCabinDTDRangeSnapshotView_T
00137     getSegmentCabinDTDRangeCancellationSnapshotView (const BlockNumber_T,
00138                                                         const BlockNumber_T,
00139                                                         const DTD_T, const DTD_T);
00140
00143     ConstSegmentCabinDTDSnapshotView_T
00144     getConstSegmentCabinDTDProductAndPriceOrientedBookingSnapshotView
00145     (const BlockNumber_T, const BlockNumber_T, const DTD_T) const;
00146
00149     ConstSegmentCabinDTDRangeSnapshotView_T
00150     getConstSegmentCabinDTDRangeProductAndPriceOrientedBookingSnapshotView
00151     (const BlockNumber_T, const BlockNumber_T, const DTD_T, const DTD_T) const;
00152
00155     SegmentCabinDTDSnapshotView_T
00156     getSegmentCabinDTDProductAndPriceOrientedBookingSnapshotView
00157     (const BlockNumber_T, const BlockNumber_T, const DTD_T);
00158
00161     SegmentCabinDTDRangeSnapshotView_T
00162     getSegmentCabinDTDRangeProductAndPriceOrientedBookingSnapshotView
00163     (const BlockNumber_T, const BlockNumber_T, const DTD_T, const DTD_T);
00164
00167     ConstSegmentCabinDTDSnapshotView_T
00168     getConstSegmentCabinDTDAvailabilitySnapshotView (const BlockNumber_T,
00169                                                         const BlockNumber_T,
00170                                                         const DTD_T) const;
00171
00174     ConstSegmentCabinDTDRangeSnapshotView_T
00175     getConstSegmentCabinDTDRangeAvailabilitySnapshotView (const BlockNumber_T,
00176                                                         const BlockNumber_T,
00177                                                         const DTD_T,
00178                                                         const DTD_T) const;
00179
00182     SegmentCabinDTDSnapshotView_T
00183     getSegmentCabinDTDAvailabilitySnapshotView (const BlockNumber_T,
00184                                                 const BlockNumber_T,
00185                                                 const DTD_T);
00186
00189     SegmentCabinDTDRangeSnapshotView_T
00190     getSegmentCabinDTDRangeAvailabilitySnapshotView (const BlockNumber_T,
00191                                                         const BlockNumber_T,
00192                                                         const DTD_T, const DTD_T);
00193
00194
00195 public:
00196     // //////////// Setters ////////////
00199     void initSnapshotBlocks (const SegmentCabinIndexMap_T&,
00200                             const ValueTypeIndexMap_T&);
00201

```

```

00202 public:
00203     // //////////// Display support methods ////////////
00209     void toStream (std::ostream& ioOut) const {
00210         ioOut << toString();
00211     }
00212
00218     void fromStream (std::istream& ioIn) {
00219     }
00220
00224     std::string toString() const;
00225
00229     const std::string describeKey() const {
00230         return _key.toString();
00231     }
00232
00233 public:
00234     // //////////// (Boost) Serialisation support methods ////////////
00239     template<class Archive>
00240     void serialize (Archive& ar, const unsigned int iFileVersion);
00241
00242 private:
00247     void serialisationImplementationExport() const;
00248     void serialisationImplementationImport();
00249
00250 protected:
00251     // //////////// Constructors and destructors ////////////
00256     GuillotineBlock (const Key_T&);
00257
00261     virtual ~GuillotineBlock();
00262
00263 private:
00267     GuillotineBlock();
00268
00272     GuillotineBlock (const GuillotineBlock&);
00273
00274 protected:
00275     // //////////// Attributes ////////////
00278     Key_T _key;
00279
00281     BomAbstract* _parent;
00282
00284     HolderMap_T _holderMap;
00285
00287     SegmentCabinIndexMap_T _segmentCabinIndexMap;
00288
00291     ValueTypeIndexMap_T _valueTypesIndexMap;
00292
00294     SnapshotBlock_T _bookingSnapshotBlock;
00295
00297     SnapshotBlock_T _cancellationSnapshotBlock;
00298
00300     SnapshotBlock_T _productAndPriceOrientedBookingSnapshotBlock;
00301
00303     SnapshotBlock_T _availabilitySnapshotBlock;
00304 };
00305
00306 }
00307 #endif // __STDAIR_BOM_GUILLOTINEBLOCK_HPP
00308

```

35.305 stdair/bom/GuillotineBlockKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/GuillotineBlockKey.hpp>
```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

Functions

- template void [stdair::GuillotineBlockKey::serialize< ba::text_oarchive >](#) (ba::text_oarchive &, unsigned int)
- template void [stdair::GuillotineBlockKey::serialize< ba::text_iarchive >](#) (ba::text_iarchive &, unsigned int)

35.306 GuillotineBlockKey.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/access.hpp>
00011 // StdAir
00012 #include <stdair/basic/BasConst_Inventory.hpp>
00013 #include <stdair/basic/BasConst_BomDisplay.hpp>
00014 #include <stdair/bom/GuillotineBlockKey.hpp>
00015
00016 namespace stdair {
00017
00018 // //////////////////////////////////////
00019 GuillotineBlockKey::GuillotineBlockKey()
00020 : _guillotineNumber (DEFAULT_GUILLOTINE_NUMBER) {
00021     assert (false);
00022 }
00023
00024 // //////////////////////////////////////
```

```

00025  GuillotineBlockKey::
00026  GuillotineBlockKey (const GuillotineNumber_T& iGuillotineNumber)
00027      : _guillotineNumber (iGuillotineNumber) {
00028  }
00029
00030  // //////////////////////////////////////
00031  GuillotineBlockKey::GuillotineBlockKey (const GuillotineBlockKey& iKey)
00032      : _guillotineNumber (iKey._guillotineNumber) {
00033  }
00034
00035  // //////////////////////////////////////
00036  GuillotineBlockKey::~GuillotineBlockKey() {
00037  }
00038
00039  // //////////////////////////////////////
00040  void GuillotineBlockKey::toStream (std::ostream& ioOut) const {
00041      ioOut << "GuillotineBlockKey: " << toString();
00042  }
00043
00044  // //////////////////////////////////////
00045  void GuillotineBlockKey::fromStream (std::istream& ioIn) {
00046  }
00047
00048  // //////////////////////////////////////
00049  const std::string GuillotineBlockKey::toString() const {
00050      std::ostringstream oStr;
00051      oStr << _guillotineNumber;
00052      return oStr.str();
00053  }
00054
00055  // //////////////////////////////////////
00056  void GuillotineBlockKey::serialisationImplementationExport() const {
00057      std::ostringstream oStr;
00058      boost::archive::text_oarchive oa (oStr);
00059      oa << *this;
00060  }
00061
00062  // //////////////////////////////////////
00063  void GuillotineBlockKey::serialisationImplementationImport() {
00064      std::istringstream iStr;
00065      boost::archive::text_iarchive ia (iStr);
00066      ia >> *this;
00067  }
00068
00069  // //////////////////////////////////////
00070  template<class Archive>
00071  void GuillotineBlockKey::serialize (Archive& ioArchive,
00072                                     const unsigned int iFileVersion) {
00073      ioArchive & _guillotineNumber;
00074  }
00075
00076  // //////////////////////////////////////
00077  // Explicit template instantiation
00078  namespace ba = boost::archive;
00079  template void GuillotineBlockKey::
00080  serialize<ba::text_oarchive> (ba::text_oarchive&, unsigned int);
00081  template void GuillotineBlockKey::
00082  serialize<ba::text_iarchive> (ba::text_iarchive&, unsigned int);
00083  // //////////////////////////////////////
00084
00085  }

```

35.307 stdair/bom/GuillotineBlockKey.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

Classes

- struct [stdair::GuillotineBlockKey](#)
Key of a given guillotine block, made of a guillotine number.

Namespaces

- namespace [boost](#)
Forward declarations.
- namespace [boost::serialization](#)
- namespace [stdair](#)
Handle on the StdAir library context.

35.308 GuillotineBlockKey.hpp

```
00001 #ifndef __STDAIR_BOM_GUILLOTINEBLOCKKEY_HPP
00002 #define __STDAIR_BOM_GUILLOTINEBLOCKKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_basic_types.hpp>
00012 #include <stdair/bom/KeyAbstract.hpp>
00013
00015 namespace boost {
00016     namespace serialization {
00017         class access;
00018     }
00019 }
00020
00021 namespace stdair {
00022
00026     struct GuillotineBlockKey : public KeyAbstract {
00027         friend class boost::serialization::access;
00028
00029         // ////////////////////////////////// Constructors and destructors //////////////////////////////////
00030     private:
00034         GuillotineBlockKey();
00035
00036     public:
```

```

00040     GuillotineBlockKey (const GuillotineNumber_T&);
00041
00045     GuillotineBlockKey (const GuillotineBlockKey&);
00046
00050     ~GuillotineBlockKey();
00051
00052
00053 public:
00054     // //////////// Getters ////////////
00056     const GuillotineNumber_T& getGuillotineNumber() const {
00057         return _guillotineNumber;
00058     }
00059
00060
00061 public:
00062     // //////////// Display support methods ////////////
00068     void toStream (std::ostream& ioOut) const;
00069
00075     void fromStream (std::istream& ioIn);
00076
00086     const std::string toString() const;
00087
00088
00089 public:
00090     // //////////// (Boost) Serialisation support methods ////////////
00094     template<class Archive>
00095     void serialize (Archive& ar, const unsigned int iFileVersion);
00096
00097 private:
00102     void serialisationImplementationExport() const;
00103     void serialisationImplementationImport();
00104
00105
00106 private:
00107     // //////////// Attributes ////////////
00111     GuillotineNumber_T _guillotineNumber;
00112
00113 };
00114
00115 }
00116 #endif // __STDAIR_BOM_GUILLOTINEBLOCKKEY_HPP

```

35.309 stdair/bom/GuillotineBlockTypes.hpp File Reference

```

#include <map>
#include <list>
#include <boost/multi_array.hpp>
#include <stdair/bom/key_types.hpp>

```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

Typedefs

- typedef std::list< GuillotineBlock * > [stdair::GuillotineBlockList_T](#)
- typedef std::map< const MapKey_T, GuillotineBlock * > [stdair::GuillotineBlockMap_T](#)
- typedef std::map< const SegmentCabin *, BlockNumber_T > [stdair::SegmentCabinIndexMap_T](#)
- typedef std::map< const MapKey_T, BlockIndex_T > [stdair::ValueTypeIndexMap_T](#)

35.310 GuillotineBlockTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_GUILLOTINEBLOCKTYPES_HPP
00003 #define __STDAIR_BOM_GUILLOTINEBLOCKTYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // BOOST
00012 #include <boost/multi_array.hpp>
00013 // StdAir
00014 #include <stdair/bom/key_types.hpp>
00015
00016 namespace stdair {
00017
00018     // Forward declarations
00019     class GuillotineBlock;
00020     class SegmentCabin;
00021
00022     // ////////////////////////////////// Type definitions //////////////////////////////////
00024     typedef std::list<GuillotineBlock*> GuillotineBlockList\_T;
00025
00027     typedef std::map<const MapKey_T, GuillotineBlock*> GuillotineBlockMap\_T;
00028
00030     typedef std::map<const SegmentCabin*, BlockNumber_T> SegmentCabinIndexMap\_T;
00031
00033     typedef std::map<const MapKey_T, BlockIndex_T> ValueTypeIndexMap\_T;
00034
00035 }
00036 #endif // __STDAIR_BOM_GUILLOTINEBLOCKTYPES_HPP
00037

```

35.311 stdair/bom/Inventory.cpp File Reference

```

#include <cassert>

#include <sstream>

#include <stdair/basic/BasConst_Inventory.hpp>

#include <stdair/bom/BomManager.hpp>

#include <stdair/bom/Inventory.hpp>

```

```
#include <stdair/bom/FlightDate.hpp>
```

Namespaces

- namespace `stdair`

Handle on the StdAir library context.

35.312 Inventory.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_Inventory.hpp>
00009 #include <stdair/bom/BomManager.hpp>
00010 #include <stdair/bom/Inventory.hpp>
00011 #include <stdair/bom/FlightDate.hpp>
00012
00013 namespace stdair {
00014
00015     // //////////////////////////////////////
00016     Inventory::Inventory() : _key (DEFAULT_AIRLINE_CODE), _parent (NULL) {
00017         // That constructor is used by the serialisation process
00018     }
00019
00020     // //////////////////////////////////////
00021     Inventory::Inventory (const Inventory&)
00022         : _key (DEFAULT_AIRLINE_CODE), _parent (NULL) {
00023         assert (false);
00024     }
00025
00026     // //////////////////////////////////////
00027     Inventory::Inventory (const Key_T& iKey) : _key (iKey), _parent (NULL) {
00028     }
00029
00030     // //////////////////////////////////////
00031     Inventory::~Inventory() {
00032     }
00033
00034     // //////////////////////////////////////
00035     std::string Inventory::toString() const {
00036         std::ostringstream oStr;
00037         oStr << describeKey();
00038         return oStr.str();
00039     }
00040
00041     // //////////////////////////////////////
00042     FlightDate* Inventory::
00043     getFlightDate (const std::string& iFlightDateKeyStr) const {
00044         FlightDate* oFlightDate_ptr =
00045             BomManager::getObjectPtr<FlightDate> (*this, iFlightDateKeyStr);
00046         return oFlightDate_ptr;
00047     }
00048
00049     // //////////////////////////////////////
```

```

00050     FlightDate* Inventory::
00051     getFlightDate (const FlightDateKey& iFlightDateKey) const {
00052         return getFlightDate (iFlightDateKey.toString());
00053     }
00054
00055 }
00056

```

35.313 stdair/bom/Inventory.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/InventoryKey.hpp>
#include <stdair/bom/InventoryTypes.hpp>

```

Classes

- class [stdair::Inventory](#)
Class representing the actual attributes for an airline inventory.

Namespaces

- namespace [boost](#)
Forward declarations.
- namespace [boost::serialization](#)
- namespace [stdair](#)
Handle on the StdAir library context.

35.314 Inventory.hpp

```

00001 #ifndef __STDAIR_BOM_INVENTORY_HPP
00002 #define __STDAIR_BOM_INVENTORY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/bom/BomAbstract.hpp>
00013 #include <stdair/bom/InventoryKey.hpp>
00014 #include <stdair/bom/InventoryTypes.hpp>
00015
00017 namespace boost {

```

```

00018 namespace serialization {
00019     class access;
00020 }
00021 }
00022
00023 namespace stdair {
00024
00026     class AirlineFeature;
00027     struct FlightDateKey;
00028     class FlightDate;
00029
00033     class Inventory : public BomAbstract {
00034     template <typename BOM> friend class FacBom;
00035     friend class FacBomManager;
00036     friend class boost::serialization::access;
00037
00038     public :
00039         // /////////// Type definitions ///////////
00043         typedef InventoryKey Key_T;
00044
00045
00046     public:
00047         // /////////// Getters ///////////
00049         const Key_T& getKey() const {
00050             return _key;
00051         }
00052
00054         const AirlineCode_T& getAirlineCode() const {
00055             return _key.getAirlineCode();
00056         }
00057
00059         BomAbstract* const getParent() const {
00060             return _parent;
00061         }
00062
00064         const HolderMap_T& getHolderMap() const {
00065             return _holderMap;
00066         }
00067
00078         FlightDate* getFlightDate (const std::string& iFlightDateKeyStr) const;
00079
00090         FlightDate* getFlightDate (const FlightDateKey&) const;
00091
00092
00093     public:
00094         // /////////// Setters ///////////
00096         void setAirlineFeature (const AirlineFeature* ioAirlineFeaturePtr) {
00097             _airlineFeature = ioAirlineFeaturePtr;
00098         }
00099
00100
00101     public:
00102         // /////////// Display support methods ///////////
00108         void toStream (std::ostream& ioOut) const {
00109             ioOut << toString();
00110         }
00111
00117         void fromStream (std::istream& ioIn) {
00118         }
00119
00123         std::string toString() const;
00124

```

```

00128     const std::string describeKey() const {
00129         return _key.toString();
00130     }
00131
00132
00133 public:
00134     // //////////// (Boost) Serialisation support methods ////////////
00138     template<class Archive>
00139     void serialize (Archive& ar, const unsigned int iFileVersion);
00140
00141 private:
00149     void serialisationImplementationExport() const;
00150     void serialisationImplementationImport();
00151
00152
00153 protected:
00154     // //////////// Constructors and destructors ////////////
00158     Inventory (const Key_T&);
00162     ~Inventory();
00163
00164 private:
00168     Inventory();
00172     Inventory (const Inventory&);
00173
00174
00175 protected:
00176     // //////////// Attributes ////////////
00180     Key_T _key;
00181
00185     BomAbstract* _parent;
00186
00190     const AirlineFeature* _airlineFeature;
00191
00195     HolderMap_T _holderMap;
00196 };
00197
00198 }
00199 #endif // __STDAIR_BOM_INVENTORY_HPP
00200

```

35.315 stdair/bom/InventoryKey.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/InventoryKey.hpp>

```

Namespaces

- namespace `stdair`

Handle on the StdAir library context.

Functions

- template void `stdair::InventoryKey::serialize< ba::text_oarchive >` (ba::text_oarchive &, unsigned int)
- template void `stdair::InventoryKey::serialize< ba::text_iarchive >` (ba::text_iarchive &, unsigned int)

35.316 InventoryKey.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/access.hpp>
00011 // StdAir
00012 #include <stdair/basic/BasConst_Inventory.hpp>
00013 #include <stdair/bom/InventoryKey.hpp>
00014
00015 namespace stdair {
00016
00017 // //////////////////////////////////////
00018 InventoryKey::InventoryKey() : _airlineCode (DEFAULT_AIRLINE_CODE) {
00019     assert (false);
00020 }
00021
00022 // //////////////////////////////////////
00023 InventoryKey::InventoryKey (const AirlineCode_T& iAirlineCode)
00024 : _airlineCode (iAirlineCode) {
00025 }
00026
00027 // //////////////////////////////////////
00028 InventoryKey::InventoryKey (const InventoryKey& iKey)
00029 : _airlineCode (iKey._airlineCode) {
00030 }
00031
00032 // //////////////////////////////////////
00033 InventoryKey::~InventoryKey() {
00034 }
00035
00036 // //////////////////////////////////////
00037 void InventoryKey::toStream (std::ostream& ioOut) const {
00038     ioOut << "InventoryKey: " << toString();
00039 }
00040
00041 // //////////////////////////////////////
00042 void InventoryKey::fromStream (std::istream& ioIn) {
00043 }
00044
00045 // //////////////////////////////////////
00046 const std::string InventoryKey::toString() const {
00047     std::ostringstream oStr;

```

```

00048     ostr << _airlineCode;
00049     return ostr.str();
00050 }
00051
00052 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00053 void InventoryKey::serialisationImplementationExport() const {
00054     std::ostringstream ostr;
00055     boost::archive::text_oarchive oa (ostr);
00056     oa << *this;
00057 }
00058
00059 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00060 void InventoryKey::serialisationImplementationImport() {
00061     std::istringstream istr;
00062     boost::archive::text_iarchive ia (istr);
00063     ia >> *this;
00064 }
00065
00066 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00067 template<class Archive>
00068 void InventoryKey::serialize (Archive& ioArchive,
00069                             const unsigned int iFileVersion) {
00070     ioArchive & _airlineCode;
00071 }
00072
00073 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00074 // Explicit template instantiation
00075 namespace ba = boost::archive;
00076 template void InventoryKey::serialize<ba::text_oarchive> (ba::text_oarchive&,
00077                                                         unsigned int);
00078 template void InventoryKey::serialize<ba::text_iarchive> (ba::text_iarchive&,
00079                                                         unsigned int);
00080 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00081
00082 }

```

35.317 stdair/bom/InventoryKey.hpp File Reference

```

#include <iosfwd>

#include <string>

#include <stdair/stdair_inventory_types.hpp>

#include <stdair/bom/KeyAbstract.hpp>

```

Classes

- struct [stdair::InventoryKey](#)
Key of a given inventory, made of the airline code.

Namespaces

- namespace [boost](#)
Forward declarations.
- namespace [boost::serialization](#)

- namespace `stdair`

Handle on the StdAir library context.

35.318 InventoryKey.hpp

```

00001 #ifndef __STDAIR_BOM_INVENTORYKEY_HPP
00002 #define __STDAIR_BOM_INVENTORYKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/bom/KeyAbstract.hpp>
00013
00015 namespace boost {
00016     namespace serialization {
00017         class access;
00018     }
00019 }
00020
00021 namespace stdair {
00022
00026     struct InventoryKey : public KeyAbstract {
00027         friend class boost::serialization::access;
00028
00029         // ////////////////////////////////// Constructors and destructors //////////////////////////////////
00030     private:
00034         InventoryKey();
00035
00036     public:
00037         // ////////////////////////////////// Construction //////////////////////////////////
00041         InventoryKey (const AirlineCode_T& iAirlineCode);
00042
00046         InventoryKey (const InventoryKey&);
00047
00051         ~InventoryKey();
00052
00053
00054         // ////////////////////////////////// Getters //////////////////////////////////
00058         const AirlineCode_T& getAirlineCode() const {
00059             return _airlineCode;
00060         }
00061
00062
00063     public:
00064         // ////////////////////////////////// Display support methods //////////////////////////////////
00070         void toStream (std::ostream& ioOut) const;
00071
00077         void fromStream (std::istream& ioIn);
00078
00088         const std::string toString() const;
00089
00090
00091     public:
00092         // ////////////////////////////////// (Boost) Serialisation support methods //////////////////////////////////
00096         template<class Archive>

```

```

00097     void serialize (Archive& ar, const unsigned int iFileVersion);
00098
00099     private:
00104         void serialisationImplementationExport() const;
00105         void serialisationImplementationImport();
00106
00107
00108     private:
00109         // ////////////////////////////////// Attributes //////////////////////////////////
00113         AirlineCode_T _airlineCode;
00114     };
00115
00116 }
00117 #endif // __STDAIR_BOM_INVENTORYKEY_HPP

```

35.319 stdair/bom/InventoryTypes.hpp File Reference

```

#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>

```

Namespaces

- namespace **stdair**
Handle on the StdAir library context.

Typedefs

- typedef std::list< Inventory * > **stdair::InventoryList_T**
- typedef std::map< const MapKey_T, Inventory * > **stdair::InventoryMap_T**

35.320 InventoryTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_INVENTORYTYPES_HPP
00003 #define __STDAIR_BOM_INVENTORYTYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // Stdair
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016     // Forward declarations.
00017     class Inventory;
00018
00020     typedef std::list<Inventory*> InventoryList_T;

```

```

00021
00023     typedef std::map<const MapKey_T, Inventory*> InventoryMap_T;
00024
00025 }
00026 #endif // __STDAIR_BOM_INVENTORYTYPES_HPP

```

35.321 stdair/bom/key_types.hpp File Reference

```

#include <string>
#include <list>

```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

Typedefs

- typedef std::string [stdair::MapKey_T](#)
- typedef std::list< std::string > [stdair::KeyList_T](#)

35.322 key_types.hpp

```

00001 #ifndef __STDAIR_BOM_KEY_TYPES_HPP
00002 #define __STDAIR_BOM_KEY_TYPES_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <list>
00010
00011 namespace stdair {
00012
00013     // ////////////////////////////////// Type definitions //////////////////////////////////
00015     typedef std::string MapKey_T;
00016
00018     typedef std::list<std::string> KeyList_T;
00019
00020 }
00021 #endif // __STDAIR_BOM_KEY_TYPES_HPP

```

35.323 stdair/bom/KeyAbstract.hpp File Reference

```

#include <iosfwd>
#include <string>

```

Classes

- struct [stdair::KeyAbstract](#)

Base class for the keys of Business Object Model (BOM) layer.

Note that that key allows to differentiate two objects at the same level only. For instance, the segment-date key allows to differentiate two segment-dates under a given flight-date, but does not allow to differentiate two segment-dates in general.

Namespaces

- namespace [stdair](#)

Handle on the StdAir library context.

Functions

- `template<class charT , class traits >
std::basic_ostream< charT, traits > & operator<< (std::basic_ostream< charT, traits > &ioOut, const stdair::KeyAbstract &iKey)`
- `template<class charT , class traits >
std::basic_istream< charT, traits > & operator>> (std::basic_istream< charT, traits > &ioln, stdair::KeyAbstract &iKey)`

35.323.1 Function Documentation

35.323.1.1 `template<class charT , class traits > std::basic_ostream<charT, traits>&
operator<< (std::basic_ostream< charT, traits > & ioOut, const
stdair::KeyAbstract & iKey) [inline]`

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (p653) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 69 of file [KeyAbstract.hpp](#).

References [stdair::KeyAbstract::toStream\(\)](#).

35.323.1.2 `template<class charT , class traits > std::basic_istream<charT, traits>&
operator>> (std::basic_istream< charT, traits > & ioln, stdair::KeyAbstract &
ioKey) [inline]`

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (pp655-657) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 97 of file [KeyAbstract.hpp](#).

References [stdair::KeyAbstract::fromStream\(\)](#).

35.324 KeyAbstract.hpp

```

00001
00007 #ifndef __STDAIR_BOM_KEYABSTRACT_HPP
00008 #define __STDAIR_BOM_KEYABSTRACT_HPP
00009
00010 // //////////////////////////////////////
00011 // Import section
00012 // //////////////////////////////////////
00013 // STL
00014 #include <iosfwd>
00015 #include <string>
00016
00017 namespace stdair {
00018
00026     struct KeyAbstract {
00027     public:
00028
00029         // /////////// Display support methods ///////////
00034         virtual void toStream (std::ostream& ioOut) const {}
00035
00040         virtual void fromStream (std::istream& ioIn) {}
00041
00051         virtual const std::string toString() const { return std::string("Hello!"); }
00052
00056         virtual ~KeyAbstract() {}
00057     };
00058
00059 }
00060
00066 template <class charT, class traits>
00067 inline
00068 std::basic_ostream<charT, traits>&
00069 operator<< (std::basic_ostream<charT, traits>& ioOut,
00070           const stdair::KeyAbstract& iKey) {
00076     std::basic_ostringstream<charT,traits> ostr;
00077     ostr.copyfmt (ioOut);
00078     ostr.width (0);
00079
00080     // Fill string stream
00081     iKey.toStream (ostr);
00082
00083     // Print string stream
00084     ioOut << ostr.str();
00085
00086     return ioOut;
00087 }
00088
00094 template <class charT, class traits>
00095 inline
00096 std::basic_istream<charT, traits>&
00097 operator>> (std::basic_istream<charT, traits>& ioIn,
00098           stdair::KeyAbstract& ioKey) {
00099     // Fill Key object with input stream
00100     ioKey.fromStream (ioIn);
00101     return ioIn;
00102 }
00103
00104 #endif // __STDAIR_BOM_KEYABSTRACT_HPP

```

35.325 stdair/bom/LegCabin.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_BookingClass.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/LegCabin.hpp>

```

Namespaces

- namespace [stdair](#)

Handle on the StdAir library context.

35.326 LegCabin.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_BookingClass.hpp>
00009 #include <stdair/basic/BasConst_Inventory.hpp>
00010 #include <stdair/basic/BasConst_BomDisplay.hpp>
00011 #include <stdair/bom/BomManager.hpp>
00012 #include <stdair/bom/LegDate.hpp>
00013 #include <stdair/bom/LegCabin.hpp>
00014
00015
00016 namespace stdair {
00017
00018 // //////////////////////////////////////
00019 LegCabin::LegCabin() : _key (DEFAULT_CABIN_CODE), _parent (NULL) {
00020     assert (false);
00021 }
00022
00023 // //////////////////////////////////////
00024 LegCabin::LegCabin (const LegCabin&)
00025     : _key (DEFAULT_CABIN_CODE), _parent (NULL) {
00026     assert (false);
00027 }
00028
00029 // //////////////////////////////////////
00030 LegCabin::LegCabin (const Key_T& iKey)
00031     : _key (iKey), _parent (NULL),
00032       _offeredCapacity (DEFAULT_CABIN_CAPACITY),
00033       _physicalCapacity (DEFAULT_CABIN_CAPACITY),

```

```

00034     _soldSeat (DEFAULT_CLASS_NB_OF_BOOKINGS),
00035     _committedSpace (DEFAULT_COMMITTED_SPACE),
00036     _availabilityPool (DEFAULT_AVAILABILITY),
00037     _availability (DEFAULT_AVAILABILITY),
00038     _currentBidPrice (DEFAULT_BID_PRICE),
00039     _bidPriceVector (DEFAULT_BID_PRICE_VECTOR) {
00040 }
00041
00042 // //////////////////////////////////////
00043 LegCabin::~LegCabin() {
00044 }
00045
00046 // //////////////////////////////////////
00047 void LegCabin::setCapacities (const CabinCapacity_T& iCapacity) {
00048     _offeredCapacity = iCapacity;
00049     _physicalCapacity = iCapacity;
00050     setAvailabilityPool (iCapacity - _committedSpace);
00051 }
00052
00053 // //////////////////////////////////////
00054 const MapKey_T LegCabin::getFullerKey() const {
00055     const LegDate& lLegDate = BomManager::getParent<LegDate> (*this);
00056
00057     const MapKey_T oFullKey =
00058         lLegDate.describeKey() + DEFAULT_KEY_FLD_DELIMITER + getCabinCode();
00059     return oFullKey;
00060 }
00061
00062 // //////////////////////////////////////
00063 std::string LegCabin::toString() const {
00064     std::ostringstream oStr;
00065     oStr << describeKey();
00066     return oStr.str();
00067 }
00068
00069 // //////////////////////////////////////
00070 const std::string LegCabin::displayVirtualClassList () const {
00071     std::ostringstream oStr;
00072
00073     for (VirtualClassList_T::const_iterator itVC = _virtualClassList.begin();
00074          itVC != _virtualClassList.end(); ++itVC) {
00075         const VirtualClassStruct& lVC = *itVC;
00076         oStr << std::endl << "Yield: " << std::fixed << std::setprecision (2)
00077             << lVC.getYield()
00078             << ", Protection: " << std::fixed << std::setprecision (2)
00079             << lVC.getCumulatedProtection()
00080             << ", Booking limit: " << std::fixed << std::setprecision (2)
00081             << lVC.getCumulatedBookingLimit();
00082     }
00083
00084     return oStr.str();
00085 }
00086
00087 // //////////////////////////////////////
00088 void LegCabin::updateFromReservation (const NbOfBookings_T& iNbOfBookings) {
00089     _committedSpace += iNbOfBookings;
00090     _availabilityPool = _offeredCapacity - _committedSpace;
00091 }
00092
00093 // //////////////////////////////////////
00094 void LegCabin::updateCurrentBidPrice() {
00095     const unsigned short lAvailabilityPool =

```

```

00096         static_cast<unsigned short> (std::floor (_availabilityPool));
00097
00098     if (lAvailabilityPool >= 1) {
00099         const unsigned short lBidPriceVectorSize = _bidPriceVector.size();
00100         if (lBidPriceVectorSize >= lAvailabilityPool) {
00101             _currentBidPrice = _bidPriceVector.at (lAvailabilityPool - 1);
00102         }
00103     }
00104 }
00105
00106 // //////////////////////////////////////
00107 void LegCabin::addDemandInformation (const YieldValue_T& iYield,
00108                                     const MeanValue_T& iMeanValue,
00109                                     const StdDevValue_T& iStdDevValue) {
00110     //
00111     const int lYieldLevel =
00112         static_cast<int> (std::floor (iYield + 0.5));
00113
00114     //
00115     YieldLevelDemandMap_T::iterator itDemand =
00116         _yieldLevelDemandMap.find (lYieldLevel);
00117
00118     if (itDemand == _yieldLevelDemandMap.end()) {
00119         MeanStdDevPair_T lMeanStdDevPair (iMeanValue, iStdDevValue);
00120         const bool hasInsertBeenSuccessful = _yieldLevelDemandMap.
00121             insert (YieldLevelDemandMap_T::value_type (lYieldLevel,
00122                                                         lMeanStdDevPair)).second;
00123         assert (hasInsertBeenSuccessful == true);
00124     } else {
00125         //
00126         MeanStdDevPair_T& lMeanStdDevPair = itDemand->second;
00127         MeanValue_T lMeanValue = iMeanValue + lMeanStdDevPair.first;
00128         StdDevValue_T lStdDevValue = iStdDevValue * iStdDevValue + lMeanStdDevPair.
00129             second * lMeanStdDevPair.second;
00130         lStdDevValue = std::sqrt (lStdDevValue);
00131
00132         //
00133         lMeanStdDevPair = MeanStdDevPair_T (lMeanValue, lStdDevValue);
00134     }
00135 }
00136
00137 }
00138

```

35.327 stdair/bom/LegCabin.hpp File Reference

```

#include <iosfwd>

#include <string>

#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/LegCabinKey.hpp>
#include <stdair/bom/LegCabinTypes.hpp>

```

```
#include <stdair/bom/VirtualClassStruct.hpp>
#include <stdair/bom/VirtualClassTypes.hpp>
```

Classes

- class `stdair::LegCabin`
Class representing the actual attributes for an airline leg-cabin.

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

35.328 LegCabin.hpp

```
00001 #ifndef __STDAIR_BOM_LEGCABIN_HPP
00002 #define __STDAIR_BOM_LEGCABIN_HPP
00003 // //////////////////////////////////////
00004 // Import section
00005 // //////////////////////////////////////
00006 // STL
00007 #include <iosfwd>
00008 #include <string>
00009 // StdAir
00010 #include <stdair/stdair_inventory_types.hpp>
00011 #include <stdair/stdair_maths_types.hpp>
00012 #include <stdair/bom/BomAbstract.hpp>
00013 #include <stdair/bom/LegCabinKey.hpp>
00014 #include <stdair/bom/LegCabinTypes.hpp>
00015 #include <stdair/bom/VirtualClassStruct.hpp>
00016 #include <stdair/bom/VirtualClassTypes.hpp>
00017
00018 namespace stdair {
00019
00020     class LegCabin : public BomAbstract {
00021     public:
00022         template <typename BOM> friend class FacBom;
00023         friend class FacBomManager;
00024
00025         // ////////////////////////////////// Type definitions //////////////////////////////////
00026         typedef LegCabinKey Key_T;
00027
00028     public:
00029         // ////////////////////////////////// Getters //////////////////////////////////
00030         const Key_T& getKey() const {
00031             return _key;
00032         }
00033
00034         BomAbstract* const getParent() const {
00035             return _parent;
00036         }
00037
00038         const CabinCode_T& getCabinCode() const {
00039             return _key.getCabinCode();
00040         }
00041     }
00042 }
```

```
00057
00065     const MapKey_T getFullerKey() const;
00066
00070     const HolderMap_T& getHolderMap() const {
00071         return _holderMap;
00072     }
00073
00075     const CabinCapacity_T& getOfferedCapacity() const {
00076         return _offeredCapacity;
00077     }
00078
00080     const CabinCapacity_T& getPhysicalCapacity() const {
00081         return _physicalCapacity;
00082     }
00083
00085     const NbOfSeats_T& getSoldSeat() const {
00086         return _soldSeat;
00087     }
00088
00090     const CommittedSpace_T& getCommittedSpace() const {
00091         return _committedSpace;
00092     }
00093
00095     const Availability_T& getAvailabilityPool() const {
00096         return _availabilityPool;
00097     }
00098
00100     const Availability_T& getAvailability() const {
00101         return _availability;
00102     }
00103
00105     const BidPrice_T& getCurrentBidPrice() const {
00106         return _currentBidPrice;
00107     }
00108
00110     const BidPrice_T& getPreviousBidPrice() const {
00111         return _previousBidPrice;
00112     }
00113
00115     const BidPriceVector_T& getBidPriceVector() const {
00116         return _bidPriceVector;
00117     }
00118
00120     const CapacityAdjustment_T& getRegradeAdjustment() const {
00121         return _dcsRegrade;
00122     }
00123
00125     const AuthorizationLevel_T& getAuthorizationLevel() const {
00126         return _au;
00127     }
00128
00130     const UPR_T& getUPR() const {
00131         return _upr;
00132     }
00133
00135     const Availability_T& getNetAvailability() const {
00136         return _nav;
00137     }
00138
00140     const Availability_T& getGrossAvailability() const {
00141         return _gav;
00142     }
```

```
00143
00144     const OverbookingRate_T& getAvgCancellationPercentage() const {
00145         return _acp;
00146     }
00147
00148     const NbOfSeats_T& getETB() const {
00149         return _etb;
00150     }
00151
00152     const NbOfSeats_T& getStaffNbOfSeats() const {
00153         return _staffNbOfBookings;
00154     }
00155
00156     const NbOfSeats_T& getWLNbOfSeats() const {
00157         return _wlNbOfBookings;
00158     }
00159
00160     const NbOfSeats_T& getGroupNbOfSeats() const {
00161         return _groupNbOfBookings;
00162     }
00163
00164     VirtualClassList_T& getVirtualClassList() {
00165         return _virtualClassList;
00166     }
00167
00168     BidPriceVector_T& getBidPriceVector() {
00169         return _bidPriceVector;
00170     }
00171
00172     const YieldLevelDemandMap_T getYieldLevelDemandMap() {
00173         return _yieldLevelDemandMap;
00174     }
00175
00176 public:
00177     // //////////// Setters ////////////
00178     void setCapacities (const CabinCapacity_T& iCapacity);
00179
00180     void setSoldSeat (const NbOfSeats_T& iSoldSeat) {
00181         _soldSeat = iSoldSeat;
00182     }
00183
00184     void setCommittedSpace (const CommittedSpace_T& iCommittedSpace) {
00185         _committedSpace = iCommittedSpace;
00186     }
00187
00188     void setAvailabilityPool (const Availability_T& iAvailabilityPool) {
00189         _availabilityPool = iAvailabilityPool;
00190     }
00191
00192     void setAvailability (const Availability_T& iAvailability) {
00193         _availability = iAvailability;
00194     }
00195
00196     void setCurrentBidPrice (const BidPrice_T& iBidPrice) {
00197         _currentBidPrice = iBidPrice;
00198     }
00199
00200     void setPreviousBidPrice (const BidPrice_T& iBidPrice) {
00201         _previousBidPrice = iBidPrice;
00202     }
00203
00204 }
```

```
00220
00222     void updatePreviousBidPrice () {
00223         _previousBidPrice = _currentBidPrice;
00224     }
00225
00227     void setRegradeAdjustment (const CapacityAdjustment_T& iRegradeAdjustment) {
00228         _dcsRegrade = iRegradeAdjustment;
00229     }
00230
00232     void setAuthorizationLevel (const AuthorizationLevel_T& iAU) {
00233         _au = iAU;
00234     }
00235
00237     void setUPR (const UPR_T& iUPR) {
00238         _upr = iUPR;
00239     }
00240
00242     void setNetAvailability (const Availability_T& iNAV) {
00243         _nav = iNAV;
00244     }
00245
00247     void setGrossAvailability (const Availability_T& iGAV) {
00248         _gav = iGAV;
00249     }
00250
00252     void setAvgCancellationPercentage (const OverbookingRate_T& iACP) {
00253         _acp = iACP;
00254     }
00255
00257     void setETB (const NbOfSeats_T& iETB) {
00258         _etb = iETB;
00259     }
00260
00262     void setStaffNbOfSeats (const NbOfSeats_T& iStaffSeats) {
00263         _staffNbOfBookings = iStaffSeats;
00264     }
00265
00267     void setWLNbOfSeats (const NbOfSeats_T& iWLSeats) {
00268         _wlnNbOfBookings = iWLSeats;
00269     }
00270
00272     void setGroupNbOfSeats (const NbOfSeats_T& iGroupSeats) {
00273         _groupNbOfBookings = iGroupSeats;
00274     }
00275
00277     void updateCurrentBidPrice();
00278
00279 public:
00281     // /////////// Display support methods ///////////
00286     void toStream (std::ostream& ioOut) const {
00287         ioOut << toString();
00288     }
00289
00294     void fromStream (std::istream& ioIn) {
00295     }
00296
00300     std::string toString() const;
00301
00305     const std::string describeKey() const {
00306         return _key.toString();
00307     }
```

```
00308
00312     const std::string displayVirtualClassList() const;
00313
00314
00315 public:
00316     // //////////// Business methods ////////////
00320     void updateFromReservation (const NbOfBookings_T&);
00321
00325     void addVirtualClass (const VirtualClassStruct& iVC) {
00326         _virtualClassList.push_back (iVC);
00327     }
00328
00332     void emptyVirtualClassList() {
00333         _virtualClassList.clear();
00334     }
00335
00339     void emptyBidPriceVector() {
00340         _bidPriceVector.clear();
00341     }
00342
00346     void addDemandInformation (const YieldValue_T&, const MeanValue_T&,
00347                               const StdDevValue_T&);
00348
00352     void emptyYieldLevelDemandMap() {
00353         _yieldLevelDemandMap.clear();
00354     }
00355
00356
00357 protected:
00358     // //////////// Constructors and destructors ////////////
00362     LegCabin (const Key_T&);
00366     ~LegCabin();
00367
00368 private:
00372     LegCabin();
00376     LegCabin (const LegCabin&);
00377
00378
00379
00380 protected:
00381     // //////////// Attributes ////////////
00385     Key_T _key;
00386
00390     BomAbstract* _parent;
00391
00395     HolderMap_T _holderMap;
00396
00398     CabinCapacity_T _offeredCapacity;
00399
00401     CabinCapacity_T _physicalCapacity;
00402
00404     NbOfSeats_T _soldSeat;
00405
00406     /* Committed space. */
00407     CommittedSpace_T _committedSpace;
00408
00410     Availability_T _availabilityPool;
00411
00413     Availability_T _availability;
00414
00416     BidPrice_T _currentBidPrice;
00417
```

```
00419     BidPrice_T _previousBidPrice;
00420
00422     BidPriceVector_T _bidPriceVector;
00423
00425     VirtualClassList_T _virtualClassList;
00426
00428     YieldLevelDemandMap_T _yieldLevelDemandMap;
00429
00430
00431 public:
00433     CapacityAdjustment_T _dcsRegrade;
00434
00436     AuthorizationLevel_T _au;
00437
00439     UPR_T _upr;
00440
00442     Availability_T _nav;
00443
00445     Availability_T _gav;
00446
00448     OverbookingRate_T _acp;
00449
00451     NbOfSeats_T _etb;
00452
00454     NbOfSeats_T _staffNbOfBookings;
00455
00457     NbOfSeats_T _wlNbOfBookings;
00458
00460     NbOfSeats_T _groupNbOfBookings;
00461 };
00462
00463 }
00464 #endif // __STDAIR_BOM_LEG_CABIN_HPP
00465
```

35.329 stdair/bom/LegCabinKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/LegCabinKey.hpp>
```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

35.330 LegCabinKey.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/access.hpp>
00011 // StdAir
00012 #include <stdair/basic/BasConst_Inventory.hpp>
00013 #include <stdair/bom/LegCabinKey.hpp>
00014
00015 namespace stdair {
00016
00017 // //////////////////////////////////////
00018 LegCabinKey::LegCabinKey() : _cabinCode (DEFAULT_CABIN_CODE) {
00019     assert (false);
00020 }
00021
00022 // //////////////////////////////////////
00023 LegCabinKey::LegCabinKey (const CabinCode_T& iCabinCode)
00024 : _cabinCode (iCabinCode) {
00025 }
00026
00027 // //////////////////////////////////////
00028 LegCabinKey::LegCabinKey (const LegCabinKey& iKey)
00029 : _cabinCode (iKey._cabinCode) {
00030 }
00031
00032 // //////////////////////////////////////
00033 LegCabinKey::~LegCabinKey () {
00034 }
00035
00036 // //////////////////////////////////////
00037 void LegCabinKey::toStream (std::ostream& ioOut) const {
00038     ioOut << "LegCabinKey: " << toString() << std::endl;
00039 }
00040
00041 // //////////////////////////////////////
00042 void LegCabinKey::fromStream (std::istream& ioIn) {
00043 }
00044
00045 // //////////////////////////////////////
00046 const std::string LegCabinKey::toString() const {
00047     std::ostringstream oStr;
00048     oStr << _cabinCode;
00049     return oStr.str();
00050 }
00051
00052 // //////////////////////////////////////
00053 void LegCabinKey::serialisationImplementationExport() const {
00054     std::ostringstream oStr;
00055     boost::archive::text_oarchive oa (oStr);
00056     oa << *this;
00057 }
00058
00059 // //////////////////////////////////////
00060 void LegCabinKey::serialisationImplementationImport() {

```

```

00061     std::stringstream iStr;
00062     boost::archive::text_iarchive ia (iStr);
00063     ia >> *this;
00064 }
00065
00066 // //////////////////////////////////////
00067 template<class Archive>
00068 void LegCabinKey::serialize (Archive& ioArchive,
00069                             const unsigned int iFileVersion) {
00074     ioArchive & _cabinCode;
00075 }
00076
00077 }

```

35.331 stdair/bom/LegCabinKey.hpp File Reference

```

#include <iosfwd>

#include <string>

#include <stdair/stdair_basic_types.hpp>

#include <stdair/bom/KeyAbstract.hpp>

```

Classes

- struct [stdair::LegCabinKey](#)
Key of a given leg-cabin, made of a cabin code (only).

Namespaces

- namespace [boost](#)
Forward declarations.
- namespace [boost::serialization](#)
- namespace [stdair](#)
Handle on the StdAir library context.

35.332 LegCabinKey.hpp

```

00001 #ifndef __STDAIR_BOM_LEGCABINKEY_HPP
00002 #define __STDAIR_BOM_LEGCABINKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_basic_types.hpp>
00012 #include <stdair/bom/KeyAbstract.hpp>
00013
00015 namespace boost {

```

```

00016 namespace serialization {
00017     class access;
00018 }
00019 }
00020
00021 namespace stdair {
00022
00026 struct LegCabinKey : public KeyAbstract {
00027     friend class boost::serialization::access;
00028
00029     // ////////// Constructors and destructors //////////
00030 private:
00034     LegCabinKey();
00035
00036 public:
00040     LegCabinKey (const CabinCode_T& iCabinCode);
00041
00045     LegCabinKey (const LegCabinKey&);
00046
00050     ~LegCabinKey();
00051
00052
00053 public:
00054     // ////////// Getters //////////
00056     const CabinCode_T& getCabinCode() const {
00057         return _cabinCode;
00058     }
00059
00060
00061 public:
00062     // ////////// Display support methods //////////
00068     void toStream (std::ostream& ioOut) const;
00069
00075     void fromStream (std::istream& ioIn);
00076
00086     const std::string toString() const;
00087
00088
00089 public:
00090     // ////////// (Boost) Serialisation support methods //////////
00094     template<class Archive>
00095     void serialize (Archive& ar, const unsigned int iFileVersion);
00096
00097 private:
00102     void serialisationImplementationExport() const;
00103     void serialisationImplementationImport();
00104
00105
00106 private:
00107     // ////////// Attributes //////////
00111     CabinCode_T _cabinCode;
00112 };
00113
00114 }
00115 #endif // __STDAIR_BOM_LEGCABINKEY_HPP

```

35.333 stdair/bom/LegCabinTypes.hpp File Reference

```
#include <map>
```

```
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace `stdair`
Handle on the `StdAir` library context.

Typedefs

- typedef `std::list< LegCabin * >` `stdair::LegCabinList_T`
- typedef `std::map< const MapKey_T, LegCabin * >` `stdair::LegCabinMap_T`

35.334 LegCabinTypes.hpp

```
00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_LEGCABINTYPES_HPP
00003 #define __STDAIR_BOM_LEGCABINTYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // StdAir
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016     // Forward declarations.
00017     class LegCabin;
00018
00019     typedef std::list<LegCabin*> LegCabinList_T;
00020
00021     typedef std::map<const MapKey_T, LegCabin*> LegCabinMap_T;
00022
00023 }
00024
00025 #endif // __STDAIR_BOM_LEGCABINTYPES_HPP
00026
00027
```

35.335 stdair/bom/LegDate.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/FlightDate.hpp>
```

```
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/LegDate.hpp>
```

Namespaces

- namespace `stdair`

Handle on the StdAir library context.

35.336 LegDate.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_General.hpp>
00009 #include <stdair/basic/BasConst_Inventory.hpp>
00010 #include <stdair/bom/BomManager.hpp>
00011 #include <stdair/bom/FlightDate.hpp>
00012 #include <stdair/bom/LegCabin.hpp>
00013 #include <stdair/bom/LegDate.hpp>
00014
00015 namespace stdair {
00016
00017     // //////////////////////////////////////
00018     LegDate::LegDate() : _key (DEFAULT_ORIGIN), _parent (NULL) {
00019         assert (false);
00020     }
00021
00022     // //////////////////////////////////////
00023     LegDate::LegDate (const LegDate&) : _key (DEFAULT_ORIGIN), _parent (NULL) {
00024         assert (false);
00025     }
00026
00027     // //////////////////////////////////////
00028     LegDate::LegDate (const Key_T& iKey)
00029         : _key (iKey), _parent (NULL), _distance (DEFAULT_DISTANCE_VALUE),
00030         _capacity (DEFAULT_CABIN_CAPACITY) {
00031     }
00032
00033     // //////////////////////////////////////
00034     LegDate::~LegDate () {
00035     }
00036
00037     // //////////////////////////////////////
00038     const AirlineCode_T& LegDate::getAirlineCode() const {
00039         const FlightDate* lFlightDate_ptr =
00040             static_cast<const FlightDate*> (getParent());
00041         assert (lFlightDate_ptr != NULL);
00042         return lFlightDate_ptr->getAirlineCode();
00043     }
00044
00045     // //////////////////////////////////////
00046     std::string LegDate::toString() const {
00047         std::ostringstream oStr;
```

```

00048     oStr << describeKey();
00049     return oStr.str();
00050 }
00051
00052 // //////////////////////////////////////
00053 LegCabin* LegDate::getLegCabin (const std::string& iLegCabinKeyStr) const {
00054     LegCabin* oLegCabin_ptr =
00055         BomManager::getObjectPtr<LegCabin> (*this, iLegCabinKeyStr);
00056     return oLegCabin_ptr;
00057 }
00058
00059 // //////////////////////////////////////
00060 LegCabin* LegDate::getLegCabin (const LegCabinKey& iLegCabinKey) const {
00061     return getLegCabin (iLegCabinKey.toString());
00062 }
00063
00064 // //////////////////////////////////////
00065 const Duration_T LegDate::getTimeOffset() const {
00066     // TimeOffset = (OffTime - BoardingTime) + (OffDate - BoardingDate) * 24
00067     //              - ElapsedTime
00068     Duration_T oTimeOffset = (_offTime - _boardingTime);
00069
00070     const DateOffset_T& lDateOffset = getDateOffset();
00071
00072     const Duration_T lDateOffsetInHours (lDateOffset.days() * 24, 0, 0);
00073
00074     oTimeOffset += lDateOffsetInHours - _elapsedTime;
00075
00076     return oTimeOffset;
00077 }
00078
00079 // //////////////////////////////////////
00080 void LegDate::setElapsedTime (const Duration_T& iElapsedTime) {
00081     // Set Elapsed time
00082     _elapsedTime = iElapsedTime;
00083
00084     // Update distance according to the mean plane speed
00085     updateDistanceFromElapsedTime();
00086 }
00087
00088 // //////////////////////////////////////
00089 void LegDate::updateDistanceFromElapsedTime() {
00090     //
00091     const double lElapseInHours =
00092         static_cast<const double> (_elapsedTime.hours());
00093
00094     // Normally, Distance_T is an unsigned long int
00095     const Distance_T lDistance =
00096         static_cast<const Distance_T> (DEFAULT_FLIGHT_SPEED * lElapseInHours);
00097
00098     _distance = lDistance;
00099 }
00100
00101 }
00102

```

35.337 stdair/bom/LegDate.hpp File Reference

```
#include <iosfwd>
```

```
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/LegDateKey.hpp>
#include <stdair/bom/LegDateTypes.hpp>
```

Classes

- class [stdair::LegDate](#)

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.338 LegDate.hpp

```
00001 #ifndef __STDAIR_BOM_LEGDATE_HPP
00002 #define __STDAIR_BOM_LEGDATE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/bom/BomAbstract.hpp>
00013 #include <stdair/bom/LegDateKey.hpp>
00014 #include <stdair/bom/LegDateTypes.hpp>
00015
00016 namespace stdair {
00017
00019     struct LegCabinKey;
00020     class LegCabin;
00021
00025     class LegDate : public BomAbstract {
00026     template <typename BOM> friend class FacBom;
00027     friend class FacBomManager;
00028
00029     public:
00030     // //////////// Type definitions //////////
00032     typedef LegDateKey Key_T;
00033
00034
00035     public:
00036     // //////////// Getters //////////
00038     const Key_T& getKey() const {
00039         return _key;
00040     }
00041
00043     BomAbstract* const getParent() const {
```

```
00044         return _parent;
00045     }
00046
00048     const AirportCode_T& getBoardingPoint() const {
00049         return _key.getBoardingPoint();
00050     }
00051
00059     const AirlineCode_T& getAirlineCode() const;
00060
00064     const HolderMap_T& getHolderMap() const {
00065         return _holderMap;
00066     }
00067
00078     LegCabin* getLegCabin (const std::string& iLegCabinKeyStr) const;
00079
00090     LegCabin* getLegCabin (const LegCabinKey&) const;
00091
00093     const AirportCode_T& getOffPoint() const {
00094         return _offPoint;
00095     }
00096
00098     const Date_T& getBoardingDate() const {
00099         return _boardingDate;
00100     }
00101
00103     const Duration_T& getBoardingTime() const {
00104         return _boardingTime;
00105     }
00106
00108     const Date_T& getOffDate() const {
00109         return _offDate;
00110     }
00111
00113     const Duration_T& getOffTime() const {
00114         return _offTime;
00115     }
00116
00118     const Duration_T& getElapsedTime() const {
00119         return _elapsedTime;
00120     }
00121
00123     const Distance_T& getDistance() const {
00124         return _distance;
00125     }
00126
00128     const CabinCapacity_T& getCapacity() const {
00129         return _capacity;
00130     }
00131
00133     const DateOffset_T getDateOffset() const {
00134         return _offDate - _boardingDate;
00135     }
00136
00141     const Duration_T getTimeOffset() const;
00142
00143
00144 public:
00145     // ////////// Setters //////////
00147     void setOffPoint (const AirportCode_T& iOffPoint) {
00148         _offPoint = iOffPoint;
00149     }
00150
```

```
00152     void setBoardingDate (const Date_T& iBoardingDate) {
00153         _boardingDate = iBoardingDate;
00154     }
00155
00157     void setBoardingTime (const Duration_T& iBoardingTime) {
00158         _boardingTime = iBoardingTime;
00159     }
00160
00162     void setOffDate (const Date_T& iOffDate) {
00163         _offDate = iOffDate;
00164     }
00165
00167     void setOffTime (const Duration_T& iOffTime) {
00168         _offTime = iOffTime;
00169     }
00170
00172     void setElapsedTime (const Duration_T&);
00173
00174     private:
00176         void updateDistanceFromElapsedTime();
00177
00178
00179     public:
00180         // //////////// Display support methods ////////////
00183         void toStream (std::ostream& ioOut) const {
00184             ioOut << toString();
00185         }
00186
00189         void fromStream (std::istream& ioIn) {
00190         }
00191
00193         std::string toString() const;
00194
00196         const std::string describeKey() const {
00197             return _key.toString();
00198         }
00199
00200
00201     protected:
00202         // //////////// Constructors and destructors ////////////
00204         LegDate (const Key_T&);
00206         virtual ~LegDate();
00207
00208     private:
00210         LegDate();
00212         LegDate (const LegDate&);
00213
00214
00215     protected:
00216         // //////////// Attributes ////////////
00218         Key_T _key;
00219
00221         BomAbstract* _parent;
00222
00224         HolderMap_T _holderMap;
00225
00227         AirportCode_T _offPoint;
00228
00230         Date_T _boardingDate;
00231
00233         Duration_T _boardingTime;
00234
```

```

00236     Date_T _offDate;
00237
00239     Duration_T _offTime;
00240
00242     Duration_T _elapsedTime;
00243
00245     Distance_T _distance;
00246
00248     CabinCapacity_T _capacity;
00249 };
00250
00251 }
00252 #endif // __STDAIR_BOM_LEGDATE_HPP
00253

```

35.339 stdair/bom/LegDateKey.cpp File Reference

```

#include <cassert>

#include <sstream>

#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/LegDateKey.hpp>

```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.340 LegDateKey.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_Inventory.hpp>
00009 #include <stdair/bom/LegDateKey.hpp>
00010
00011 namespace stdair {
00012
00013 // //////////////////////////////////////
00014 LegDateKey::LegDateKey() : _boardingPoint (DEFAULT_ORIGIN) {
00015     assert (false);
00016 }
00017
00018 // //////////////////////////////////////
00019 LegDateKey::LegDateKey (const AirportCode_T& iBoardingPoint)
00020 : _boardingPoint (iBoardingPoint) {
00021 }
00022
00023 // //////////////////////////////////////
00024 LegDateKey::LegDateKey (const LegDateKey& iKey)

```

```

00025     : _boardingPoint (iKey._boardingPoint) {
00026     }
00027
00028     // //////////////////////////////////////
00029     LegDateKey::~LegDateKey () {
00030     }
00031
00032     // //////////////////////////////////////
00033     void LegDateKey::toStream (std::ostream& ioOut) const {
00034         ioOut << "LegDateKey: " << toString();
00035     }
00036
00037     // //////////////////////////////////////
00038     void LegDateKey::fromStream (std::istream& ioIn) {
00039     }
00040
00041     // //////////////////////////////////////
00042     const std::string LegDateKey::toString() const {
00043         std::ostringstream oStr;
00044         oStr << _boardingPoint;
00045         return oStr.str();
00046     }
00047
00048 }

```

35.341 stdair/bom/LegDateKey.hpp File Reference

```

#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>

```

Classes

- struct [stdair::LegDateKey](#)

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.342 LegDateKey.hpp

```

00001 #ifndef __STDAIR_BOM_LEGDATEKEY_HPP
00002 #define __STDAIR_BOM_LEGDATEKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009 #include <stdair/bom/KeyAbstract.hpp>
00010
00011 namespace stdair {
00012

```

```

00016 struct LegDateKey : public KeyAbstract {
00017
00018     // ////////// Constructors and destructors //////////
00019 private:
00021     LegDateKey();
00022
00023 public:
00025     LegDateKey (const AirportCode_T& iBoardingPoint);
00027     LegDateKey (const LegDateKey&);
00029     ~LegDateKey();
00030
00031
00032     // ////////// Getters //////////
00034     const AirportCode_T& getBoardingPoint() const {
00035         return _boardingPoint;
00036     }
00037
00038
00039     // ////////// Display support methods //////////
00042     void toStream (std::ostream& ioOut) const;
00043
00046     void fromStream (std::istream& ioIn);
00047
00053     const std::string toString() const;
00054
00055 private:
00057     // ////////////////// Attributes //////////////////
00059     AirportCode_T _boardingPoint;
00060 };
00061
00062 }
00063 #endif // __STDAIR_BOM_LEGDATEKEY_HPP

```

35.343 stdair/bom/LegDateTypes.hpp File Reference

```

#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>

```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

Typedefs

- typedef std::list< LegDate * > [stdair::LegDateList_T](#)
- typedef std::map< const MapKey_T, LegDate * > [stdair::LegDateMap_T](#)

35.344 LegDateTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_LEGDATE_Types_HPP
00003 #define __STDAIR_BOM_LEGDATE_Types_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // StdAir
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016     // Forward declarations.
00017     class LegDate;
00018
00019     typedef std::list<LegDate*> LegDateList_T;
00020
00021     typedef std::map<const MapKey_T, LegDate*> LegDateMap_T;
00022 }
00023 #endif // __STDAIR_BOM_LEGDATE_Types_HPP
00024

```

35.345 stdair/bom/OnDDate.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/OnDDate.hpp>

```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

35.346 OnDDate.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>

```

```

00007 // StdAir
00008 #include <stdair/basic/BasConst_Inventory.hpp>
00009 #include <stdair/basic/BasConst_General.hpp>
00010 #include <stdair/bom/BomManager.hpp>
00011 #include <stdair/bom/Inventory.hpp>
00012 #include <stdair/bom/OnDDate.hpp>
00013
00014 namespace stdair {
00015
00016 // //////////////////////////////////////
00017 OnDDate::OnDDate()
00018 : _key (DEFAULT_OND_STRING_LIST), _parent (NULL) {
00019     assert (false);
00020 }
00021
00022 // //////////////////////////////////////
00023 OnDDate::OnDDate (const OnDDate& iOnDDate)
00024 : _key (iOnDDate.getKey()), _parent (NULL) {
00025     assert (false);
00026 }
00027
00028 // //////////////////////////////////////
00029 OnDDate::OnDDate (const Key_T& iKey)
00030 : _key (iKey), _parent (NULL) {
00031 }
00032
00033 // //////////////////////////////////////
00034 OnDDate::~OnDDate() {
00035 }
00036
00037 // //////////////////////////////////////
00038 std::string OnDDate::toString() const {
00039     std::ostringstream oStr;
00040     oStr << describeKey();
00041     return oStr.str();
00042 }
00043
00044 // //////////////////////////////////////
00045 const AirlineCode_T& OnDDate::getAirlineCode() const {
00046     const Inventory* lInventory_ptr =
00047         static_cast<const Inventory*> (getParent());
00048     assert (lInventory_ptr != NULL);
00049     return lInventory_ptr->getAirlineCode();
00050 }
00051
00052 // //////////////////////////////////////
00053 void OnDDate::
00054 setDemandInformation (const CabinClassPairList_T& iCabinClassPairList,
00055                      const YieldDemandPair_T& iYieldDemandPair) {
00056     std::ostringstream oStr;
00057     for(CabinClassPairList_T::const_iterator itCCP = iCabinClassPairList.begin();
00058
00059         itCCP != iCabinClassPairList.end(); ++itCCP) {
00059         oStr << itCCP->first << ":" << itCCP->second << ";";
00060     }
00061     std::string lCabinClassPath = oStr.str();
00062     StringDemandStructMap_T::iterator it =
00063         _classPathDemandMap.find(lCabinClassPath);
00064     if (it == _classPathDemandMap.end()) {
00065         const StringDemandStructPair_T lPairStringDemandChar (lCabinClassPath,
00066                                                                iYieldDemandPair);
00067         _classPathDemandMap.insert (lPairStringDemandChar);

```

```

00068         const StringCabinClassPair_T lStringCabinClassPair (lCabinClassPath,
00069                                                             iCabinClassPairList);
00070         _stringCabinClassPairListMap.insert (lStringCabinClassPair);
00071     } else {
00072         it->second = iYieldDemandPair;
00073     }
00074 }
00075
00076 // //////////////////////////////////////
00077 void OnDDate::setTotalForecast (const CabinCode_T& iCabinCode,
00078                               const WTPDemandPair_T& iWTPDemandPair) {
00079
00080     CabinForecastMap_T::iterator it =
00081         _cabinForecastMap.find (iCabinCode);
00082     if (it == _cabinForecastMap.end()) {
00083         const CabinForecastPair_T lPairCabinForecastChar (iCabinCode,
00084                                                         iWTPDemandPair);
00085         _cabinForecastMap.insert (lPairCabinForecastChar);
00086     } else {
00087         assert (false);
00088     }
00089 }
00090
00091 }

```

35.347 stdair/bom/OnDDate.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/stdair_rm_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/OnDDateKey.hpp>
#include <stdair/bom/OnDDateTypes.hpp>

```

Classes

- class [stdair::OnDDate](#)
Class representing the actual attributes for an airline flight-date.

Namespaces

- namespace [boost](#)
Forward declarations.

- namespace `boost::serialization`
- namespace `stdair`

Handle on the StdAir library context.

35.348 OnDDate.hpp

```

00001 #ifndef __STDAIR_BOM_ONDDATE_HPP
00002 #define __STDAIR_BOM_ONDDATE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/stdair_maths_types.hpp>
00013 #include <stdair/stdair_basic_types.hpp>
00014 #include <stdair/stdair_demand_types.hpp>
00015 #include <stdair/stdair_rm_types.hpp>
00016 #include <stdair/bom/BomAbstract.hpp>
00017 #include <stdair/bom/OnDDateKey.hpp>
00018 #include <stdair/bom/OnDDateTypes.hpp>
00019
00021 namespace boost {
00022     namespace serialization {
00023         class access;
00024     }
00025 }
00026
00027 namespace stdair {
00028
00033     class OnDDate : public BomAbstract {
00034     public:
00035         template <typename BOM> friend class FacBom;
00036         friend class FacBomManager;
00037         friend class boost::serialization::access;
00038
00039         // ////////////////////////////////// Type definitions //////////////////////////////////
00040         typedef OnDDateKey Key_T;
00041
00042     public:
00043         // ////////////////////////////////// Getters //////////////////////////////////
00044         const Key_T& getKey() const {
00045             return _key;
00046         }
00047
00048         BomAbstract* const getParent() const {
00049             return _parent;
00050         }
00051
00052         const AirlineCode_T& getAirlineCode() const;
00053
00054         const stdair::Date_T getDate() const {
00055             return _key.getDate();
00056         }
00057     }
00058 }

```

```

00074     const stdair::AirportCode_T getOrigin() const {
00075         return _key.getOrigin();
00076     }
00077
00079     const stdair::AirportCode_T getDestination() const {
00080         return _key.getDestination();
00081     }
00082
00086     const HolderMap_T& getHolderMap() const {
00087         return _holderMap;
00088     }
00089
00093     const StringDemandStructMap_T& getDemandInfoMap () const {
00094         return _classPathDemandMap;
00095     }
00096
00100     const CabinForecastMap_T& getTotalForecastMap () const {
00101         return _cabinForecastMap;
00102     }
00103
00107     const WTPDemandPair_T& getTotalForecast (const CabinCode_T& iCC) const {
00108         assert (_cabinForecastMap.find(iCC)!=_cabinForecastMap.end());
00109         return _cabinForecastMap.find(iCC)->second;
00110     }
00111
00115     const CabinClassPairList_T& getCabinClassPairList (const std::string& iStr) c
const {
00116         assert (_stringCabinClassPairListMap.find(iStr)!=
_stringCabinClassPairListMap.end());
00117         return _stringCabinClassPairListMap.find(iStr)->second;
00118     }
00119
00123     const short getNbOfSegments () const {
00124         return _key.getNbOfSegments();
00125     }
00126
00127 public:
00128     // //////////// Setters ////////////
00130     void setDemandInformation (const CabinClassPairList_T&,
00131                             const YieldDemandPair_T&);
00132
00133
00135     void setTotalForecast (const CabinCode_T&,
00136                         const WTPDemandPair_T&);
00137
00138
00139 public:
00140     // //////////// Display support methods ////////////
00146     void toStream (std::ostream& ioOut) const {
00147         ioOut << toString();
00148     }
00149
00155     void fromStream (std::istream& ioIn) {
00156     }
00157
00161     std::string toString() const;
00162
00166     const std::string describeKey() const {
00167         return _key.toString();
00168     }
00169
00170

```

```

00171 public:
00172     // //////////// (Boost) Serialisation support methods ////////////
00176     template<class Archive>
00177     void serialize (Archive& ar, const unsigned int iFileVersion);
00178
00179 private:
00184     void serialisationImplementation();
00185
00186
00187 protected:
00188     // //////////// Constructors and destructors ////////////
00192     OnDDate (const Key_T&);
00193
00197     virtual ~OnDDate();
00198
00199 private:
00203     OnDDate();
00204
00208     OnDDate (const OnDDate&);
00209
00210
00211 protected:
00212     // //////////// Attributes ////////////
00216     Key_T _key;
00217
00221     BomAbstract* _parent;
00222
00226     HolderMap_T _holderMap;
00227
00231     StringDemandStructMap_T _classPathDemandMap;
00232
00236     StringCabinClassPairListMap_T _stringCabinClassPairListMap;
00237
00241     CabinForecastMap_T _cabinForecastMap;
00242 };
00243
00244 }
00245 #endif // __STDAIR_BOM_ONDDATE_HPP

```

35.349 stdair/bom/OnDDateKey.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <boost/date_time/gregorian/formatters.hpp>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/OnDDateKey.hpp>
#include <stdair/bom/BomKeyManager.hpp>

```

```
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/BomDisplay.hpp>
```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

Functions

- template void `stdair::OnDDateKey::serialize< ba::text_oarchive >` (ba::text_oarchive &, unsigned int)
- template void `stdair::OnDDateKey::serialize< ba::text_iarchive >` (ba::text_iarchive &, unsigned int)

35.350 OnDDateKey.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost Date-Time
00008 #include <boost/date_time/gregorian/formatters.hpp>
00009 // Boost.Serialization
00010 #include <boost/archive/text_iarchive.hpp>
00011 #include <boost/archive/text_oarchive.hpp>
00012 #include <boost/serialization/access.hpp>
00013 // StdAir
00014 #include <stdair/basic/BasConst_Inventory.hpp>
00015 #include <stdair/basic/BasConst_BomDisplay.hpp>
00016 #include <stdair/basic/BasConst_General.hpp>
00017 #include <stdair/bom/OnDDateKey.hpp>
00018 #include <stdair/bom/BomKeyManager.hpp>
00019 #include <stdair/bom/Inventory.hpp>
00020 #include <stdair/bom/FlightDate.hpp>
00021 #include <stdair/bom/SegmentDate.hpp>
00022 #include <stdair/bom/BomDisplay.hpp>
00023
00024 namespace stdair {
00025
00026 // //////////////////////////////////////
00027 OnDDateKey::OnDDateKey()
00028     : _OnDStringList (DEFAULT_OND_STRING_LIST) {
00029     assert (false);
00030 }
00031
00032 // //////////////////////////////////////
00033 OnDDateKey::OnDDateKey (const OnDStringList_T& iOnDStringList)
00034     : _OnDStringList (iOnDStringList) {
```

```

00035     }
00036
00037     // //////////////////////////////////////
00038     OnDDateKey::OnDDateKey (const OnDDateKey& iKey)
00039         : _OnDStringList (iKey._OnDStringList) {
00040     }
00041
00042     // //////////////////////////////////////
00043     OnDDateKey::~OnDDateKey () {
00044     }
00045
00046     // //////////////////////////////////////
00047     const Date_T OnDDateKey::getDate() const {
00048         assert(_OnDStringList.empty() == false);
00049         const OnDString_T& lFrontOnDString = _OnDStringList.front();
00050         return BomKeyManager::extractFlightDateKey (lFrontOnDString).
getDepartureDate();
00051     }
00052
00053     // //////////////////////////////////////
00054     const AirportCode_T OnDDateKey::getOrigin() const {
00055         assert(_OnDStringList.empty() == false);
00056         const OnDString_T& lFrontOnDString = _OnDStringList.front();
00057         return BomKeyManager::extractSegmentDateKey (lFrontOnDString).
getBoardingPoint();
00058     }
00059
00060     // //////////////////////////////////////
00061     const AirportCode_T OnDDateKey::getDestination() const {
00062         assert(_OnDStringList.empty() == false);
00063         const OnDString_T& lLastOnDString = _OnDStringList.back();
00064         return BomKeyManager::extractSegmentDateKey (lLastOnDString).getOffPoint();
00065     }
00066
00067     // //////////////////////////////////////
00068     void OnDDateKey::toStream (std::ostream& ioOut) const {
00069         ioOut << "OnDDateKey: " << toString();
00070     }
00071
00072     // //////////////////////////////////////
00073     void OnDDateKey::fromStream (std::istream& ioIn) {
00074     }
00075
00076     // //////////////////////////////////////
00077     const std::string OnDDateKey::toString() const {
00078         std::ostringstream oStr;
00079         for (OnDStringList_T::const_iterator itOnDString = _OnDStringList.begin();
00080              itOnDString != _OnDStringList.end(); ++itOnDString){
00081             oStr << *itOnDString << " ";
00082         }
00083         return oStr.str();
00084     }
00085
00086     // //////////////////////////////////////
00087     void OnDDateKey::serialisationImplementationExport() const {
00088         std::ostringstream oStr;
00089         boost::archive::text_oarchive oa (oStr);
00090         oa << *this;
00091     }
00092
00093     // //////////////////////////////////////
00094     void OnDDateKey::serialisationImplementationImport() {

```

```

00095     std::stringstream iStr;
00096     boost::archive::text_iarchive ia (iStr);
00097     ia >> *this;
00098 }
00099
00100 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00101 template<class Archive>
00102 void OnDDateKey::serialize (Archive& ioArchive,
00103                             const unsigned int iFileVersion) {
00104 }
00105
00106 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00107 // Explicit template instantiation
00108 namespace ba = boost::archive;
00109 template void OnDDateKey::serialize<ba::text_oarchive> (ba::text_oarchive&,
00110                                                         unsigned int);
00111 template void OnDDateKey::serialize<ba::text_iarchive> (ba::text_iarchive&,
00112                                                         unsigned int);
00113 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00114 }
00115

```

35.351 stdair/bom/OnDDateKey.hpp File Reference

```

#include <iosfwd>

#include <string>

#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>

```

Classes

- struct [stdair::OnDDateKey](#)

Key of a given O&D-date, made of a list of OnD strings. a OnD string contains the airline code, the flight number, the date and the segment (origin and destination).

Namespaces

- namespace [stdair](#)

Handle on the StdAir library context.

35.352 OnDDateKey.hpp

```

00001 #ifndef __STDAIR_BOM_ONDDATEKEY_HPP
00002 #define __STDAIR_BOM_ONDDATEKEY_HPP
00003
00004 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00005 // Import section

```

```

00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_basic_types.hpp>
00012 #include <stdair/stdair_demand_types.hpp>
00013 #include <stdair/stdair_date_time_types.hpp>
00014 #include <stdair/bom/KeyAbstract.hpp>
00015
00016 namespace stdair {
00017
00023     struct OnDDateKey : public KeyAbstract {
00024         friend class boost::serialization::access;
00025
00026         // ////////////////////////////////// Constructors and destructors //////////////////////////////////
00027     private:
00031         OnDDateKey();
00032
00033     public:
00037         OnDDateKey (const OnDStringList_T&);
00038
00042         OnDDateKey (const OnDDateKey&);
00043
00047         ~OnDDateKey();
00048
00049
00050     public:
00051         // ////////////////////////////////// Getters //////////////////////////////////
00055         const Date_T getDate() const;
00056
00060         const AirportCode_T getOrigin() const;
00061
00065         const AirportCode_T getDestination() const;
00066
00070         const short getNbOfSegments () const {
00071             return _OnDStringList.size();
00072         }
00073
00074     public:
00075         // ////////////////////////////////// Display support methods //////////////////////////////////
00081         void toStream (std::ostream& ioOut) const;
00082
00088         void fromStream (std::istream& ioIn);
00089
00099         const std::string toString() const;
00100
00101
00102     public:
00103         // ////////////////////////////////// (Boost) Serialisation support methods //////////////////////////////////
00107         template<class Archive>
00108         void serialize (Archive& ar, const unsigned int iFileVersion);
00109
00110     private:
00115         void serialisationImplementationExport() const;
00116         void serialisationImplementationImport();
00117
00118
00119     private:
00120         // ////////////////////////////////// Attributes //////////////////////////////////
00121         OnDStringList_T _OnDStringList;
00122

```

```

00123     };
00124
00125 }
00126 #endif // __STDAIR_BOM_ONDDATEKEY_HPP

```

35.353 stdair/bom/OnDDateTypes.hpp File Reference

```

#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/stdair_demand_types.hpp>

```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

Typedefs

- typedef std::list< OnDDate * > [stdair::OnDDateList_T](#)
- typedef std::map< const MapKey_T, OnDDate * > [stdair::OnDDateMap_T](#)
- typedef std::pair< std::string, YieldDemandPair_T > [stdair::StringDemandStructPair_T](#)
- typedef std::map< std::string, YieldDemandPair_T > [stdair::StringDemandStructMap_T](#)
- typedef std::map< std::string, CabinClassPairList_T > [stdair::StringCabinClassPairListMap_T](#)
- typedef std::pair< std::string, CabinClassPairList_T > [stdair::StringCabinClassPair_T](#)
- typedef std::map< CabinCode_T, WTPDemandPair_T > [stdair::CabinForecastMap_T](#)
- typedef std::pair< CabinCode_T, WTPDemandPair_T > [stdair::CabinForecastPair_T](#)

35.354 OnDDateTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_ONDDATETYPES_HPP
00003 #define __STDAIR_BOM_ONDDATETYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <map>

```

```

00010 #include <list>
00011 // Stdair
00012 #include <stdair/bom/key_types.hpp>
00013 #include <stdair/stdair_maths_types.hpp>
00014 #include <stdair/stdair_demand_types.hpp>
00015
00016 namespace stdair {
00017
00018     // Forward declarations.
00019     class OnDDate;
00020
00021     typedef std::list<OnDDate*> OnDDateList_T;
00022
00023     typedef std::map<const MapKey_T, OnDDate*> OnDDateMap_T;
00024
00025     typedef std::pair<std::string, YieldDemandPair_T> StringDemandStructPair_T;
00026     typedef std::map<std::string, YieldDemandPair_T> StringDemandStructMap_T;
00027
00028     typedef std::map<std::string, CabinClassPairList_T>
00029     StringCabinClassPairListMap_T;
00030     typedef std::pair<std::string, CabinClassPairList_T> StringCabinClassPair_T;
00031
00032     typedef std::map<CabinCode_T, WTPDemandPair_T> CabinForecastMap_T;
00033     typedef std::pair<CabinCode_T, WTPDemandPair_T> CabinForecastPair_T;
00034
00035 }
00036 #endif // __STDAIR_BOM_ONDDATETYPES_HPP

```

35.355 stdair/bom/OptimisationNotificationStruct.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/bom/OptimisationNotificationStruct.hpp>

```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

35.356 OptimisationNotificationStruct.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/bom/OptimisationNotificationStruct.hpp>
00009
00010 namespace stdair {
00011
00012     // //////////////////////////////////////
00013     OptimisationNotificationStruct::OptimisationNotificationStruct()

```

```

00014     : _partySize (0), _stayDuration (0), _wtp (0.0), _valueOfTime (0.0) {
00015     assert (false);
00016 }
00017
00018 // //////////////////////////////////////
00019 OptimisationNotificationStruct::
00020 OptimisationNotificationStruct (const OptimisationNotificationStruct& iOptimisa
tionNotification)
00021     : _origin (iOptimisationNotification._origin),
00022       _destination (iOptimisationNotification._destination),
00023       _pos (iOptimisationNotification._pos),
00024       _preferredDepartureDate (iOptimisationNotification._preferredDepartureDate)
00025 ,
00026       _notificationDateTime (iOptimisationNotification._notificationDateTime),
00027       _preferredCabin (iOptimisationNotification._preferredCabin),
00028       _partySize (iOptimisationNotification._partySize),
00029       _channel (iOptimisationNotification._channel),
00030       _tripType (iOptimisationNotification._tripType),
00031       _stayDuration (iOptimisationNotification._stayDuration),
00032       _frequentFlyerType (iOptimisationNotification._frequentFlyerType),
00033       _preferredDepartureTime (iOptimisationNotification._preferredDepartureTime)
00034 ,
00035       _wtp (iOptimisationNotification._wtp),
00036       _valueOfTime (iOptimisationNotification._valueOfTime) {
00037 }
00038 // //////////////////////////////////////
00039 OptimisationNotificationStruct::
00040 OptimisationNotificationStruct (const AirportCode_T& iOrigin,
00041                                const AirportCode_T& iDestination,
00042                                const CityCode_T& iPOS,
00043                                const Date_T& iDepartureDate,
00044                                const DateTime_T& iNotificationDateTime,
00045                                const CabinCode_T& iPreferredCabin,
00046                                const NbOfSeats_T& iPartySize,
00047                                const ChannelLabel_T& iChannel,
00048                                const TripType_T& iTripType,
00049                                const DayDuration_T& iStayDuration,
00050                                const FrequentFlyer_T& iFrequentFlyerType,
00051                                const Duration_T& iPreferredDepartureTime,
00052                                const WTP_T& iWTP,
00053                                const PriceValue_T& iValueOfTime)
00054     : _origin (iOrigin), _destination (iDestination),
00055       _pos (iPOS), _preferredDepartureDate (iDepartureDate),
00056       _notificationDateTime (iNotificationDateTime),
00057       _preferredCabin (iPreferredCabin), _partySize (iPartySize),
00058       _channel (iChannel), _tripType (iTripType),
00059       _stayDuration (iStayDuration), _frequentFlyerType (iFrequentFlyerType),
00060       _preferredDepartureTime (iPreferredDepartureTime), _wtp (iWTP),
00061       _valueOfTime (iValueOfTime) {
00062 }
00063 // //////////////////////////////////////
00064 OptimisationNotificationStruct::~OptimisationNotificationStruct() {
00065 }
00066 // //////////////////////////////////////
00067 void OptimisationNotificationStruct::toStream (std::ostream& ioOut) const {
00068     ioOut << describe();
00069 }
00070 // //////////////////////////////////////
00071
00072

```

```

00073 void OptimisationNotificationStruct::fromStream (std::istream& ioIn) {
00074 }
00075
00076 // //////////////////////////////////////
00077 const std::string OptimisationNotificationStruct::describe() const {
00078     std::ostringstream ostr;
00079     ostr << "At " << _notificationDateTime
00080         << ", for (" << _pos << ") " << _origin << "-" << _destination
00081         << " " << _preferredDepartureDate << " " << _preferredCabin
00082         << " " << _partySize << " " << _channel << " " << _tripType
00083         << " " << _stayDuration << " " << _frequentFlyerType
00084         << " " << _preferredDepartureTime << " " << _wtp
00085         << " " << _valueOfTime;
00086     return ostr.str();
00087 }
00088
00089 }

```

35.357 stdair/bom/OptimisationNotificationStruct.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/OptimisationNotificationTypes.hpp>

```

Classes

- struct [stdair::OptimisationNotificationStruct](#)

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.358 OptimisationNotificationStruct.hpp

```

00001 #ifndef __STDAIR_BOM_OPTIMISATIONNOTIFICATIONSTRUCT_HPP
00002 #define __STDAIR_BOM_OPTIMISATIONNOTIFICATIONSTRUCT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/stdair_demand_types.hpp>

```

```
00013 #include <stdair/basic/StructAbstract.hpp>
00014 #include <stdair/bom/OptimisationNotificationTypes.hpp>
00015
00016 namespace stdair {
00017
00018     struct OptimisationNotificationStruct : public StructAbstract {
00019     public:
00020         // ////////////////////////////////// Getters //////////////////////////////////
00021         const AirportCode_T& getOrigin() const {
00022             return _origin;
00023         }
00024
00025         const AirportCode_T& getDestination() const {
00026             return _destination;
00027         }
00028
00029         const CityCode_T& getPOS() const {
00030             return _pos;
00031         }
00032
00033         const Date_T& getPreferredDepartureDate() const {
00034             return _preferredDepartureDate;
00035         }
00036
00037         const DateTime_T& getNotificationDateTime() const {
00038             return _notificationDateTime;
00039         }
00040
00041         const CabinCode_T& getPreferredCabin() const {
00042             return _preferredCabin;
00043         }
00044
00045         const NbOfSeats_T& getPartySize() const {
00046             return _partySize;
00047         }
00048
00049         const ChannelLabel_T& getOptimisationChannel() const {
00050             return _channel;
00051         }
00052
00053         const TripType_T& getTripType() const {
00054             return _tripType;
00055         }
00056
00057         const DayDuration_T& getStayDuration() const {
00058             return _stayDuration;
00059         }
00060
00061         const FrequentFlyer_T& getFrequentFlyerType() const {
00062             return _frequentFlyerType;
00063         }
00064
00065         const Duration_T& getPreferredDepartureTime() const {
00066             return _preferredDepartureTime;
00067         }
00068
00069         const WTP_T& getWTP() const {
00070             return _wtp;
00071         }
00072
00073         const PriceValue_T& getValueOfTime () const {
00074             return _valueOfTime;
00075         }
00076
00077     };
00078 }
```

```

00090     }
00091
00092     // //////////// Display support method ////////////
00095     void toStream (std::ostream& ioOut) const;
00096
00099     void fromStream (std::istream& ioIn);
00100
00102     const std::string describe() const;
00103
00104
00105     // //////////// Constructors and Destructors ////////////
00106 public:
00108     OptimisationNotificationStruct (const AirportCode_T& iOrigin,
00109                                     const AirportCode_T& iDestination,
00110                                     const CityCode_T& iPOS,
00111                                     const Date_T& iDepartureDate,
00112                                     const DateTime_T& iNotificationDateTime,
00113                                     const CabinCode_T& iPreferredCabin,
00114                                     const NbOfSeats_T& iPartySize,
00115                                     const ChannelLabel_T& iChannel,
00116                                     const TripType_T& iTripType,
00117                                     const DayDuration_T& iStayDuration,
00118                                     const FrequentFlyer_T& iFrequentFlyerType,
00119                                     const Duration_T& iPreferredDepartureTime,
00120                                     const WTP_T& iWTP,
00121                                     const PriceValue_T& iValueOfTime);
00122
00124     OptimisationNotificationStruct (const OptimisationNotificationStruct&);
00125
00126 private:
00129     OptimisationNotificationStruct ();
00130
00131 public:
00133     ~OptimisationNotificationStruct ();
00134
00135
00136 private:
00137     // //////////// Attributes ////////////
00139     const AirportCode_T _origin;
00140
00142     const AirportCode_T _destination;
00143
00145     const CityCode_T _pos;
00146
00148     const Date_T _preferredDepartureDate;
00149
00151     const DateTime_T _notificationDateTime;
00152
00154     const CabinCode_T _preferredCabin;
00155
00157     const NbOfSeats_T _partySize;
00158
00160     const ChannelLabel_T _channel;
00161
00164     const TripType_T _tripType;
00165
00167     const DayDuration_T _stayDuration;
00168
00170     const FrequentFlyer_T _frequentFlyerType;
00171
00173     const Duration_T _preferredDepartureTime;
00174

```

```

00176     const WTP_T _wtp;
00177
00179     const PriceValue_T _valueOfTime;
00180 };
00181
00182 }
00183 #endif // __STDAIR_BOM_OPTIMISATIONNOTIFICATIONSTRUCT_HPP

```

35.359 stdair/bom/OptimisationNotificationTypes.hpp File Reference

```
#include <boost/shared_ptr.hpp>
```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

Typedefs

- typedef boost::shared_ptr< OptimisationNotificationStruct > [stdair::OptimisationNotificationPtr_T](#)

35.360 OptimisationNotificationTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_OPTIMISATIONNOTIFICATIONTYPES_HPP
00003 #define __STDAIR_BOM_OPTIMISATIONNOTIFICATIONTYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // Boost
00009 #include <boost/shared_ptr.hpp>
00010
00011 namespace stdair {
00012
00013     // Forward declarations
00014     struct OptimisationNotificationStruct;
00015
00016     // ////////////////////////////////// Type definitions //////////////////////////////////
00017     typedef boost::
00018         shared_ptr<OptimisationNotificationStruct> OptimisationNotificationPtr_T;
00019
00020 }
00021
00022 #endif // __STDAIR_BOM_OPTIMISATIONNOTIFICATIONTYPES_HPP
00023

```

35.361 stdair/bom/ParsedKey.cpp File Reference

```

#include <cassert>

#include <sstream>

```

```

#include <boost/tokenizer.hpp>
#include <boost/lexical_cast.hpp>
#include <boost/date_time/gregorian/parsers.hpp>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/InventoryKey.hpp>
#include <stdair/bom/FlightDateKey.hpp>
#include <stdair/bom/SegmentDateKey.hpp>
#include <stdair/bom/ParsedKey.hpp>
#include <stdair/service/Logger.hpp>

```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

Functions

- const boost::char_separator< char > `stdair::TokeniserDashSeparator` ("-")
- const boost::char_separator< char > `stdair::TokeniserTimeSeparator` (":")

35.362 ParsedKey.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost
00008 #include <boost/tokenizer.hpp>
00009 #include <boost/lexical_cast.hpp>
00010 #include <boost/date_time/gregorian/parsers.hpp>
00011 // StdAir
00012 #include <stdair/stdair_exceptions.hpp>
00013 #include <stdair/basic/BasConst_Inventory.hpp>
00014 #include <stdair/basic/BasConst_BomDisplay.hpp>
00015 #include <stdair/bom/InventoryKey.hpp>
00016 #include <stdair/bom/FlightDateKey.hpp>
00017 #include <stdair/bom/SegmentDateKey.hpp>
00018 #include <stdair/bom/ParsedKey.hpp>
00019 #include <stdair/service/Logger.hpp>
00020
00021 namespace stdair {
00022
00023 // /////////////////////////////////// Tokenising support ///////////////////////////////////

```

```

00027 typedef boost::tokenizer<boost::char_separator<char> > Tokeniser_T;
00028
00032 const boost::char_separator<char> TokeniserDashSeparator ("-");
00033
00037 const boost::char_separator<char> TokeniserTimeSeparator (":");
00038
00039 // //////////////////////////////////////
00040 ParsedKey::ParsedKey() : _fullKey (""), _airlineCode (""), _flightNumber (""),
00041                        _departureDate (""), _boardingPoint (""),
00042                        _offPoint (""), _boardingTime ("") {
00043 }
00044
00045 // //////////////////////////////////////
00046 ParsedKey::~ParsedKey() {
00047 }
00048
00049 // //////////////////////////////////////
00050 InventoryKey ParsedKey::getInventoryKey() const {
00051     if (_airlineCode.size() < 2 || _airlineCode.size() > 3) {
00052         STDAIR_LOG_ERROR ("No airline code can be found in '" << _fullKey << "'");
00053         STDAIR_LOG_DEBUG ("Parsed key: " << toString());
00054         throw KeyNotFoundException ("No airline code can be found in '"
00055                                     + _fullKey + "'");
00056     }
00057     return _airlineCode;
00058 }
00059
00060 // //////////////////////////////////////
00061 FlightDateKey ParsedKey::getFlightDateKey() const {
00062     // Check whether the departure date has been parsed correctly.
00063     Tokeniser_T lDateTokens (_departureDate, TokeniserDashSeparator);
00064
00065     if (lDateTokens.begin() == lDateTokens.end()) {
00066         STDAIR_LOG_ERROR ("No date can be found in '" << _fullKey << "'");
00067         STDAIR_LOG_DEBUG ("Parsed key: " << toString());
00068         throw KeyNotFoundException ("No date can be found in '" + _fullKey + "'");
00069     }
00070
00071     const FlightNumber_T lFlightNumber =
00072         boost::lexical_cast<FlightNumber_T> (_flightNumber);
00073
00074     const Date_T lDepartureDate =
00075         boost::gregorian::from_simple_string (_departureDate);
00076
00077     const FlightDateKey oFlightDateKey (lFlightNumber, lDepartureDate);
00078
00079     return oFlightDateKey;
00080 }
00081
00082 // //////////////////////////////////////
00083 SegmentDateKey ParsedKey::getSegmentKey() const {
00084     if (_boardingPoint.size() != 3 || _offPoint.size() != 3) {
00085         STDAIR_LOG_ERROR ("No airport code can be found in '" << _fullKey << "'");
00086         STDAIR_LOG_DEBUG ("Parsed key: " << toString());
00087         throw KeyNotFoundException ("No airport code can be found in '"
00088                                     + _fullKey + "'");
00089     }
00090
00091     const SegmentDateKey oSegmentDateKey (_boardingPoint, _offPoint);
00092
00093     return oSegmentDateKey;
00094 }

```

```

00095
00096 // //////////////////////////////////////
00097 const Duration_T ParsedKey::getBoardingTime() const {
00098     // Check whether the boarding time has been parsed correctly.
00099     Tokeniser_T lTimeTokens (_boardingTime, TokeniserTimeSeparator);
00100
00101     if (lTimeTokens.begin() == lTimeTokens.end()) {
00102         STDAIR_LOG_ERROR ("No boarding time can be found in '" << _fullKey << "'");
00103
00104         STDAIR_LOG_DEBUG ("Parsed key: " << toString());
00105         throw KeyNotFoundException ("No boarding time can be found in '"
00106                                     + _fullKey + "'");
00107     }
00108
00109     const Duration_T oBoardingTime (boost::posix_time::
00110                                     duration_from_string (_boardingTime));
00111
00112     return oBoardingTime;
00113 }
00114 // //////////////////////////////////////
00115 void ParsedKey::toStream (std::ostream& ioOut) const {
00116     ioOut << "ParsedKey: " << toString();
00117 }
00118
00119 // //////////////////////////////////////
00120 void ParsedKey::fromStream (std::istream& ioIn) {
00121 }
00122
00123 // //////////////////////////////////////
00124 const std::string ParsedKey::toString() const {
00125     std::ostringstream oStr;
00126
00127     oStr << _airlineCode
00128         << DEFAULT_KEY_FLD_DELIMITER << " "
00129         << _flightNumber
00130         << DEFAULT_KEY_SUB_FLD_DELIMITER << " "
00131         << _departureDate
00132         << DEFAULT_KEY_FLD_DELIMITER << " "
00133         << _boardingPoint
00134         << DEFAULT_KEY_SUB_FLD_DELIMITER << " "
00135         << _offPoint
00136         << DEFAULT_KEY_FLD_DELIMITER << " "
00137         << _boardingTime;
00138
00139     return oStr.str();
00140 }
00141
00142 }

```

35.363 stdair/bom/ParsedKey.hpp File Reference

```

#include <iosfwd>

#include <string>

#include <stdair/stdair_date_time_types.hpp>

#include <stdair/bom/KeyAbstract.hpp>

```

Classes

- struct [stdair::ParsedKey](#)

Namespaces

- namespace [stdair](#)

Handle on the StdAir library context.

35.364 ParsedKey.hpp

```

00001 #ifndef __STDAIR_BOM_PARSEDKEY_HPP
00002 #define __STDAIR_BOM_PARSEDKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_date_time_types.hpp>
00012 #include <stdair/bom/KeyAbstract.hpp>
00013
00014 namespace stdair {
00015     struct InventoryKey;
00016     struct FlightDateKey;
00017     struct SegmentDateKey;
00018
00021     struct ParsedKey : public KeyAbstract{
00022
00023         // /////////// Getter ///////////
00025         InventoryKey getInventoryKey () const;
00026
00028         FlightDateKey getFlightDateKey () const;
00029
00031         SegmentDateKey getSegmentKey () const;
00032
00034         const Duration_T getBoardingTime () const;
00035
00036     public:
00037         // /////////// Display support methods ///////////
00043         void toStream (std::ostream& ioOut) const;
00044
00050         void fromStream (std::istream& ioIn);
00051
00061         const std::string toString() const;
00062
00063     public:
00064         // /////////// Constructor and destructor. ///////////
00065         // Default constructor
00066         ParsedKey ();
00067         // Default destructor
00068         ~ParsedKey ();
00069
00070     public:
00071         // /////////// Attributes ///////////
00072         std::string _fullKey;

```

```

00073     std::string _airlineCode;
00074     std::string _flightNumber;
00075     std::string _departureDate;
00076     std::string _boardingPoint;
00077     std::string _offPoint;
00078     std::string _boardingTime;
00079 };
00080
00081 }
00082 #endif // __STDAIR_BOM_PARSEDKEY_HPP

```

35.365 stdair/bom/PeriodStruct.cpp File Reference

```

#include <sstream>
#include <cassert>
#include <stdair/basic/BasConst_Period_BOM.hpp>
#include <stdair/bom/PeriodStruct.hpp>

```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.366 PeriodStruct.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <sstream>
00006 #include <cassert>
00007 // StdAir
00008 #include <stdair/basic/BasConst_Period_BOM.hpp>
00009 #include <stdair/bom/PeriodStruct.hpp>
00010
00011 namespace stdair {
00012
00013 // //////////////////////////////////////
00014 PeriodStruct::PeriodStruct ()
00015 : _dateRange (BOOST_DEFAULT_DATE_PERIOD), _dow () {
00016 }
00017
00018 // //////////////////////////////////////
00019 PeriodStruct::PeriodStruct (const DatePeriod_T& iDateRange,
00020                             const DoWStruct& iDoW)
00021 : _dateRange (iDateRange), _dow (iDoW) {
00022 }
00023
00024 // //////////////////////////////////////
00025 PeriodStruct::PeriodStruct (const PeriodStruct& iPeriodStruct)
00026 : _dateRange (iPeriodStruct._dateRange), _dow (iPeriodStruct._dow) {
00027 }
00028 }

```

```

00029
00030 ///////////////////////////////////////////////////////////////////
00031 const std::string PeriodStruct::describeShort() const {
00032     std::ostringstream ostr;
00033     ostr << _dateRange << ", " << _dow.describeShort ();
00034     return ostr.str();
00035 }
00036
00037 ///////////////////////////////////////////////////////////////////
00038 const std::string PeriodStruct::describe() const {
00039     std::ostringstream ostr;
00040     ostr << _dateRange << ", " << _dow.describe ();
00041     return ostr.str();
00042 }
00043
00044 ///////////////////////////////////////////////////////////////////
00045 PeriodStruct PeriodStruct::
00046 addDateOffset (const DateOffset_T& iDateOffset) const {
00047     // Create a new date range by shifting the date range of this object with
00048     // iDateOffset.
00049     DatePeriod_T lNewDateRange = getDateRange();
00050     lNewDateRange.shift (iDateOffset);
00051
00052     // Create a new DoWStruct by shifting the DoWStruct of this object with
00053     // iDateOffset.
00054     const long lNbOfDaysOffset = iDateOffset.days();
00055     const DoWStruct& lDoW = getDoW();
00056     const DoWStruct lNewDoW = lDoW.shift (lNbOfDaysOffset);
00057
00058     return PeriodStruct (lNewDateRange, lNewDoW);
00059 }
00060
00061 ///////////////////////////////////////////////////////////////////
00062 PeriodStruct PeriodStruct::
00063 intersection (const PeriodStruct& iPeriodStruct) const {
00064     const DatePeriod_T lNewDateRange =
00065         _dateRange.intersection (iPeriodStruct._dateRange);
00066     const DoWStruct lNewDoW = _dow.intersection (iPeriodStruct._dow);
00067
00068     return PeriodStruct (lNewDateRange, lNewDoW);
00069 }
00070
00071 ///////////////////////////////////////////////////////////////////
00072 const bool PeriodStruct::isValid () const {
00073     if (_dateRange.is_null() == false && _dow.isValid()) {
00074         return true;
00075     }
00076     return false;
00077 }
00078
00079 }

```

35.367 stdair/bom/PeriodStruct.hpp File Reference

```

#include <string>

#include <vector>

#include <stdair/stdair_types.hpp>

#include <stdair/basic/StructAbstract.hpp>

```

```
#include <stdair/bom/DoWStruct.hpp>
```

Classes

- struct `stdair::PeriodStruct`

Namespaces

- namespace `stdair`

Handle on the StdAir library context.

35.368 PeriodStruct.hpp

```
00001 #ifndef __STDAIR_BOM_PERIODSTRUCT_HPP
00002 #define __STDAIR_BOM_PERIODSTRUCT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010 // StdAir
00011 #include <stdair/stdair_types.hpp>
00012 #include <stdair/basic/StructAbstract.hpp>
00013 #include <stdair/bom/DoWStruct.hpp>
00014
00015 namespace stdair {
00016
00017     struct PeriodStruct : public StructAbstract {
00018     public:
00019         // //////////// Getters ////////////
00020         const DatePeriod_T& getDateRange () const {
00021             return _dateRange;
00022         }
00023         const DoWStruct& getDoW () const {
00024             return _dow;
00025         }
00026     public:
00027         // //////////// Setters ////////////
00028         void setDateRange (const DatePeriod_T& iDateRange) {
00029             _dateRange = iDateRange;
00030         }
00031         void setDoW (const DoWStruct& iDoW) { _dow = iDoW; }
00032     public:
00033         const std::string describe() const;
00034
00035         const std::string describeShort() const;
00036     public:
00037         // //////////// Business Methods ////////////
00038         PeriodStruct addDateOffset (const DateOffset_T&) const;
00039
00040         PeriodStruct intersection (const PeriodStruct&) const;
```

```

00053
00055     const bool isValid () const;
00056
00057 public:
00059     PeriodStruct (const DatePeriod_T&, const DoWStruct&);
00061     PeriodStruct ();
00062     PeriodStruct (const PeriodStruct&);
00064     ~PeriodStruct () { }
00065
00066 private:
00067     // Attributes
00068     DatePeriod_T _dateRange;
00069     DoWStruct _dow;
00070 };
00071
00072 }
00073 #endif // __STDAIR_BOM_PERIODSTRUCT_HPP

```

35.369 stdair/bom/PosChannel.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_Request.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/PosChannel.hpp>

```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

35.370 PosChannel.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_Request.hpp>
00009 #include <stdair/service/Logger.hpp>
00010 #include <stdair/bom/PosChannel.hpp>
00011
00012 namespace stdair {
00013
00014 // //////////////////////////////////////
00015 PosChannel::PosChannel()
00016     : _key (DEFAULT_POS,
00017           DEFAULT_CHANNEL),
00018     _parent (NULL) {
00019     // That constructor is used by the serialisation process

```

```

00020     }
00021
00022     // //////////////////////////////////////
00023     PosChannel::PosChannel (const PosChannel& iPosChannel)
00024         : _key (iPosChannel.getKey()), _parent (NULL) {
00025         assert (false);
00026     }
00027
00028     // //////////////////////////////////////
00029     PosChannel::PosChannel (const Key_T& iKey)
00030         : _key (iKey), _parent (NULL) {
00031     }
00032
00033     // //////////////////////////////////////
00034     PosChannel::~PosChannel () {
00035     }
00036
00037     // //////////////////////////////////////
00038     std::string PosChannel::toString() const {
00039         std::ostringstream oStr;
00040         oStr << describeKey();
00041         return oStr.str();
00042     }
00043 }

```

35.371 stdair/bom/PosChannel.hpp File Reference

```

#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/PosChannelKey.hpp>
#include <stdair/bom/PosChannelTypes.hpp>

```

Classes

- class [stdair::PosChannel](#)
Class representing the actual attributes for a fare point of sale.

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.372 PosChannel.hpp

```

00001 #ifndef __STDAIR_BOM_POSCHANNEL_HPP
00002 #define __STDAIR_BOM_POSCHANNEL_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STDAIR
00008 #include <stdair/bom/BomAbstract.hpp>
00009 #include <stdair/bom/PosChannelKey.hpp>

```

```
00010 #include <stdair/bom/PosChannelTypes.hpp>
00011
00012 // Forward declaration
00013 namespace stdair {
00014
00015     class PosChannel : public BomAbstract {
00020         template <typename BOM> friend class FacBom;
00021         friend class FacBomManager;
00022
00023     public:
00024         // ////////// Type definitions
00028         typedef PosChannelKey Key_T;
00029
00030     public:
00031         // ////////// Display support methods //////////
00037         void toStream (std::ostream& ioOut) const {
00038             ioOut << toString();
00039         }
00040
00046         void fromStream (std::istream& ioIn) {
00047         }
00048
00052         std::string toString() const;
00053
00057         const std::string describeKey() const {
00058             return _key.toString();
00059         }
00060
00061     public:
00062         // ////////// Getters //////////
00066         const Key_T& getKey() const {
00067             return _key;
00068         }
00069
00073         BomAbstract* const getParent() const {
00074             return _parent;
00075         }
00076
00080         const stdair::HolderMap_T& getHolderMap() const {
00081             return _holderMap;
00082         }
00083
00087         const CityCode_T& getPos() const {
00088             return _key.getPos();
00089         }
00090
00094         const ChannelLabel_T& getChannel() const {
00095             return _key.getChannel();
00096         }
00097
00098     protected:
00099         // ////////// Constructors and destructors //////////
00103         PosChannel (const Key_T&);
00104
00108         virtual ~PosChannel();
00109
00110     private:
00114         PosChannel ();
00115
00119         PosChannel (const PosChannel&);
00120
00121     protected:
```

```

00122      // //////////// Attributes ////////////
00126      Key_T _key;
00127
00131      BomAbstract* _parent;
00132
00136      HolderMap_T _holderMap;
00137
00138  };
00139
00140 }
00141 #endif // __STDAIR_BOM_POSCHANNEL_HPP
00142

```

35.373 stdair/bom/PosChannelKey.cpp File Reference

```

#include <ostream>
#include <sstream>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/basic/BasConst_Request.hpp>
#include <stdair/bom/PosChannelKey.hpp>

```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.374 PosChannelKey.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <ostream>
00006 #include <sstream>
00007 // STDAIR
00008 #include <stdair/basic/BasConst_BomDisplay.hpp>
00009 #include <stdair/basic/BasConst_Request.hpp>
00010 #include <stdair/bom/PosChannelKey.hpp>
00011
00012 namespace stdair {
00013
00014      // //////////////////////////////////////
00015      PosChannelKey::PosChannelKey()
00016          : _pos (DEFAULT_POS),
00017            _channel (DEFAULT_CHANNEL) {
00018          assert (false);
00019      }
00020
00021      // //////////////////////////////////////
00022      PosChannelKey::PosChannelKey (const CityCode_T& iPos,
00023                                   const ChannelLabel_T& iChannel)
00024          : _pos (iPos), _channel(iChannel) {

```

```

00025     }
00026
00027     // //////////////////////////////////////
00028     PosChannelKey::PosChannelKey (const PosChannelKey& iKey)
00029         : _pos (iKey._pos), _channel (iKey._channel) {
00030     }
00031
00032     // //////////////////////////////////////
00033     PosChannelKey::~PosChannelKey () {
00034     }
00035
00036     // //////////////////////////////////////
00037     void PosChannelKey::toStream (std::ostream& ioOut) const {
00038         ioOut << "PosChannelKey: " << toString() << std::endl;
00039     }
00040
00041     // //////////////////////////////////////
00042     void PosChannelKey::fromStream (std::istream& ioIn) {
00043     }
00044
00045     // //////////////////////////////////////
00046     const std::string PosChannelKey::toString() const {
00047         std::ostringstream oStr;
00048         oStr << _pos << DEFAULT_KEY_SUB_FLD_DELIMITER
00049             << " " << _channel;
00050         return oStr.str();
00051     }
00052
00053 }

```

35.375 stdair/bom/PosChannelKey.hpp File Reference

```

#include <stdair/bom/KeyAbstract.hpp>
#include <stdair/stdair_types.hpp>

```

Classes

- struct [stdair::PosChannelKey](#)
Key of point of sale and channel.

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.376 PosChannelKey.hpp

```

00001 #ifndef __STDAIR_BOM_POSCHANNELKEY_HPP
00002 #define __STDAIR_BOM_POSCHANNELKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section

```

```

00006 // //////////////////////////////////////
00007 // SIMFQT
00008 #include <stdair/bom/KeyAbstract.hpp>
00009 #include <stdair/stdair_types.hpp>
00010
00011 namespace stdair {
00015     struct PosChannelKey : public KeyAbstract {
00016     public:
00017         // ////////////////////////////////// Construction //////////////////////////////////
00018         PosChannelKey (const stdair::CityCode_T&, const stdair::ChannelLabel_T&);
00026         PosChannelKey (const PosChannelKey&);
00030         ~PosChannelKey ();
00031     private:
00035         PosChannelKey ();
00036
00037     public:
00038         // ////////////////////////////////// Getters //////////////////////////////////
00039
00043         const stdair::CityCode_T& getPos() const {
00044             return _pos;
00045         }
00046
00050         const stdair::ChannelLabel_T& getChannel() const {
00051             return _channel;
00052         }
00053
00054     public:
00055         // ////////////////////////////////// Display support methods //////////////////////////////////
00060         void toStream (std::ostream& ioOut) const;
00061
00066         void fromStream (std::istream& ioIn);
00067
00072         const std::string toString() const;
00073
00074     private:
00075         // ////////////////////////////////// Attributes //////////////////////////////////
00079         CityCode_T _pos;
00080
00085         ChannelLabel_T _channel;
00086
00087     };
00088
00089 }
00090 #endif // __STDAIR_BOM_POSCHANNELKEY_HPP

```

35.377 stdair/bom/PosChannelTypes.hpp File Reference

```

#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>

```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

Typedefs

- typedef std::list< PosChannel * > [stdair::PosChannelList_T](#)
- typedef std::map< const MapKey_T, PosChannel * > [stdair::PosChannelMap_T](#)
- typedef std::pair< MapKey_T, PosChannel * > [stdair::PosChannelWithKey_T](#)
- typedef std::list< PosChannelWithKey_T > [stdair::PosChannelDetailedList_T](#)

35.378 PosChannelTypes.hpp

```

00001 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00002 #ifndef __STDAIR_BOM_POSCHANNELTYPES_HPP
00003 #define __STDAIR_BOM_POSCHANNELTYPES_HPP
00004
00005 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00006 // Import section
00007 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // STDAIR
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016     // Forward declarations.
00017     class PosChannel;
00018
00020     typedef std::list<PosChannel*> PosChannelList_T;
00021
00023     typedef std::map<const MapKey_T, PosChannel*> PosChannelMap_T;
00024
00026     typedef std::pair<MapKey_T, PosChannel*> PosChannelWithKey_T;
00027     typedef std::list<PosChannelWithKey_T> PosChannelDetailedList_T;
00028 }
00029 #endif // __STDAIR_BOM_POSCHANNELTYPES_HPP

```

35.379 stdair/bom/RMEventStruct.cpp File Reference

```

#include <cassert>

#include <sstream>

#include <stdair/bom/RMEventStruct.hpp>

```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.380 RMEventStruct.cpp

```

00001 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00002 // Import section

```

```

00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/bom/RMEventStruct.hpp>
00009
00010 namespace stdair {
00011
00012 // //////////////////////////////////////
00013 RMEventStruct::RMEventStruct() {
00014     assert (false);
00015 }
00016
00017 // //////////////////////////////////////
00018 RMEventStruct::
00019 RMEventStruct (const RMEventStruct& iRMEvent)
00020 : _airlineCode (iRMEvent._airlineCode),
00021   _flightDateDescription (iRMEvent._flightDateDescription),
00022   _snapshotTime (iRMEvent._snapshotTime) {
00023 }
00024
00025 // //////////////////////////////////////
00026 RMEventStruct::
00027 RMEventStruct (const AirlineCode_T& iAirlineCode,
00028               const KeyDescription_T& iFlightDateDescription,
00029               const DateTime_T& iRMEventTime)
00030 : _airlineCode (iAirlineCode),
00031   _flightDateDescription (iFlightDateDescription),
00032   _snapshotTime (iRMEventTime) {
00033 }
00034
00035 // //////////////////////////////////////
00036 RMEventStruct::~RMEventStruct() {
00037 }
00038
00039 // //////////////////////////////////////
00040 void RMEventStruct::toStream (std::ostream& ioOut) const {
00041     ioOut << describe();
00042 }
00043
00044 // //////////////////////////////////////
00045 void RMEventStruct::fromStream (std::istream& ioIn) {
00046 }
00047
00048 // //////////////////////////////////////
00049 const std::string RMEventStruct::describe() const {
00050     std::ostringstream oStr;
00051     oStr << _airlineCode << ", " << _flightDateDescription << ", "
00052         << _snapshotTime;
00053     return oStr.str();
00054 }
00055
00056 }

```

35.381 stdair/bom/RMEventStruct.hpp File Reference

```

#include <iosfwd>

#include <string>

```

```
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/RMEventTypes.hpp>
```

Classes

- struct [stdair::RMEventStruct](#)

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.382 RMEventStruct.hpp

```
00001 #ifndef __STDAIR_BOM_RMEVENTSTRUCT_HPP
00002 #define __STDAIR_BOM_RMEVENTSTRUCT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/stdair_demand_types.hpp>
00013 #include <stdair/basic/StructAbstract.hpp>
00014 #include <stdair/bom/RMEventTypes.hpp>
00015
00016 namespace stdair {
00017
00019     struct RMEventStruct : public StructAbstract {
00020     public:
00021         // ////////////////////////////////////// Getters //////////////////////////////////////
00023         const AirlineCode_T& getAirlineCode() const {
00024             return _airlineCode;
00025         }
00026
00028         const KeyDescription_T& getFlightDateDescription() const {
00029             return _flightDateDescription;
00030         }
00031
00033         const DateTime_T& getRMEventTime() const {
00034             return _snapshotTime;
00035         }
00036
00037         // ////////////////////////////////////// Display support method //////////////////////////////////////
00040         void toStream (std::ostream& ioOut) const;
00041
00044         void fromStream (std::istream& ioIn);
00045
00047         const std::string describe() const;
```

```

00048
00049
00050 // ////////////////////////////////// Constructors and Destructors //////////////////////////////////
00051 public:
00052     RMEventStruct (const AirlineCode_T&, const KeyDescription_T&,
00053                   const DateTime_T&);
00054
00055     RMEventStruct (const RMEventStruct&);
00056
00057     RMEventStruct ();
00058
00059 public:
00060     ~RMEventStruct ();
00061
00062 private:
00063 // ////////////////////////////////// Attributes //////////////////////////////////
00064     const AirlineCode_T _airlineCode;
00065
00066     const KeyDescription_T _flightDateDescription;
00067
00068     const DateTime_T _snapshotTime;
00069 };
00070
00071 }
00072 #endif // __STDAIR_BOM_RMEVENTSTRUCT_HPP

```

35.383 stdair/bom/RMEventTypes.hpp File Reference

```

#include <list>
#include <boost/shared_ptr.hpp>

```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

Typedefs

- typedef boost::shared_ptr< RMEventStruct > [stdair::RMEventPtr_T](#)
- typedef std::list< RMEventStruct > [stdair::RMEventList_T](#)

35.384 RMEventTypes.hpp

```

00001 // //////////////////////////////////
00002 #ifndef __STDAIR_BOM_RMEVENTTYPES_HPP
00003 #define __STDAIR_BOM_RMEVENTTYPES_HPP
00004
00005 // //////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////
00008 // STL
00009 #include <list>

```

```

00010 // Boost
00011 #include <boost/shared_ptr.hpp>
00012
00013 namespace stdair {
00014
00015     // Forward declarations
00016     struct RMEventStruct;
00017
00018     // ////////////////// Type definitions //////////////////
00020     typedef boost::shared_ptr<RMEventStruct> RMEventPtr_T;
00021
00023     typedef std::list<RMEventStruct> RMEventList_T;
00024
00025 }
00026 #endif // __STDAIR_BOM_RMEVENTTYPES_HPP
00027

```

35.385 stdair/bom/SegmentCabin.cpp File Reference

```

#include <cassert>
#include <sstream>

#include <stdair/basic/BasConst_BookingClass.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/SegmentCabin.hpp>

```

Namespaces

- namespace `stdair`
Handle on the *StdAir* library context.

35.386 SegmentCabin.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_BookingClass.hpp>
00009 #include <stdair/basic/BasConst_Inventory.hpp>
00010 #include <stdair/basic/BasConst_BomDisplay.hpp>
00011 #include <stdair/bom/BomManager.hpp>
00012 #include <stdair/bom/SegmentDate.hpp>
00013 #include <stdair/bom/SegmentCabin.hpp>
00014
00015 namespace stdair {

```

```

00016
00017 // //////////////////////////////////////
00018 SegmentCabin::SegmentCabin() : _key (DEFAULT_CABIN_CODE), _parent (NULL) {
00019     assert (false);
00020 }
00021
00022 // //////////////////////////////////////
00023 SegmentCabin::SegmentCabin (const SegmentCabin&)
00024 : _key (DEFAULT_CABIN_CODE), _parent (NULL) {
00025     assert (false);
00026 }
00027
00028 // //////////////////////////////////////
00029 SegmentCabin::SegmentCabin (const Key_T& iKey)
00030 : _key (iKey), _parent (NULL),
00031   _capacity (DEFAULT_CABIN_CAPACITY),
00032   _blockSpace (DEFAULT_BLOCK_SPACE),
00033   _bookingCounter (DEFAULT_CLASS_NB_OF_BOOKINGS),
00034   _committedSpace (DEFAULT_COMMITTED_SPACE),
00035   _availabilityPool (DEFAULT_AVAILABILITY),
00036   _bidPriceVector (DEFAULT_BID_PRICE_VECTOR),
00037   _currentBidPrice (DEFAULT_BID_PRICE),
00038   _fareFamilyActivation (false) {
00039 }
00040
00041 // //////////////////////////////////////
00042 SegmentCabin::~SegmentCabin() {
00043 }
00044
00045 // //////////////////////////////////////
00046 const MapKey_T SegmentCabin::getFullerKey() const {
00047     const SegmentDate& lSegmentDate = BomManager::getParent<SegmentDate>(*this);
00048
00049     const MapKey_T oFullKey =
00050         lSegmentDate.describeKey() + DEFAULT_KEY_FLD_DELIMITER + getCabinCode();
00051     return oFullKey;
00052 }
00053
00054 // //////////////////////////////////////
00055 std::string SegmentCabin::toString() const {
00056     std::ostringstream oStr;
00057     oStr << describeKey();
00058     return oStr.str();
00059 }
00060
00061 // //////////////////////////////////////
00062 void SegmentCabin::
00063 updateFromReservation (const NbOfBookings_T& iNbOfBookings) {
00064     _committedSpace += iNbOfBookings;
00065 }
00066
00067 }
00068

```

35.387 stdair/bom/SegmentCabin.hpp File Reference

```
#include <iosfwd>
```

```
#include <string>
```

```
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/SegmentCabinKey.hpp>
#include <stdair/bom/SegmentCabinTypes.hpp>
```

Classes

- class `stdair::SegmentCabin`
Class representing the actual attributes for an airline segment-cabin.

Namespaces

- namespace `boost`
Forward declarations.
- namespace `boost::serialization`
- namespace `stdair`
Handle on the StdAir library context.

35.388 SegmentCabin.hpp

```
00001 #ifndef __STDAIR_BOM_SEGMENTCABIN_HPP
00002 #define __STDAIR_BOM_SEGMENTCABIN_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/bom/BomAbstract.hpp>
00013 #include <stdair/bom/SegmentCabinKey.hpp>
00014 #include <stdair/bom/SegmentCabinTypes.hpp>
00015
00016 namespace boost {
00017     namespace serialization {
00018         class access;
00019     }
00020 }
00021
00022 namespace stdair {
00023     // Forward declarations
00024     class GuillotineBlock;
00025
00026     class SegmentCabin : public BomAbstract {
00027     template <typename BOM> friend class FacBom;
00028     friend class FacBomManager;
00029     friend class boost::serialization::access;
00030
00031     public:
00032         // ////////////////////////////////// Type definitions //////////////////////////////////
00033 }
```

```
00041     typedef SegmentCabinKey Key_T;
00042
00043
00044 public:
00045     // //////////// Getters ////////////
00049     const Key_T& getKey() const {
00050         return _key;
00051     }
00052
00056     BomAbstract* const getParent() const {
00057         return _parent;
00058     }
00059
00063     const HolderMap_T& getHolderMap() const {
00064         return _holderMap;
00065     }
00066
00070     const CabinCode_T& getCabinCode() const {
00071         return _key.getCabinCode();
00072     }
00073
00082     const MapKey_T getFullerKey() const;
00083
00085     const GuillotineBlock& getGuillotineBlock() const {
00086         assert (_guillotineBlock != NULL);
00087         return *_guillotineBlock;
00088     }
00089
00091     const CabinCapacity_T& getCapacity() const {
00092         return _capacity;
00093     }
00094
00096     const BlockSpace_T& getBlockSpace() const {
00097         return _blockSpace;
00098     }
00099
00101     const BlockSpace_T& getMIN() const {
00102         return _min;
00103     }
00104
00106     const UPR_T& getUPR() const {
00107         return _upr;
00108     }
00109
00111     const NbOfBookings_T& getBookingCounter() const {
00112         return _bookingCounter;
00113     }
00114
00116     const CommittedSpace_T& getCommittedSpace() const {
00117         return _committedSpace;
00118     }
00119
00121     const Availability_T& getAvailabilityPool() const {
00122         return _availabilityPool;
00123     }
00124
00126     const BidPrice_T& getCurrentBidPrice() const {
00127         return _currentBidPrice;
00128     }
00129
00131     const BidPriceVector_T& getBidPriceVector() const {
00132         return _bidPriceVector;
```

```
00133     }
00134
00136     const bool getFareFamilyStatus() const {
00137         return _fareFamilyActivation;
00138     }
00139
00140 public:
00141     // ////////// Setters //////////
00143     void setGuillotineBlock (GuillotineBlock& ioGuillotine) {
00144         _guillotineBlock = &ioGuillotine;
00145     }
00146
00148     void setCapacity (const CabinCapacity_T& iCapacity) {
00149         _capacity = iCapacity;
00150     }
00151
00153     void setBlockSpace (const BlockSpace_T& iBlockSpace) {
00154         _blockSpace = iBlockSpace;
00155     }
00156
00158     void setMIN (const BlockSpace_T& iMIN) {
00159         _min = iMIN;
00160     }
00161
00163     void setUPR (const UPR_T& iUPR) {
00164         _upr = iUPR;
00165     }
00166
00168     void setBookingCounter (const NbOfBookings_T& iBookingCounter) {
00169         _bookingCounter = iBookingCounter;
00170     }
00171
00173     void setCommittedSpace (const CommittedSpace_T& iCommittedSpace) {
00174         _committedSpace = iCommittedSpace;
00175     }
00176
00178     void setAvailabilityPool (const Availability_T& iAvailabilityPool) {
00179         _availabilityPool = iAvailabilityPool;
00180     }
00181
00183     void setBidPriceVector (const BidPriceVector_T& iBPV) {
00184         _bidPriceVector = iBPV;
00185     }
00186
00188     void activateFareFamily () {
00189         _fareFamilyActivation = true;
00190     }
00191
00192 public:
00193     // ////////// Business methods //////////
00195     void updateFromReservation (const NbOfBookings_T&);
00196
00197 public:
00198     // ////////// Display support methods //////////
00199     void toStream (std::ostream& ioOut) const {
00200         ioOut << toString();
00201     }
00202
00208
00214     void fromStream (std::istream& ioIn) {
00215     }
00216
```

```

00220     std::string toString() const;
00221
00225     const std::string describeKey() const {
00226         return _key.toString();
00227     }
00228
00229
00230 public:
00231     // ////////// (Boost) Serialisation support methods //////////
00235     template<class Archive>
00236     void serialize (Archive& ar, const unsigned int iFileVersion);
00237
00238 private:
00246     void serialisationImplementationExport() const;
00247     void serialisationImplementationImport();
00248
00249
00250 protected:
00251     // ////////// Constructors and destructors //////////
00255     SegmentCabin (const Key_T&);
00256
00260     virtual ~SegmentCabin();
00261
00262 private:
00266     SegmentCabin();
00267
00271     SegmentCabin (const SegmentCabin&);
00272
00273
00274 protected:
00275     // ////////// Attributes //////////
00279     Key_T _key;
00280
00284     BomAbstract* _parent;
00285
00289     HolderMap_T _holderMap;
00290
00294     GuillotineBlock* _guillotineBlock;
00295
00297     CabinCapacity_T _capacity;
00298
00300     BlockSpace_T _blockSpace;
00301
00303     BlockSpace_T _min;
00304
00306     UPR_T _upr;
00307
00309     NbOfBookings_T _bookingCounter;
00310
00312     CommittedSpace_T _committedSpace;
00313
00315     Availability_T _availabilityPool;
00316
00318     BidPriceVector_T _bidPriceVector;
00319
00321     BidPrice_T _currentBidPrice;
00322
00324     bool _fareFamilyActivation;
00325 };
00326
00327 }
00328 #endif // __STDAIR_BOM_SEGMENTCABIN_HPP

```

00329

35.389 stdair/bom/SegmentCabinKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/SegmentCabinKey.hpp>
```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

Functions

- template void [stdair::SegmentCabinKey::serialize< ba::text_oarchive >](#) (ba::text_oarchive &, unsigned int)
- template void [stdair::SegmentCabinKey::serialize< ba::text_iarchive >](#) (ba::text_iarchive &, unsigned int)

35.390 SegmentCabinKey.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/access.hpp>
00011 // StdAir
00012 #include <stdair/basic/BasConst_Inventory.hpp>
00013 #include <stdair/bom/SegmentCabinKey.hpp>
00014
00015 namespace stdair {
00016
00017 // //////////////////////////////////////
00018 SegmentCabinKey::SegmentCabinKey() : _cabinCode (DEFAULT_CABIN_CODE) {
00019     assert (false);
00020 }
00021
00022 // //////////////////////////////////////
```

```

00023 SegmentCabinKey::SegmentCabinKey (const CabinCode_T& iCabinCode)
00024     : _cabinCode (iCabinCode) {
00025 }
00026
00027 ///////////////////////////////////////////////////////////////////
00028 SegmentCabinKey::SegmentCabinKey (const SegmentCabinKey& iKey)
00029     : _cabinCode (iKey._cabinCode) {
00030 }
00031
00032 ///////////////////////////////////////////////////////////////////
00033 SegmentCabinKey::~SegmentCabinKey () {
00034 }
00035
00036 ///////////////////////////////////////////////////////////////////
00037 void SegmentCabinKey::toStream (std::ostream& ioOut) const {
00038     ioOut << "SegmentCabinKey: " << toString();
00039 }
00040
00041 ///////////////////////////////////////////////////////////////////
00042 void SegmentCabinKey::fromStream (std::istream& ioIn) {
00043 }
00044
00045 ///////////////////////////////////////////////////////////////////
00046 const std::string SegmentCabinKey::toString() const {
00047     std::ostringstream oStr;
00048     oStr << _cabinCode;
00049     return oStr.str();
00050 }
00051
00052 ///////////////////////////////////////////////////////////////////
00053 void SegmentCabinKey::serialisationImplementationExport() const {
00054     std::ostringstream oStr;
00055     boost::archive::text_oarchive oa (oStr);
00056     oa << *this;
00057 }
00058
00059 ///////////////////////////////////////////////////////////////////
00060 void SegmentCabinKey::serialisationImplementationImport() {
00061     std::istringstream iStr;
00062     boost::archive::text_iarchive ia (iStr);
00063     ia >> *this;
00064 }
00065
00066 ///////////////////////////////////////////////////////////////////
00067 template<class Archive>
00068 void SegmentCabinKey::serialize (Archive& ioArchive,
00069     const unsigned int iFileVersion) {
00074     ioArchive & _cabinCode;
00075 }
00076
00077 ///////////////////////////////////////////////////////////////////
00078 // Explicit template instantiation
00079 namespace ba = boost::archive;
00080 template void SegmentCabinKey::
00081     serialize<ba::text_oarchive> (ba::text_oarchive&, unsigned int);
00082 template void SegmentCabinKey::
00083     serialize<ba::text_iarchive> (ba::text_iarchive&, unsigned int);
00084 ///////////////////////////////////////////////////////////////////
00085
00086 }

```

35.391 stdair/bom/SegmentCabinKey.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

Classes

- struct [stdair::SegmentCabinKey](#)
Key of a given segment-cabin, made of a cabin code (only).

Namespaces

- namespace [boost](#)
Forward declarations.
- namespace [boost::serialization](#)
- namespace [stdair](#)
Handle on the StdAir library context.

35.392 SegmentCabinKey.hpp

```
00001 #ifndef __STDAIR_BOM_SEGMENTCABINKEY_HPP
00002 #define __STDAIR_BOM_SEGMENTCABINKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_basic_types.hpp>
00012 #include <stdair/bom/KeyAbstract.hpp>
00013
00015 namespace boost {
00016     namespace serialization {
00017         class access;
00018     }
00019 }
00020
00021 namespace stdair {
00022
00026     struct SegmentCabinKey : public KeyAbstract {
00027         friend class boost::serialization::access;
00028
00029         // ////////////////////////////////// Constructors and destructors //////////////////////////////////
00030     private:
00034         SegmentCabinKey();
00035
00036     public:
```

```

00040     SegmentCabinKey (const CabinCode_T& iCabinCode);
00041
00045     SegmentCabinKey (const SegmentCabinKey&);
00046
00050     ~SegmentCabinKey();
00051
00052
00053 public:
00054     // //////////// Getters ////////////
00056     const CabinCode_T& getCabinCode() const {
00057         return _cabinCode;
00058     }
00059
00060
00061 public:
00062     // //////////// Display support methods ////////////
00068     void toStream (std::ostream& ioOut) const;
00069
00075     void fromStream (std::istream& ioIn);
00076
00086     const std::string toString() const;
00087
00088
00089 public:
00090     // //////////// (Boost) Serialisation support methods ////////////
00094     template<class Archive>
00095     void serialize (Archive& ar, const unsigned int iFileVersion);
00096
00097 private:
00102     void serialisationImplementationExport() const;
00103     void serialisationImplementationImport();
00104
00105
00106 private:
00107     // //////////// Attributes ////////////
00111     CabinCode_T _cabinCode;
00112 };
00113
00114 }
00115 #endif // __STDAIR_BOM_SEGMENTCABINKEY_HPP

```

35.393 stdair/bom/SegmentCabinTypes.hpp File Reference

```

#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>

```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

Typedefs

- typedef `std::list< SegmentCabin * >` `stdair::SegmentCabinList_T`

- typedef std::map< const MapKey_T, SegmentCabin * > [stdair::SegmentCabinMap_T](#)

35.394 SegmentCabinTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_SEGMENTCABINTYPES_HPP
00003 #define __STDAIR_BOM_SEGMENTCABINTYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // StdAir
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016     // Forward declarations.
00017     class SegmentCabin;
00018
00019     typedef std::list<SegmentCabin*> SegmentCabinList_T;
00020
00021     typedef std::map<const MapKey_T, SegmentCabin*> SegmentCabinMap_T;
00022
00023 }
00024
00025 #endif // __STDAIR_BOM_SEGMENTCABINTYPES_HPP
00026
00027

```

35.395 stdair/bom/SegmentDate.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_BookingClass.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/SegmentCabin.hpp>

```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.396 SegmentDate.cpp

```

00001 // //////////////////////////////////////

```

```

00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_BookingClass.hpp>
00009 #include <stdair/basic/BasConst_Inventory.hpp>
00010 #include <stdair/bom/BomManager.hpp>
00011 #include <stdair/bom/SegmentDate.hpp>
00012 #include <stdair/bom/SegmentCabin.hpp>
00013
00014 namespace stdair {
00015
00016 // //////////////////////////////////////
00017 SegmentDate::SegmentDate()
00018 : _key (DEFAULT_ORIGIN, DEFAULT_DESTINATION), _parent (NULL),
00019   _operatingSegmentDate (NULL) {
00020   assert (false);
00021 }
00022
00023 // //////////////////////////////////////
00024 SegmentDate::SegmentDate (const SegmentDate&)
00025 : _key (DEFAULT_ORIGIN, DEFAULT_DESTINATION), _parent (NULL),
00026   _operatingSegmentDate (NULL) {
00027   assert (false);
00028 }
00029
00030 // //////////////////////////////////////
00031 SegmentDate::SegmentDate (const Key_T& iKey)
00032 : _key (iKey), _parent (NULL) ,
00033   _operatingSegmentDate (NULL) {
00034 }
00035
00036 // //////////////////////////////////////
00037 SegmentDate::~SegmentDate () {
00038 }
00039
00040 // //////////////////////////////////////
00041 std::string SegmentDate::toString() const {
00042   std::ostringstream oStr;
00043   oStr << describeKey();
00044   return oStr.str();
00045 }
00046
00047 // //////////////////////////////////////
00048 const Duration_T SegmentDate::getTimeOffset() const {
00049   // TimeOffset = (OffTime - BoardingTime) + (OffDate - BoardingDate) * 24
00050   //              - ElapsedTime
00051   Duration_T oTimeOffset = (_offTime - _boardingTime);
00052   const DateOffset_T& lDateOffset = getDateOffset();
00053   const Duration_T lDateOffsetInHours (lDateOffset.days() * 24, 0, 0);
00054   oTimeOffset += lDateOffsetInHours - _elapsedTime;
00055   return oTimeOffset;
00056 }
00057 }
00058

```

35.397 stdair/bom/SegmentDate.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/SegmentDateKey.hpp>
#include <stdair/bom/SegmentDateTypes.hpp>
```

Classes

- class [stdair::SegmentDate](#)
Class representing the actual attributes for an airline segment-date.

Namespaces

- namespace [boost](#)
Forward declarations.
- namespace [boost::serialization](#)
- namespace [stdair](#)
Handle on the StdAir library context.

35.398 SegmentDate.hpp

```
00001 #ifndef __STDAIR_BOM_SEGMENTDATE_HPP
00002 #define __STDAIR_BOM_SEGMENTDATE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/bom/BomAbstract.hpp>
00013 #include <stdair/bom/SegmentDateKey.hpp>
00014 #include <stdair/bom/SegmentDateTypes.hpp>
00015
00016 namespace boost {
00017     namespace serialization {
00018         class access;
00019     }
00020 }
00021
00022 namespace stdair {
00023
00024     struct SegmentCabinKey;
00025     class SegmentCabin;
```

```
00028
00033 class SegmentDate : public BomAbstract {
00034     template <typename BOM> friend class FacBom;
00035     friend class FacBomManager;
00036     friend class boost::serialization::access;
00037
00038 public:
00039     // //////////// Type definitions ////////////
00043     typedef SegmentDateKey Key_T;
00044
00045
00046 public:
00047     // //////////// Getters ////////////
00049     const Key_T& getKey() const {
00050         return _key;
00051     }
00052
00054     BomAbstract* const getParent() const {
00055         return _parent;
00056     }
00057
00059     const AirportCode_T& getBoardingPoint() const {
00060         return _key.getBoardingPoint();
00061     }
00062
00064     const AirportCode_T& getOffPoint() const {
00065         return _key.getOffPoint();
00066     }
00067
00069     const HolderMap_T& getHolderMap() const {
00070         return _holderMap;
00071     }
00072
00074     const Date_T& getBoardingDate() const {
00075         return _boardingDate;
00076     }
00077
00079     const Duration_T& getBoardingTime() const {
00080         return _boardingTime;
00081     }
00082
00084     const Date_T& getOffDate() const {
00085         return _offDate;
00086     }
00087
00089     const Duration_T& getOffTime() const {
00090         return _offTime;
00091     }
00092
00094     const Duration_T& getElapsedTime() const {
00095         return _elapsedTime;
00096     }
00097
00099     const Distance_T& getDistance() const {
00100         return _distance;
00101     }
00102
00104     const DateOffset_T getDateOffset() const {
00105         return _offDate - _boardingDate;
00106     }
00107
00116     const Duration_T getTimeOffset() const;
```

```

00117
00121     SegmentDate* getOperatingSegmentDate () const {
00122         return _operatingSegmentDate;
00123     }
00124
00128     const SegmentDateList_T& getMarketingSegmentDateList () const {
00129         return _marketingSegmentDateList;
00130     }
00131
00132 public:
00133     // //////////// Setters ////////////
00135     void setBoardingDate (const Date_T& iBoardingDate) {
00136         _boardingDate = iBoardingDate;
00137     }
00138
00140     void setBoardingTime (const Duration_T& iBoardingTime) {
00141         _boardingTime = iBoardingTime;
00142     }
00143
00145     void setOffDate (const Date_T& iOffDate) {
00146         _offDate = iOffDate;
00147     }
00148
00150     void setOffTime (const Duration_T& iOffTime) {
00151         _offTime = iOffTime;
00152     }
00153
00155     void setElapsedTime (const Duration_T& iElapsedTime) {
00156         _elapsedTime = iElapsedTime;
00157     }
00158
00160     void setDistance (const Distance_T& iDistance) {
00161         _distance = iDistance;
00162     }
00163
00165     void linkWithOperating (SegmentDate& iSegmentDate) {
00166         _operatingSegmentDate = &iSegmentDate;
00167     }
00168
00169 public:
00170     // //////////// Display support methods ////////////
00176     void toStream (std::ostream& ioOut) const {
00177         ioOut << toString();
00178     }
00179
00185     void fromStream (std::istream& ioIn) {
00186     }
00187
00191     std::string toString() const;
00192
00196     const std::string describeKey() const {
00197         return _key.toString();
00198     }
00199
00200
00201 public:
00202     // //////////// (Boost) Serialisation support methods ////////////
00206     template<class Archive>
00207     void serialize (Archive& ar, const unsigned int iFileVersion);
00208
00209 private:
00217     void serialisationImplementationExport() const;

```

```

00218     void serialisationImplementationImport();
00219
00220
00221     protected:
00222         // ////////// Constructors and destructors //////////
00226         SegmentDate (const Key_T&);
00227
00231         virtual ~SegmentDate();
00232
00233     private:
00237         SegmentDate();
00238
00242         SegmentDate (const SegmentDate&);
00243
00244
00245     protected:
00246         // ////////// Attributes //////////
00250         Key_T _key;
00251
00255         BomAbstract* _parent;
00256
00260         HolderMap_T _holderMap;
00261
00268         SegmentDate* _operatingSegmentDate;
00269
00276         SegmentDateList_T _marketingSegmentDateList;
00277
00281         Date_T _boardingDate;
00282
00286         Duration_T _boardingTime;
00287
00291         Date_T _offDate;
00292
00296         Duration_T _offTime;
00297
00301         Duration_T _elapsedTime;
00302
00306         Distance_T _distance;
00307     };
00308
00309 }
00310 #endif // __STDAIR_BOM_SEGMENTDATE_HPP
00311

```

35.399 stdair/bom/SegmentDateKey.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/SegmentDateKey.hpp>

```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

Functions

- template void `stdair::SegmentDateKey::serialize< ba::text_oarchive >` (ba::text_oarchive &, unsigned int)
- template void `stdair::SegmentDateKey::serialize< ba::text_iarchive >` (ba::text_iarchive &, unsigned int)

35.400 SegmentDateKey.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/access.hpp>
00011 // StdAir
00012 #include <stdair/basic/BasConst_Inventory.hpp>
00013 #include <stdair/basic/BasConst_BomDisplay.hpp>
00014 #include <stdair/bom/SegmentDateKey.hpp>
00015
00016 namespace stdair {
00017
00018 // //////////////////////////////////////
00019 SegmentDateKey::SegmentDateKey()
00020 : _boardingPoint (DEFAULT_ORIGIN), _offPoint (DEFAULT_DESTINATION) {
00021     assert (false);
00022 }
00023
00024 // //////////////////////////////////////
00025 SegmentDateKey::SegmentDateKey (const AirportCode_T& iBoardingPoint,
00026                                 const AirportCode_T& iOffPoint)
00027 : _boardingPoint (iBoardingPoint), _offPoint (iOffPoint) {
00028 }
00029
00030 // //////////////////////////////////////
00031 SegmentDateKey::SegmentDateKey (const SegmentDateKey& iKey)
00032 : _boardingPoint (iKey._boardingPoint), _offPoint (iKey._offPoint) {
00033 }
00034
00035 // //////////////////////////////////////
00036 SegmentDateKey::~SegmentDateKey() {
00037 }
00038
00039 // //////////////////////////////////////
00040 void SegmentDateKey::toStream (std::ostream& ioOut) const {
00041     ioOut << "SegmentDateKey: " << toString() << std::endl;
00042 }
00043

```

```

00044 ///////////////////////////////////////////////////////////////////
00045 void SegmentDateKey::fromStream (std::istream& ioIn) {
00046 }
00047
00048 ///////////////////////////////////////////////////////////////////
00049 const std::string SegmentDateKey::toString() const {
00050     std::ostringstream ostr;
00051     ostr << _boardingPoint
00052         << DEFAULT_KEY_SUB_FLD_DELIMITER << " " << _offPoint;
00053     return ostr.str();
00054 }
00055
00056 ///////////////////////////////////////////////////////////////////
00057 void SegmentDateKey::serialisationImplementationExport() const {
00058     std::ostringstream ostr;
00059     boost::archive::text_oarchive oa (ostr);
00060     oa << *this;
00061 }
00062
00063 ///////////////////////////////////////////////////////////////////
00064 void SegmentDateKey::serialisationImplementationImport() {
00065     std::istringstream istr;
00066     boost::archive::text_iarchive ia (istr);
00067     ia >> *this;
00068 }
00069
00070 ///////////////////////////////////////////////////////////////////
00071 template<class Archive>
00072 void SegmentDateKey::serialize (Archive& ioArchive,
00073                                 const unsigned int iFileVersion) {
00074     ioArchive & _boardingPoint & _offPoint;
00075 }
00076
00077 ///////////////////////////////////////////////////////////////////
00078 // Explicit template instantiation
00079 namespace ba = boost::archive;
00080 template void SegmentDateKey::serialize<ba::text_oarchive>(ba::text_oarchive&,
00081                                                            unsigned int);
00082 template void SegmentDateKey::serialize<ba::text_iarchive>(ba::text_iarchive&,
00083                                                            unsigned int);
00084 ///////////////////////////////////////////////////////////////////
00085
00086 }

```

35.401 stdair/bom/SegmentDateKey.hpp File Reference

```
#include <stdair/stdair_basic_types.hpp>
```

```
#include <stdair/bom/KeyAbstract.hpp>
```

Classes

- struct [stdair::SegmentDateKey](#)

Key of a given segment-date, made of an origin and a destination airports.

Namespaces

- namespace `boost`
Forward declarations.
- namespace `boost::serialization`
- namespace `stdair`
Handle on the StdAir library context.

35.402 SegmentDateKey.hpp

```

00001 #ifndef __STDAIR_BOM_SEGMENTDATEKEY_HPP
00002 #define __STDAIR_BOM_SEGMENTDATEKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009 #include <stdair/bom/KeyAbstract.hpp>
00010
00012 namespace boost {
00013     namespace serialization {
00014         class access;
00015     }
00016 }
00017
00018 namespace stdair {
00019
00024     struct SegmentDateKey : public KeyAbstract {
00025         friend class boost::serialization::access;
00026
00027         // ////////////////////////////////// Constructors and destructors //////////////////////////////////
00028     private:
00032         SegmentDateKey();
00033
00034     public:
00038         SegmentDateKey (const AirportCode_T&, const AirportCode_T&);
00042         SegmentDateKey (const SegmentDateKey&);
00046         ~SegmentDateKey();
00047
00048
00049         // ////////////////////////////////// Getters //////////////////////////////////
00051         const AirportCode_T& getBoardingPoint() const {
00052             return _boardingPoint;
00053         }
00054
00056         const AirportCode_T& getOffPoint() const {
00057             return _offPoint;
00058         }
00059
00060
00061         // ////////////////////////////////// Display support methods //////////////////////////////////
00067         void toStream (std::ostream& ioOut) const;
00068
00074         void fromStream (std::istream& ioIn);
00075
00085         const std::string toString() const;
00086

```

```

00087
00088     public:
00089         // //////////// (Boost) Serialisation support methods ////////////
00093         template<class Archive>
00094         void serialize (Archive& ar, const unsigned int iFileVersion);
00095
00096     private:
00101         void serialisationImplementationExport() const;
00102         void serialisationImplementationImport();
00103
00104
00105     private:
00106         // //////////// Attributes ////////////
00110         AirportCode_T _boardingPoint;
00111
00115         AirportCode_T _offPoint;
00116     };
00117
00118 }
00119 #endif // __STDAIR_BOM_SEGMENTDATEKEY_HPP

```

35.403 stdair/bom/SegmentDateTypes.hpp File Reference

```

#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>

```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

Typedefs

- typedef `std::list< SegmentDate * >` `stdair::SegmentDateList_T`
- typedef `std::map< const MapKey_T, SegmentDate * >` `stdair::SegmentDateMap_T`

35.404 SegmentDateTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_SEGMENTDATETYPES_HPP
00003 #define __STDAIR_BOM_SEGMENTDATETYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // StdAir
00012 #include <stdair/bom/key_types.hpp>

```

```

00013
00014 namespace stdair {
00015
00016     // Forward declarations.
00017     class SegmentDate;
00018
00020     typedef std::list<SegmentDate*> SegmentDateList_T;
00021
00023     typedef std::map<const MapKey_T, SegmentDate*> SegmentDateMap_T;
00024 }
00025
00026 #endif // __STDAIR_BOM_SEGMENTDATETYPES_HPP
00027

```

35.405 stdair/bom/SegmentPeriod.cpp File Reference

```

#include <cassert>

#include <stdair/basic/BasConst_BookingClass.hpp>

#include <stdair/bom/SegmentPeriod.hpp>

```

Namespaces

- namespace `stdair`
Handle on the *StdAir* library context.

35.406 SegmentPeriod.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // STDAIR
00007 #include <stdair/basic/BasConst_BookingClass.hpp>
00008 #include <stdair/bom/SegmentPeriod.hpp>
00009
00010 namespace stdair {
00011
00012     // //////////////////////////////////////
00013     SegmentPeriod::SegmentPeriod (const Key_T& iKey)
00014         : _key (iKey), _parent (NULL), _boardingDateOffset (0), _offDateOffset (0) {
00015     }
00016
00017     // //////////////////////////////////////
00018     SegmentPeriod::~SegmentPeriod () {
00019     }
00020
00021     // //////////////////////////////////////
00022     std::string SegmentPeriod::toString() const {
00023         std::ostringstream oStr;
00024         oStr << describeKey();
00025         return oStr.str();
00026     }

```

```

00027
00028 // //////////////////////////////////////
00029 void SegmentPeriod::
00030 addCabinBookingClassList (const CabinCode_T& iCabinCode,
00031                          const ClassList_String_T& iClassCodeList) {
00032     const bool insert = _cabinBookingClassMap.
00033         insert (CabinBookingClassMap_T::value_type (iCabinCode,
00034                                                     iClassCodeList)).second;
00035     assert (insert == true);
00036 }
00037
00038 }

```

35.407 stdair/bom/SegmentPeriod.hpp File Reference

```

#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/SegmentPeriodKey.hpp>
#include <stdair/bom/SegmentPeriodTypes.hpp>

```

Classes

- class [stdair::SegmentPeriod](#)

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.408 SegmentPeriod.hpp

```

00001 #ifndef __STDAIR_BOM_SEGMENTPERIOD_HPP
00002 #define __STDAIR_BOM_SEGMENTPERIOD_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STDAIR
00008 #include <stdair/bom/BomAbstract.hpp>
00009 #include <stdair/bom/SegmentPeriodKey.hpp>
00010 #include <stdair/bom/SegmentPeriodTypes.hpp>
00011
00012 namespace stdair {
00013
00014     class SegmentPeriod : public BomAbstract {
00015     public:
00016         template <typename BOM> friend class FacBom;
00017         friend class FacBomManager;
00018
00019     public:
00020         // Type definitions.
00021         typedef SegmentPeriodKey Key_T;
00022
00023     public:

```

```

00025 // //////////// Getters ////////////
00027 const Key_T& getKey() const { return _key; }
00028
00030 BomAbstract* const getParent() const { return _parent; }
00031
00033 const AirportCode_T& getBoardingPoint () const {
00034     return _key.getBoardingPoint();
00035 }
00036
00038 const AirportCode_T& getOffPoint () const { return _key.getOffPoint(); }
00039
00041 const Duration_T& getBoardingTime () const { return _boardingTime; }
00042
00044 const Duration_T& getOffTime () const { return _offTime; }
00045
00047 const DateOffset_T& getBoardingDateOffset () const {
00048     return _boardingDateOffset;
00049 }
00050
00052 const DateOffset_T& getOffDateOffset () const { return _offDateOffset; }
00053
00055 const Duration_T& getElapsedTime() const { return _elapsedTime; }
00056
00058 const CabinBookingClassMap_T& getCabinBookingClassMap () const {
00059     return _cabinBookingClassMap;
00060 }
00061
00063 const HolderMap_T& getHolderMap() const { return _holderMap; }
00064
00065 public:
00066 // //////////// Setters ////////////
00068 void setBoardingTime (const Duration_T& iBoardingTime) {
00069     _boardingTime = iBoardingTime;
00070 }
00071
00073 void setOffTime (const Duration_T& iOffTime) { _offTime = iOffTime; }
00074
00076 void setBoardingDateOffset (const DateOffset_T& iDateOffset) {
00077     _boardingDateOffset = iDateOffset;
00078 }
00079
00081 void setOffDateOffset (const DateOffset_T& iDateOffset) {
00082     _offDateOffset = iDateOffset;
00083 }
00084
00086 void setElapsedTime (const Duration_T& iElapsedTime) {
00087     _elapsedTime = iElapsedTime;
00088 }
00089
00092 void addCabinBookingClassList (const CabinCode_T&,
00093                               const ClassList_String_T&);
00094
00095 public:
00096 // //////////// Display support methods ////////////
00099 void toStream (std::ostream& ioOut) const { ioOut << toString(); }
00100
00103 void fromStream (std::istream& ioIn) { }
00104
00106 std::string toString() const;
00107
00109 const std::string describeKey() const { return _key.toString(); }
00110
00111

```

```

00111     protected:
00112         SegmentPeriod (const Key_T&);
00113         SegmentPeriod (const SegmentPeriod&);
00114         ~SegmentPeriod();
00115     protected:
00116         // Attributes
00117         Key_T _key;
00118         BomAbstract* _parent;
00119         Duration_T _boardingTime;
00120         Duration_T _offTime;
00121         DateOffset_T _boardingDateOffset;
00122         DateOffset_T _offDateOffset;
00123         Duration_T _elapsedTime;
00124         CabinBookingClassMap_T _cabinBookingClassMap;
00125         HolderMap_T _holderMap;
00126     };
00127 }
00128 #endif // __STDAIR_BOM_SEGMENTPERIOD_HPP
00129

```

35.409 stdair/bom/SegmentPeriodKey.cpp File Reference

```

#include <sstream>

#include <stdair/bom/SegmentPeriodKey.hpp>

```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

35.410 SegmentPeriodKey.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <sstream>
00006 // StdAir
00007 #include <stdair/bom/SegmentPeriodKey.hpp>
00008
00009 namespace stdair {
00010
00011     // //////////////////////////////////////
00012     SegmentPeriodKey::SegmentPeriodKey (const AirportCode_T& iBoardingPoint,
00013                                         const AirportCode_T& iOffPoint)
00014         : _boardingPoint (iBoardingPoint), _offPoint (iOffPoint) {
00015     }
00016
00017     // //////////////////////////////////////
00018     SegmentPeriodKey::SegmentPeriodKey (const SegmentPeriodKey& iKey)
00019         : _boardingPoint (iKey._boardingPoint), _offPoint (iKey._offPoint) {
00020     }
00021

```

```

00022 // //////////////////////////////////////
00023 SegmentPeriodKey::~SegmentPeriodKey () {
00024 }
00025
00026 // //////////////////////////////////////
00027 void SegmentPeriodKey::toStream (std::ostream& ioOut) const {
00028     ioOut << "SegmentPeriodKey: " << toString() << std::endl;
00029 }
00030
00031 // //////////////////////////////////////
00032 void SegmentPeriodKey::fromStream (std::istream& ioIn) {
00033 }
00034
00035 // //////////////////////////////////////
00036 const std::string SegmentPeriodKey::toString() const {
00037     std::ostringstream ostr;
00038     ostr << _boardingPoint << "-" << _offPoint;
00039     return ostr.str();
00040 }
00041
00042 }

```

35.411 stdair/bom/SegmentPeriodKey.hpp File Reference

```

#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>

```

Classes

- struct [stdair::SegmentPeriodKey](#)

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.412 SegmentPeriodKey.hpp

```

00001 #ifndef __STDAIR_BOM_SEGMENTPERIODKEY_HPP
00002 #define __STDAIR_BOM_SEGMENTPERIODKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009 #include <stdair/bom/KeyAbstract.hpp>
00010
00011 namespace stdair {
00012
00014     struct SegmentPeriodKey : public KeyAbstract {
00015
00016     private:

```

```

00017 // //////////// Default constructor ////////////
00018 SegmentPeriodKey () { };
00019 public:
00020 // //////////// Construction ////////////
00022 SegmentPeriodKey (const AirportCode_T&, const AirportCode_T&);
00023 SegmentPeriodKey (const SegmentPeriodKey&);
00025 ~SegmentPeriodKey ();
00026
00027 // //////////// Getters ////////////
00029 const AirportCode_T& getBoardingPoint() const {
00030     return _boardingPoint;
00031 }
00032
00034 const AirportCode_T& getOffPoint() const {
00035     return _offPoint;
00036 }
00037
00038 // //////////// Display support methods ////////////
00041 void toStream (std::ostream& ioOut) const;
00042
00045 void fromStream (std::istream& ioIn);
00046
00052 const std::string toString() const;
00053
00054 private:
00055 // Attributes
00057 AirportCode_T _boardingPoint;
00058
00060 AirportCode_T _offPoint;
00061 };
00062
00063 }
00064 #endif // __STDAIR_BOM_SEGMENTPERIODKEY_HPP

```

35.413 stdair/bom/SegmentPeriodTypes.hpp File Reference

```

#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>

```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

Typedefs

- typedef std::list< SegmentPeriod * > [stdair::SegmentPeriodList_T](#)
- typedef std::map< const MapKey_T, SegmentPeriod * > [stdair::SegmentPeriodMap_T](#)
- typedef std::pair< MapKey_T, SegmentPeriod * > [stdair::SegmentPeriodWithKey_T](#)

- typedef std::list< SegmentPeriodWithKey_T > [stdair::SegmentPeriodDetailedList_T](#)

35.414 SegmentPeriodTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_SEGMENTPERIODTYPES_HPP
00003 #define __STDAIR_BOM_SEGMENTPERIODTYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // StdAir
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016     // Forward declarations.
00017     class SegmentPeriod;
00018
00019     typedef std::list<SegmentPeriod*> SegmentPeriodList_T;
00020
00021     typedef std::map<const MapKey_T, SegmentPeriod*> SegmentPeriodMap_T;
00022
00023     typedef std::pair<MapKey_T, SegmentPeriod*> SegmentPeriodWithKey_T;
00024     typedef std::list<SegmentPeriodWithKey_T> SegmentPeriodDetailedList_T;
00025 }
00026 #endif // __STDAIR_BOM_SEGMENTPERIODTYPES_HPP
00027
00028
00029
00030

```

35.415 stdair/bom/SnapshotStruct.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/bom/SnapshotStruct.hpp>

```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.416 SnapshotStruct.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>

```

```

00007 // StdAir
00008 #include <stdair/bom/SnapshotStruct.hpp>
00009
00010 namespace stdair {
00011
00012     // //////////////////////////////////////
00013     SnapshotStruct::SnapshotStruct() {
00014         assert (false);
00015     }
00016
00017     // //////////////////////////////////////
00018     SnapshotStruct::
00019     SnapshotStruct (const SnapshotStruct& iSnapshot)
00020         : _airlineCode (iSnapshot._airlineCode),
00021           _snapshotTime (iSnapshot._snapshotTime) {
00022     }
00023
00024     // //////////////////////////////////////
00025     SnapshotStruct::
00026     SnapshotStruct (const AirlineCode_T& iAirlineCode,
00027                   const DateTime_T& iSnapshotTime)
00028         : _airlineCode (iAirlineCode), _snapshotTime (iSnapshotTime) {
00029     }
00030
00031     // //////////////////////////////////////
00032     SnapshotStruct::~~SnapshotStruct() {
00033     }
00034
00035     // //////////////////////////////////////
00036     void SnapshotStruct::toStream (std::ostream& ioOut) const {
00037         ioOut << describe();
00038     }
00039
00040     // //////////////////////////////////////
00041     void SnapshotStruct::fromStream (std::istream& ioIn) {
00042     }
00043
00044     // //////////////////////////////////////
00045     const std::string SnapshotStruct::describe() const {
00046         std::ostringstream ostr;
00047         ostr << _airlineCode << ", " << _snapshotTime;
00048         return ostr.str();
00049     }
00050
00051 }

```

35.417 stdair/bom/SnapshotStruct.hpp File Reference

```

#include <iosfwd>

#include <string>

#include <stdair/stdair_inventory_types.hpp>

#include <stdair/stdair_demand_types.hpp>

#include <stdair/basic/StructAbstract.hpp>

#include <stdair/bom/SnapshotTypes.hpp>

```

Classes

- struct `stdair::SnapshotStruct`

Namespaces

- namespace `stdair`

Handle on the StdAir library context.

35.418 SnapshotStruct.hpp

```

00001 #ifndef __STDAIR_BOM_SNAPSHOTSTRUCT_HPP
00002 #define __STDAIR_BOM_SNAPSHOTSTRUCT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/stdair_demand_types.hpp>
00013 #include <stdair/basic/StructAbstract.hpp>
00014 #include <stdair/bom/SnapshotTypes.hpp>
00015
00016 namespace stdair {
00017
00018     struct SnapshotStruct : public StructAbstract {
00019     public:
00020         // ////////////////////////////////////// Getters //////////////////////////////////////
00021         const AirlineCode_T& getAirlineCode() const {
00022             return _airlineCode;
00023         }
00024
00025         const DateTime_T& getSnapshotTime() const {
00026             return _snapshotTime;
00027         }
00028
00029         // ////////////////////////////////////// Display support method //////////////////////////////////////
00030         void toStream (std::ostream& ioOut) const;
00031
00032         void fromStream (std::istream& ioIn);
00033
00034         const std::string describe() const;
00035
00036         // ////////////////////////////////////// Constructors and Destructors //////////////////////////////////////
00037     public:
00038         SnapshotStruct (const AirlineCode_T&, const DateTime_T&);
00039
00040         SnapshotStruct (const SnapshotStruct&);
00041
00042     private:
00043         SnapshotStruct ();
00044
00045     public:
00046         ~SnapshotStruct ();

```

```

00061
00062
00063 private:
00064     // //////////// Attributes ////////////
00066     const AirlineCode_T _airlineCode;
00067
00069     const DateTime_T _snapshotTime;
00070 };
00071
00072 }
00073 #endif // __STDAIR_BOM_SNAPSHOTSTRUCT_HPP

```

35.419 stdair/bom/SnapshotTypes.hpp File Reference

```
#include <boost/shared_ptr.hpp>
```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

Typedefs

- typedef `boost::shared_ptr< SnapshotStruct >` `stdair::SnapshotPtr_T`

35.420 SnapshotTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_SNAPSHOTTYPES_HPP
00003 #define __STDAIR_BOM_SNAPSHOTTYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // Boost
00009 #include <boost/shared_ptr.hpp>
00010
00011 namespace stdair {
00012
00013     // Forward declarations
00014     struct SnapshotStruct;
00015
00016     // //////////// Type definitions ////////////
00018     typedef boost::shared_ptr<SnapshotStruct> SnapshotPtr_T;
00019
00020 }
00021 #endif // __STDAIR_BOM_SNAPSHOTTYPES_HPP
00022

```

35.421 stdair/bom/TimePeriod.cpp File Reference

```
#include <cassert>
```

```
#include <sstream>

#include <stdair/basic/BasConst_General.hpp>

#include <stdair/service/Logger.hpp>

#include <stdair/bom/TimePeriod.hpp>
```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

35.422 TimePeriod.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_General.hpp>
00009 #include <stdair/service/Logger.hpp>
00010 #include <stdair/bom/TimePeriod.hpp>
00011
00012 namespace stdair {
00013
00014 // //////////////////////////////////////
00015 TimePeriod::TimePeriod()
00016 : _key (DEFAULT_EPSILON_DURATION, DEFAULT_EPSILON_DURATION),
00017   _parent (NULL) {
00018     // That constructor is used by the serialisation process
00019 }
00020
00021 // //////////////////////////////////////
00022 TimePeriod::TimePeriod (const TimePeriod& iTimePeriod)
00023 : _key (iTimePeriod.getKey()), _parent (NULL) {
00024     assert (false);
00025 }
00026
00027 // //////////////////////////////////////
00028 TimePeriod::TimePeriod (const Key_T& iKey)
00029 : _key (iKey), _parent (NULL) {
00030 }
00031
00032 // //////////////////////////////////////
00033 TimePeriod::~TimePeriod () {
00034 }
00035
00036 // //////////////////////////////////////
00037 std::string TimePeriod::toString() const {
00038     std::ostringstream oStr;
00039     oStr << describeKey();
00040     return oStr.str();
00041 }
00042
00043 // //////////////////////////////////////
00044 bool TimePeriod::
```

```

00045  isDepartureTimeValid (const Time_T& iFlightTime) const {
00046
00047      const Time_T& lTimeRangeStart = getTimeRangeStart();
00048      const Time_T& lTimeRangeEnd = getTimeRangeEnd();
00049
00050      // Check if the departure time is within the time range.
00051      if (lTimeRangeStart >= iFlightTime) {
00052          // DEBUG
00053          STDAIR_LOG_DEBUG ("Time range begin: " << lTimeRangeStart << ", "
00054                          << "time: " << iFlightTime);
00055          return false;
00056      }
00057      if (lTimeRangeEnd <= iFlightTime) {
00058          // DEBUG
00059          STDAIR_LOG_DEBUG ("Time range end: " << lTimeRangeEnd << ", "
00060                          << "time: " << iFlightTime);
00061          return false;
00062      }
00063
00064      return true;
00065  }
00066
00067 }
00068

```

35.423 stdair/bom/TimePeriod.hpp File Reference

```

#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/TimePeriodKey.hpp>
#include <stdair/bom/TimePeriodTypes.hpp>

```

Classes

- class [stdair::TimePeriod](#)
Class representing the actual attributes for a fare time-period.

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.424 TimePeriod.hpp

```

00001 #ifndef __STDAIR_BOM_FARETIMEPERIOD_HPP
00002 #define __STDAIR_BOM_FARETIMEPERIOD_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STDAIR
00008 #include <stdair/bom/BomAbstract.hpp>
00009 #include <stdair/bom/TimePeriodKey.hpp>

```

```
00010 #include <stdair/bom/TimePeriodTypes.hpp>
00011
00012 // Forward declaration
00013 namespace stdair {
00014
00015     class TimePeriod : public BomAbstract {
00016     public:
00017         template <typename BOM> friend class FacBom;
00018         friend class FacBomManager;
00019
00020     public:
00021         // //////////// Type definitions ////////////
00022         typedef TimePeriodKey Key_T;
00023
00024     public:
00025         // //////////// Display support methods ////////////
00026         // //////////// Display support methods ////////////
00027         void toStream (std::ostream& ioOut) const {
00028             ioOut << toString();
00029         }
00030
00031         void fromStream (std::istream& ioIn) {
00032         }
00033
00034         std::string toString() const;
00035
00036         const std::string describeKey() const {
00037             return _key.toString();
00038         }
00039
00040     public:
00041         // //////////// Getters ////////////
00042         const Key_T& getKey() const {
00043             return _key;
00044         }
00045
00046         BomAbstract* const getParent() const {
00047             return _parent;
00048         }
00049
00050         const HolderMap_T& getHolderMap() const {
00051             return _holderMap;
00052         }
00053
00054         const Time_T& getTimeRangeStart() const {
00055             return _key.getTimeRangeStart();
00056         }
00057
00058         const Time_T& getTimeRangeEnd() const {
00059             return _key.getTimeRangeEnd();
00060         }
00061
00062     public:
00063         // //////////// Business methods ////////////
00064         bool isDepartureTimeValid (const Time_T&) const;
00065
00066     protected:
00067         // //////////// Constructors and destructors ////////////
00068         TimePeriod (const Key_T&);
00069         virtual ~TimePeriod();
00070
00071     private:
00072         TimePeriod();
00073
00074 }
```

```

00125     TimePeriod (const TimePeriod&);
00126
00127 protected:
00128     // //////////// Attributes ////////////
00132     Key_T _key;
00133
00137     BomAbstract* _parent;
00138
00142     HolderMap_T _holderMap;
00143
00144 };
00145
00146 }
00147 #endif // __STDAIR_BOM_FARETIMEPERIOD_HPP
00148

```

35.425 stdair/bom/TimePeriodKey.cpp File Reference

```

#include <ostream>
#include <sstream>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/TimePeriodKey.hpp>

```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

35.426 TimePeriodKey.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <ostream>
00006 #include <sstream>
00007 // STDAIR
00008 #include <stdair/basic/BasConst_General.hpp>
00009 #include <stdair/bom/TimePeriodKey.hpp>
00010
00011 namespace stdair {
00012
00013     // //////////////////////////////////////
00014     TimePeriodKey::TimePeriodKey ()
00015         : _timeRangeStart (DEFAULT_EPSILON_DURATION),
00016           _timeRangeEnd (DEFAULT_EPSILON_DURATION) {
00017         assert (false);
00018     }
00019
00020     // //////////////////////////////////////
00021     TimePeriodKey::TimePeriodKey (const Time_T& iTimeRangeStart,
00022                                   const Time_T& iTimeRangeEnd)
00023         : _timeRangeStart (iTimeRangeStart),

```

```

00024     _timeRangeEnd(iTimeRangeEnd) {
00025     }
00026
00027     // //////////////////////////////////////
00028     TimePeriodKey::TimePeriodKey (const TimePeriodKey& iKey)
00029         : _timeRangeStart(iKey.getTimeRangeStart()),
00030           _timeRangeEnd(iKey.getTimeRangeEnd())    {
00031     }
00032
00033     // //////////////////////////////////////
00034     TimePeriodKey::~TimePeriodKey () {
00035     }
00036
00037     // //////////////////////////////////////
00038     void TimePeriodKey::toStream (std::ostream& ioOut) const {
00039         ioOut << "TimePeriodKey: " << toString() << std::endl;
00040     }
00041
00042     // //////////////////////////////////////
00043     void TimePeriodKey::fromStream (std::istream& ioIn) {
00044     }
00045
00046     // //////////////////////////////////////
00047     const std::string TimePeriodKey::toString() const {
00048         std::ostringstream oStr;
00049         oStr << _timeRangeStart << "-" << _timeRangeEnd;
00050         return oStr.str();
00051     }
00052
00053 }

```

35.427 stdair/bom/TimePeriodKey.hpp File Reference

```

#include <stdair/bom/KeyAbstract.hpp>
#include <stdair/stdair_date_time_types.hpp>

```

Classes

- struct [stdair::TimePeriodKey](#)
Key of time-period.

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.428 TimePeriodKey.hpp

```

00001 #ifndef __STDAIR_BOM_TIMEPERIODKEY_HPP
00002 #define __STDAIR_BOM_TIMEPERIODKEY_HPP
00003
00004 // //////////////////////////////////////

```

```

00005 // Import section
00006 // //////////////////////////////////////
00007 // STDAIR
00008 #include <stdair/bom/KeyAbstract.hpp>
00009 #include <stdair/stdair_date_time_types.hpp>
00010
00011 namespace stdair {
00015     struct TimePeriodKey : public KeyAbstract {
00016
00017     public:
00018         // ////////////////////////////////// Construction //////////////////////////////////
00020         TimePeriodKey (const Time_T&,
00021             const Time_T&);
00023         TimePeriodKey (const TimePeriodKey&);
00025         ~TimePeriodKey ();
00026     private:
00028         TimePeriodKey ();
00029
00030     public:
00031         // ////////////////////////////////// Getter //////////////////////////////////
00035         const Time_T& getTimeRangeStart() const {
00036             return _timeRangeStart;
00037         }
00038
00042         const Time_T& getTimeRangeEnd() const {
00043             return _timeRangeEnd;
00044         }
00045
00046         // ////////////////////////////////// Display support methods //////////////////////////////////
00052         void toStream (std::ostream& ioOut) const;
00053
00059         void fromStream (std::istream& ioIn);
00060
00066         const std::string toString() const;
00067
00068     private:
00069         // ////////////////////////////////// Attributes //////////////////////////////////
00073         Time_T _timeRangeStart;
00074
00078         Time_T _timeRangeEnd;
00079
00080     };
00081
00082 }
00083 #endif // __STDAIR_BOM_TIMEPERIODKEY_HPP

```

35.429 stdair/bom/TimePeriodTypes.hpp File Reference

```

#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>

```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

Typedefs

- typedef std::list< TimePeriod * > [stdair::TimePeriodList_T](#)
- typedef std::map< const MapKey_T, TimePeriod * > [stdair::TimePeriodMap_T](#)
- typedef std::pair< MapKey_T, TimePeriod * > [stdair::TimePeriodWithKey_T](#)
- typedef std::list< TimePeriodWithKey_T > [stdair::TimePeriodDetailedList_T](#)

35.430 TimePeriodTypes.hpp

```

00001 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00002 #ifndef __STDAIR_BOM_TIMEPERIODTYPES_HPP
00003 #define __STDAIR_BOM_TIMEPERIODTYPES_HPP
00004
00005 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00006 // Import section
00007 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // STDAIR
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016     // Forward declarations.
00017     class TimePeriod;
00018
00020     typedef std::list<TimePeriod*> TimePeriodList_T;
00021
00023     typedef std::map<const MapKey_T, TimePeriod*> TimePeriodMap_T;
00024
00026     typedef std::pair<MapKey_T, TimePeriod*> TimePeriodWithKey_T;
00027     typedef std::list<TimePeriodWithKey_T> TimePeriodDetailedList_T;
00028 }
00029 #endif // __STDAIR_BOM_TIMEPERIODTYPES_HPP
00030

```

35.431 stdair/bom/TravelSolutionStruct.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_BookingClass.hpp>
#include <stdair/bom/TravelSolutionStruct.hpp>
#include <stdair/bom/BomKeyManager.hpp>
#include <stdair/bom/ParsedKey.hpp>

```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.432 TravelSolutionStruct.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_BookingClass.hpp>
00009 #include <stdair/bom/TravelSolutionStruct.hpp>
00010 #include <stdair/bom/BomKeyManager.hpp>
00011 #include <stdair/bom/ParsedKey.hpp>
00012
00013 namespace stdair {
00014 // //////////////////////////////////////
00015 TravelSolutionStruct::TravelSolutionStruct() : _chosenFareOption (NULL) {
00016 }
00017
00018 // //////////////////////////////////////
00019 TravelSolutionStruct::~TravelSolutionStruct() {
00020 }
00021
00022 // //////////////////////////////////////
00023 void TravelSolutionStruct::toStream (std::ostream& ioOut) const {
00024     ioOut << describe();
00025 }
00026
00027 // //////////////////////////////////////
00028 void TravelSolutionStruct::fromStream (std::istream& ioIn) {
00029 }
00030
00031 // //////////////////////////////////////
00032 const std::string TravelSolutionStruct::describe() const {
00033     std::ostringstream oStr;
00034
00035     //
00036     oStr << "Segment path: ";
00037     unsigned short idx = 0;
00038     for (SegmentPath_T::const_iterator lItSegmentPath = _segmentPath.begin();
00039          lItSegmentPath != _segmentPath.end(); ++lItSegmentPath, ++idx) {
00040         if (idx != 0) {
00041             oStr << "-";
00042         }
00043         const std::string& lSegmentPathString = *lItSegmentPath;
00044         const stdair::ParsedKey& lSegmentParsedKey =
00045             stdair::BomKeyManager::extractKeys (lSegmentPathString);
00046         const std::string& lSegmentKey = lSegmentParsedKey.toString();
00047         oStr << lSegmentKey;
00048     }
00049     oStr << " ### ";
00050
00051     //
00052     if (_chosenFareOption != NULL) {
00053         oStr << "Chosen fare option: " << _chosenFareOption->describe()
00054             << " ## Among: ";
00055     } else {
00056         oStr << "Fare options: ";
00057     }
00058
00059     //
00060     idx = 0;

```

```

00061     for (FareOptionList_T::const_iterator lItFareOption= _fareOptionList.begin();
00062          lItFareOption != _fareOptionList.end(); ++lItFareOption, ++idx) {
00063         if (idx != 0) {
00064             ostr << " , ";
00065         }
00066         const FareOptionStruct& lFareOption = *lItFareOption;
00067         ostr << lFareOption.describe();
00068     }
00069     return ostr.str();
00070 }
00071
00072 // //////////////////////////////////////
00073 const std::string TravelSolutionStruct::display() const {
00074     std::ostringstream ostr;
00075
00076     // List of segment keys (one per segment)
00077     unsigned short idx = 0;
00078     for (SegmentPath_T::const_iterator itSegPath = _segmentPath.begin();
00079          itSegPath != _segmentPath.end(); ++itSegPath, ++idx) {
00080         if (idx != 0) {
00081             ostr << " ; ";
00082         }
00083         const std::string& lSegmentPathString = *itSegPath;
00084         const stdair::ParsedKey& lSegmentParsedKey =
00085             stdair::BomKeyManager::extractKeys (lSegmentPathString);
00086         const std::string& lSegmentKey = lSegmentParsedKey.toString();
00087         ostr << "[" << idx << "]" " " << lSegmentKey;
00088     }
00089
00090     // List of fare options (for the whole O&D)
00091     ostr << " --- ";
00092     idx = 0;
00093     for (FareOptionList_T::const_iterator itFareOption = _fareOptionList.begin();
00094          itFareOption != _fareOptionList.end(); ++itFareOption, ++idx) {
00095         if (idx != 0) {
00096             ostr << " , ";
00097         }
00098         const FareOptionStruct& lFareOption = *itFareOption;
00099         ostr << lFareOption.display();
00100     }
00101
00102     // List of booking class availability maps: one map per segment
00103     ostr << " --- ";
00104     idx = 0;
00105     for (ClassAvailabilityMapHolder_T::const_iterator itSegMap =
00106          _classAvailabilityMapHolder.begin();
00107          itSegMap != _classAvailabilityMapHolder.end(); ++itSegMap, ++idx) {
00108         if (idx != 0) {
00109             ostr << " ; ";
00110         }
00111         // Retrieve the booking class availability map
00112         const ClassAvailabilityMap_T& lClassAvlMap = *itSegMap;
00113         ostr << "[" << idx << "]" ";
00114
00115         // List (map) of booking class availabilities
00116         unsigned short jdx = 0;
00117         for (ClassAvailabilityMap_T::const_iterator itClass = lClassAvlMap.begin();
00118              itClass != lClassAvlMap.end(); ++itClass, ++jdx) {

```

```

00120         if (jdx != 0) {
00121             ostr << " ";
00122         }
00123         const ClassCode_T& lClassCode = itClass->first;
00124         const Availability_T& lAvl = itClass->second;
00125         ostr << lClassCode << ":" << lAvl;
00126     }
00127 }
00128
00129 return ostr.str();
00130 }
00131
00132 // //////////////////////////////////////
00133 void TravelSolutionStruct::addSegment (const std::string& iKey) {
00134     _segmentPath.push_back (iKey);
00135 }
00136
00137 // //////////////////////////////////////
00138 void TravelSolutionStruct::
00139 addClassAvailabilityMap (const ClassAvailabilityMap_T& iMap) {
00140     _classAvailabilityMapHolder.push_back (iMap);
00141 }
00142
00143 // //////////////////////////////////////
00144 void TravelSolutionStruct::
00145 addClassYieldMap (const ClassYieldMap_T& iMap) {
00146     _classYieldMapHolder.push_back (iMap);
00147 }
00148
00149 // //////////////////////////////////////
00150 void TravelSolutionStruct::
00151 addBidPriceVector (const BidPriceVector_T& iBpv) {
00152     _bidPriceVectorHolder.push_back (iBpv);
00153 }
00154
00155 // //////////////////////////////////////
00156 void TravelSolutionStruct::
00157 addClassBpvMap (const ClassBpvMap_T& iMap) {
00158     _classBpvMapHolder.push_back (iMap);
00159 }
00160
00161 // //////////////////////////////////////
00162 void TravelSolutionStruct::
00163 addFareOption (const FareOptionStruct& iFareOption) {
00164     _fareOptionList.push_back (iFareOption);
00165 }
00166
00167 }

```

35.433 stdair/bom/TravelSolutionStruct.hpp File Reference

```

#include <iosfwd>

#include <string>

#include <vector>

#include <stdair/stdair_basic_types.hpp>

#include <stdair/basic/StructAbstract.hpp>

```

```
#include <stdair/bom/BookingClassTypes.hpp>
#include <stdair/bom/FareOptionStruct.hpp>
#include <stdair/bom/FareOptionTypes.hpp>
#include <stdair/bom/TravelSolutionTypes.hpp>
```

Classes

- struct [stdair::TravelSolutionStruct](#)
Structure holding the elements of a travel solution.

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.434 TravelSolutionStruct.hpp

```
00001 #ifndef __STDAIR_BOM_TRAVELSOLUTIONSTRUCT_HPP
00002 #define __STDAIR_BOM_TRAVELSOLUTIONSTRUCT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 #include <vector>
00011 // StdAir
00012 #include <stdair/stdair_basic_types.hpp>
00013 #include <stdair/basic/StructAbstract.hpp>
00014 #include <stdair/bom/BookingClassTypes.hpp>
00015 #include <stdair/bom/FareOptionStruct.hpp>
00016 #include <stdair/bom/FareOptionTypes.hpp>
00017 #include <stdair/bom/TravelSolutionTypes.hpp>
00018
00019 namespace stdair {
00020
00024     struct TravelSolutionStruct : public StructAbstract {
00025     public:
00026         // //////////// Getters ////////////
00028         const SegmentPath_T& getSegmentPath() const {
00029             return _segmentPath;
00030         }
00031
00033         const ClassAvailabilityMapHolder_T& getClassAvailabilityMapHolder() const {
00034             return _classAvailabilityMapHolder;
00035         }
00036
00038         const ClassYieldMapHolder_T& getClassYieldMapHolder() const {
00039             return _classYieldMapHolder;
00040         }
00041
00043         const BidPriceVectorHolder_T& getBidPriceVectorHolder() const {
```

```

00044         return _bidPriceVectorHolder;
00045     }
00046
00048     const ClassBpvMapHolder_T& getClassBpvMapHolder() const {
00049         return _classBpvMapHolder;
00050     }
00051
00053     const FareOptionList_T& getFareOptionList() const {
00054         return _fareOptionList;
00055     }
00056
00058     FareOptionList_T& getFareOptionListRef() {
00059         return _fareOptionList;
00060     }
00061
00063     const FareOptionStruct& getChosenFareOption() const {
00064         assert (_chosenFareOption != NULL);
00065         return *_chosenFareOption;
00066     }
00067
00068 public:
00069     // //////////// Setters ////////////
00071     void addSegment (const std::string&);
00072
00074     void addClassAvailabilityMap (const ClassAvailabilityMap_T&);
00075
00077     void addClassYieldMap (const ClassYieldMap_T&);
00078
00080     void addBidPriceVector (const BidPriceVector_T&);
00081
00083     void addClassBpvMap (const ClassBpvMap_T&);
00084
00086     void addFareOption (const FareOptionStruct&);
00087
00089     void setChosenFareOption (const FareOptionStruct& iChosenFO) {
00090         _chosenFareOption = &iChosenFO;
00091     }
00092
00093
00094 public:
00095     // //////////// Display support method ////////////
00101     void toStream (std::ostream& ioOut) const;
00102
00107     void fromStream (std::istream& ioIn);
00108
00112     const std::string describe() const;
00113
00117     const std::string display() const;
00118
00119
00120 public:
00121     // //////////// Constructors & Destructor ////////////
00125     TravelSolutionStruct();
00126
00130     ~TravelSolutionStruct();
00131
00132
00133 private:
00134     // ////////////////////////////////// Attributes //////////////////////////////////
00138     SegmentPath_T _segmentPath;
00139
00143     ClassAvailabilityMapHolder_T _classAvailabilityMapHolder;

```

```

00144
00148     ClassYieldMapHolder_T _classYieldMapHolder;
00149
00153     BidPriceVectorHolder_T _bidPriceVectorHolder;
00154
00158     ClassBpvMapHolder_T _classBpvMapHolder;
00159
00163     FareOptionList_T _fareOptionList;
00164
00168     const FareOptionStruct* _chosenFareOption;
00169 };
00170
00171 }
00172 #endif // __STDAIR_BOM_TRAVELSOLUTIONSTRUCT_HPP

```

35.435 stdair/bom/TravelSolutionTypes.hpp File Reference

```

#include <list>
#include <map>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/key_types.hpp>
#include <stdair/stdair_inventory_types.hpp>

```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

Typedefs

- typedef std::list< TravelSolutionStruct > [stdair::TravelSolutionList_T](#)
- typedef KeyList_T [stdair::SegmentPath_T](#)
- typedef std::list< SegmentPath_T > [stdair::SegmentPathList_T](#)
- typedef std::map< const ClassCode_T, Availability_T > [stdair::ClassAvailabilityMap_T](#)
- typedef std::list< ClassAvailabilityMap_T > [stdair::ClassAvailabilityMapHolder_T](#)
- typedef std::map< const ClassCode_T, YieldValue_T > [stdair::ClassYieldMap_T](#)
- typedef std::list< ClassYieldMap_T > [stdair::ClassYieldMapHolder_T](#)
- typedef std::list< BidPriceVector_T > [stdair::BidPriceVectorHolder_T](#)
- typedef std::map< const ClassCode_T, const BidPriceVector_T* > [stdair::ClassBpvMap_T](#)
- typedef std::list< ClassBpvMap_T > [stdair::ClassBpvMapHolder_T](#)

35.436 TravelSolutionTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_TRAVELSOLUTIONTYPES_HPP

```

```

00003 #define __STDAIR_BOM_TRAVELSOLUTIONTYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <list>
00010 #include <map>
00011 // StdAir
00012 #include <stdair/stdair_basic_types.hpp>
00013 #include <stdair/bom/key_types.hpp>
00014 #include <stdair/stdair_inventory_types.hpp> // bid price related types.
00015
00016 namespace stdair {
00017
00018     // Forward declarations.
00019     struct TravelSolutionStruct;
00020
00022     typedef std::list<TravelSolutionStruct> TravelSolutionList_T;
00023
00025     typedef KeyList_T SegmentPath_T;
00026
00028     typedef std::list<SegmentPath_T> SegmentPathList_T;
00029
00031     typedef std::map<const ClassCode_T, Availability_T> ClassAvailabilityMap_T;
00032
00034     typedef std::list<ClassAvailabilityMap_T> ClassAvailabilityMapHolder_T;
00035
00037     typedef std::map<const ClassCode_T, YieldValue_T> ClassYieldMap_T;
00038
00040     typedef std::list<ClassYieldMap_T> ClassYieldMapHolder_T;
00041
00043     typedef std::list<BidPriceVector_T> BidPriceVectorHolder_T;
00044
00046     typedef std::map<const ClassCode_T, const BidPriceVector_T*> ClassBpvMap_T;
00047
00049     typedef std::list<ClassBpvMap_T> ClassBpvMapHolder_T;
00050 }
00051 #endif // __STDAIR_BOM_TRAVELSOLUTIONTYPES_HPP
00052

```

35.437 stdair/bom/VirtualClassStruct.cpp File Reference

```

#include <cassert>

#include <sstream>

#include <stdair/bom/VirtualClassStruct.hpp>

#include <stdair/bom/BookingClass.hpp>

```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

35.438 VirtualClassStruct.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/bom/VirtualClassStruct.hpp>
00009 #include <stdair/bom/BookingClass.hpp>
00010
00011 namespace stdair {
00012
00013 // //////////////////////////////////////
00014 VirtualClassStruct::VirtualClassStruct() : _bookingClass (NULL) {
00015     assert (false);
00016 }
00017
00018 // //////////////////////////////////////
00019 VirtualClassStruct::VirtualClassStruct (const VirtualClassStruct& iVC)
00020     : _bookingClass (iVC._bookingClass), _yield (iVC._yield),
00021     _mean (iVC._mean), _stdDev (iVC._stdDev) {
00022 }
00023
00024 // //////////////////////////////////////
00025 VirtualClassStruct::VirtualClassStruct (BookingClass& ioBookingClass) {
00026     _bookingClass = &ioBookingClass;
00027 }
00028
00029 // //////////////////////////////////////
00030 VirtualClassStruct::~VirtualClassStruct() {
00031     _bookingClass = NULL;
00032 }
00033
00034 // //////////////////////////////////////
00035 void VirtualClassStruct::toStream (std::ostream& ioOut) const {
00036     ioOut << describe();
00037 }
00038
00039 // //////////////////////////////////////
00040 void VirtualClassStruct::fromStream (std::istream& ioIn) {
00041 }
00042
00043 // //////////////////////////////////////
00044 const std::string VirtualClassStruct::describe() const {
00045     std::ostringstream oStr;
00046     oStr << "Yield: " << _yield
00047         << ", Demand N (" << _mean << ", " << _stdDev << ")";
00048     return oStr.str();
00049 }
00050
00051 // //////////////////////////////////////
00052 const GeneratedDemandVector_T& VirtualClassStruct::
00053 getGeneratedDemandVector() const {
00054     assert (_bookingClass != NULL);
00055     return _bookingClass->getGeneratedDemandVector();
00056 }
00057 }

```

35.439 stdair/bom/VirtualClassStruct.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <vector>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/stdair_rm_types.hpp>
#include <stdair/basic/StructAbstract.hpp>

```

Classes

- struct [stdair::VirtualClassStruct](#)

Namespaces

- namespace [stdair](#)

Handle on the StdAir library context.

35.440 VirtualClassStruct.hpp

```

00001 #ifndef __STDAIR_BOM_VIRTUALCLASSSTRUCT_HPP
00002 #define __STDAIR_BOM_VIRTUALCLASSSTRUCT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 #include <vector>
00011 // StdAir
00012 #include <stdair/stdair_basic_types.hpp>
00013 #include <stdair/stdair_inventory_types.hpp>
00014 #include <stdair/stdair_maths_types.hpp>
00015 #include <stdair/stdair_rm_types.hpp>
00016 #include <stdair/basic/StructAbstract.hpp>
00017
00018 namespace stdair {
00019     // Forward declarations.
00020     class BookingClass;
00021
00022     struct VirtualClassStruct : public StructAbstract {
00023     public:
00024         // //////////// Getters ////////////
00027         const Yield_T& getYield() const {
00028             return _yield;
00029         }
00030

```

```

00032     const MeanValue_T& getMean() const {
00033         return _mean;
00034     }
00035
00037     const StdDevValue_T& getStdDev() const {
00038         return _stdDev;
00039     }
00040
00042     const BookingLimit_T& getCumulatedBookingLimit () const {
00043         return _cumulatedBookingLimit;
00044     }
00045
00047     const ProtectionLevel_T& getCumulatedProtection () const {
00048         return _cumulatedProtection;
00049     }
00050
00052     const GeneratedDemandVector_T& getGeneratedDemandVector () const;
00053
00054 public:
00055     // //////////// Setters ////////////
00057     void setYield (const Yield_T& iYield) {
00058         _yield = iYield;
00059     }
00060
00062     void setMean (const MeanValue_T& iMean) {
00063         _mean = iMean;
00064     }
00065
00067     void setStdDev (const StdDevValue_T& iStdDev) {
00068         _stdDev = iStdDev;
00069     }
00070
00072     void setCumulatedBookingLimit (const BookingLimit_T& iBL) {
00073         _cumulatedBookingLimit = iBL;
00074     }
00075
00077     void setCumulatedProtection (const ProtectionLevel_T& iP) {
00078         _cumulatedProtection = iP;
00079     }
00080
00081 public:
00082     // //////////// Display support method ////////////
00085     void toStream (std::ostream& ioOut) const;
00086
00089     void fromStream (std::istream& ioIn);
00090
00092     const std::string describe() const;
00093
00094
00095 public:
00096     // //////////// Constructors & Destructor ////////////
00098     VirtualClassStruct (const VirtualClassStruct&);
00100     VirtualClassStruct (BookingClass&);
00102     ~VirtualClassStruct ();
00103
00104 private:
00106     VirtualClassStruct ();
00107
00108
00109 private:
00110     // //////////// Attributes ////////////
00112     BookingClass* _bookingClass;

```

```

00113
00115     Yield_T _yield;
00116
00118     MeanValue_T _mean;
00119
00121     StdDevValue_T _stdDev;
00122
00124     BookingLimit_T _cumulatedBookingLimit;
00125
00127     ProtectionLevel_T _cumulatedProtection;
00128 };
00129
00130 }
00131 #endif // __STDAIR_BOM_VIRTUALCLASSSTRUCT_HPP

```

35.441 stdair/bom/VirtualClassTypes.hpp File Reference

```

#include <list>

#include <map>

#include <stdair/stdair_basic_types.hpp>

```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

Typedefs

- typedef std::list< VirtualClassStruct > [stdair::VirtualClassList_T](#)
- typedef std::map< const Yield_T, VirtualClassStruct > [stdair::VirtualClassMap_T](#)

35.442 VirtualClassTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_VIRTUALCLASSTYPES_HPP
00003 #define __STDAIR_BOM_VIRTUALCLASSTYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <list>
00010 #include <map>
00011 // STDAIR
00012 #include <stdair/stdair_basic_types.hpp>
00013
00014 namespace stdair {
00015
00016     // Forward declarations.
00017     struct VirtualClassStruct;
00018

```

```

00020     typedef std::list<VirtualClassStruct> VirtualClassList_T;
00021
00023     typedef std::map<const Yield_T, VirtualClassStruct> VirtualClassMap_T;
00024 }
00025 #endif // __STDAIR_BOM_VIRTUALCLASSTYPES_HPP
00026

```

35.443 stdair/bom/YieldFeatures.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_Request.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/YieldFeatures.hpp>

```

Namespaces

- namespace `stdair`
Handle on the *StdAir* library context.

35.444 YieldFeatures.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_Request.hpp>
00009 #include <stdair/service/Logger.hpp>
00010 #include <stdair/bom/YieldFeatures.hpp>
00011
00012 namespace stdair {
00013
00014     // //////////////////////////////////////
00015     YieldFeatures::YieldFeatures()
00016         : _key (TRIP_TYPE_ONE_WAY,
00017              DEFAULT_PREFERRED_CABIN),
00018         _parent (NULL) {
00019         // That constructor is used by the serialisation process
00020     }
00021
00022     // //////////////////////////////////////
00023     YieldFeatures::YieldFeatures (const YieldFeatures& iFeatures)
00024         : _key (iFeatures.getKey()), _parent (NULL) {
00025         assert (false);
00026     }
00027
00028     // //////////////////////////////////////
00029     YieldFeatures::YieldFeatures (const Key_T& iKey)
00030         : _key (iKey), _parent (NULL) {

```

```

00031     }
00032
00033     // //////////////////////////////////////
00034     YieldFeatures::~YieldFeatures () {
00035     }
00036
00037     // //////////////////////////////////////
00038     std::string YieldFeatures::toString() const {
00039         std::ostringstream ostr;
00040         ostr << describeKey();
00041         return ostr.str();
00042     }
00043
00044     // //////////////////////////////////////
00045     bool YieldFeatures::
00046     isTripTypeValid (const TripType_T& iBookingRequestTripType) const {
00047         bool oIsTripTypeValidFlag = true;
00048
00049         // Check whether the yield trip type is the same as the booking request
00050         // trip type
00051         const TripType_T& lYieldTripType = getTripType();
00052         if (iBookingRequestTripType == lYieldTripType) {
00053             // One way case
00054             return oIsTripTypeValidFlag;
00055         }
00056
00057         if (iBookingRequestTripType == TRIP_TYPE_INBOUND ||
00058             iBookingRequestTripType == TRIP_TYPE_OUTBOUND) {
00059             // Round trip case.
00060             if (lYieldTripType == TRIP_TYPE_ROUND_TRIP) {
00061                 return oIsTripTypeValidFlag;
00062             }
00063         }
00064
00065         oIsTripTypeValidFlag = false;
00066         return false;
00067     }
00068
00069 }
00070

```

35.445 stdair/bom/YieldFeatures.hpp File Reference

```

#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/YieldFeaturesKey.hpp>
#include <stdair/bom/YieldFeaturesTypes.hpp>

```

Classes

- class [stdair::YieldFeatures](#)
Class representing the actual attributes for a yield date-period.

Namespaces

- namespace [stdair](#)

Handle on the StdAir library context.

35.446 YieldFeatures.hpp

```

00001 #ifndef __STDAIR_BOM_YIELDFEATURES_HPP
00002 #define __STDAIR_BOM_YIELDFEATURES_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/bom/BomAbstract.hpp>
00009 #include <stdair/bom/YieldFeaturesKey.hpp>
00010 #include <stdair/bom/YieldFeaturesTypes.hpp>
00011
00012 // Forward declaration
00013 namespace stdair {
00014
00015     class YieldFeatures : public BomAbstract {
00020     template <typename BOM> friend class FacBom;
00021     friend class FacBomManager;
00022
00023     public:
00024         // ////////// Type definitions
00028         typedef YieldFeaturesKey Key_T;
00029
00030     public:
00031         // ////////// Display support methods //////////
00037         void toStream (std::ostream& ioOut) const {
00038             ioOut << toString();
00039         }
00040
00046         void fromStream (std::istream& ioIn) {
00047         }
00048
00052         std::string toString() const;
00053
00057         const std::string describeKey() const {
00058             return _key.toString();
00059         }
00060
00061     public:
00062         // ////////// Getters //////////
00066         const Key_T& getKey() const {
00067             return _key;
00068         }
00069
00073         BomAbstract* const getParent() const {
00074             return _parent;
00075         }
00076
00080         const HolderMap_T& getHolderMap() const {
00081             return _holderMap;
00082         }
00083
00087         const CabinCode_T& getCabinCode() const {
00088             return _key.getCabinCode();
00089         }
00090
00094         const TripType_T& getTripType() const {

```

```

00095         return _key.getTripType();
00096     }
00097
00098
00099     public:
00100         // //////////////// Business methods ////////////////
00101         bool isTripTypeValid (const TripType_T&) const;
00102
00103
00104     protected:
00105         // //////////////// Constructors and destructors ////////////////
00106         YieldFeatures (const Key_T&);
00107
00108         virtual ~YieldFeatures();
00109
00110     private:
00111         YieldFeatures();
00112
00113         YieldFeatures (const YieldFeatures&);
00114
00115
00116     protected:
00117         // //////////////// Attributes ////////////////
00118         Key_T _key;
00119
00120         BomAbstract* _parent;
00121
00122         HolderMap_T _holderMap;
00123     };
00124 }
00125 #endif // __STDAIR_BOM_YIELDFEATURES_HPP
00126

```

35.447 stdair/bom/YieldFeaturesKey.cpp File Reference

```

#include <ostream>
#include <sstream>
#include <stdair/basic/BasConst_Request.hpp>
#include <stdair/bom/YieldFeaturesKey.hpp>

```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

35.448 YieldFeaturesKey.cpp

```

00001 // ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <ostream>

```

```

00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_Request.hpp>
00009 #include <stdair/bom/YieldFeaturesKey.hpp>
00010
00011 namespace stdair {
00012
00013 // //////////////////////////////////////
00014 YieldFeaturesKey::YieldFeaturesKey()
00015     : _tripType (TRIP_TYPE_ONE_WAY),
00016       _cabinCode (DEFAULT_PREFERRED_CABIN) {
00017     assert (false);
00018 }
00019
00020 // //////////////////////////////////////
00021 YieldFeaturesKey::YieldFeaturesKey (const stdair::TripType_T& iTripType,
00022                                     const stdair::CabinCode_T& iCabin)
00023     : _tripType (iTripType), _cabinCode (iCabin) {
00024 }
00025
00026 // //////////////////////////////////////
00027 YieldFeaturesKey::YieldFeaturesKey (const YieldFeaturesKey& iKey)
00028     : _tripType (iKey.getTripType()), _cabinCode (iKey.getCabinCode()) {
00029 }
00030
00031 // //////////////////////////////////////
00032 YieldFeaturesKey::~YieldFeaturesKey () {
00033 }
00034
00035 // //////////////////////////////////////
00036 void YieldFeaturesKey::toStream (std::ostream& ioOut) const {
00037     ioOut << "YieldFeaturesKey: " << toString() << std::endl;
00038 }
00039
00040 // //////////////////////////////////////
00041 void YieldFeaturesKey::fromStream (std::istream& ioIn) {
00042 }
00043
00044 // //////////////////////////////////////
00045 const std::string YieldFeaturesKey::toString() const {
00046     std::ostringstream oStr;
00047     oStr << _tripType << " -- " << _cabinCode;
00048     return oStr.str();
00049 }
00050
00051 }

```

35.449 stdair/bom/YieldFeaturesKey.hpp File Reference

```

#include <stdair/bom/KeyAbstract.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/stdair_inventory_types.hpp>

```

Classes

- struct `stdair::YieldFeaturesKey`

Key of date-period.

Namespaces

- namespace `stdair`

Handle on the StdAir library context.

35.450 YieldFeaturesKey.hpp

```

00001 #ifndef __STDAIR_BOM_YIELDFEATURESKEY_HPP
00002 #define __STDAIR_BOM_YIELDFEATURESKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/bom/KeyAbstract.hpp>
00009 #include <stdair/stdair_date_time_types.hpp>
00010 #include <stdair/stdair_demand_types.hpp>
00011 #include <stdair/stdair_inventory_types.hpp>
00012
00013 namespace stdair {
00014
00015     struct YieldFeaturesKey : public KeyAbstract {
00016     public:
00017         // ////////////////////////////////// Construction //////////////////////////////////
00018         YieldFeaturesKey (const TripType_T&, const CabinCode_T&);
00019         YieldFeaturesKey (const YieldFeaturesKey&);
00020         ~YieldFeaturesKey ();
00021     private:
00022         YieldFeaturesKey ();
00023
00024     public:
00025         // ////////////////////////////////// Getters //////////////////////////////////
00026         const TripType_T& getTripType() const {
00027             return _tripType;
00028         }
00029         const CabinCode_T& getCabinCode() const {
00030             return _cabinCode;
00031         }
00032     public:
00033         // ////////////////////////////////// Display support methods //////////////////////////////////
00034         void toStream (std::ostream& ioOut) const;
00035         void fromStream (std::istream& ioIn);
00036         const std::string toString() const;
00037     private:
00038         // ////////////////////////////////// Attributes //////////////////////////////////
00039         TripType_T _tripType;
00040
00041     };
00042
00043 }

```

```

00086     CabinCode_T _cabinCode;
00087 };
00088
00089 }
00090 #endif // __STDAIR_BOM_YIELDFEATURESKEY_HPP

```

35.451 stdair/bom/YieldFeaturesTypes.hpp File Reference

```

#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>

```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

Typedefs

- typedef std::list< YieldFeatures * > [stdair::YieldFeaturesList_T](#)
- typedef std::map< const MapKey_T, YieldFeatures * > [stdair::YieldFeaturesMap_T](#)
- typedef std::pair< MapKey_T, YieldFeatures * > [stdair::YieldFeaturesWithKey_T](#)
- typedef std::list< YieldFeaturesWithKey_T > [stdair::YieldFeaturesDetailedList_T](#)

35.452 YieldFeaturesTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_YIELDFEATURESTYPES_HPP
00003 #define __STDAIR_BOM_YIELDFEATURESTYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // STDAIR
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016     // Forward declarations.
00017     class YieldFeatures;
00018
00019     typedef std::list<YieldFeatures*> YieldFeaturesList_T;
00020
00021     typedef std::map<const MapKey_T, YieldFeatures*> YieldFeaturesMap_T;
00022
00023     typedef std::pair<MapKey_T, YieldFeatures*> YieldFeaturesWithKey_T;
00024
00025     typedef std::list<YieldFeaturesWithKey_T> YieldFeaturesDetailedList_T;

```

```

00028 }
00029 #endif // __STDAIR_BOM_YIELDFEATURESTYPES_HPP
00030

```

35.453 stdair/bom/YieldStore.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/bom/YieldStore.hpp>

```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.454 YieldStore.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/bom/YieldStore.hpp>
00009
00010 namespace stdair {
00011
00012     // //////////////////////////////////////
00013     YieldStore::YieldStore (const Key_T& iKey) : _key (iKey), _parent (NULL) {
00014     }
00015
00016     // //////////////////////////////////////
00017     YieldStore::~YieldStore () {
00018     }
00019
00020     // //////////////////////////////////////
00021     std::string YieldStore::toString() const {
00022         std::ostringstream oStr;
00023         oStr << _key.toString();
00024         return oStr.str();
00025     }
00026
00027 }
00028

```

35.455 stdair/bom/YieldStore.hpp File Reference

```

#include <string>
#include <stdair/stdair_inventory_types.hpp>

```

```
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/YieldStoreKey.hpp>
#include <stdair/bom/YieldStoreTypes.hpp>
```

Classes

- class [stdair::YieldStore](#)

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.456 YieldStore.hpp

```
00001 #ifndef __STDAIR_BOM_YIELDSTORE_HPP
00002 #define __STDAIR_BOM_YIELDSTORE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/stdair_inventory_types.hpp>
00011 #include <stdair/bom/BomAbstract.hpp>
00012 #include <stdair/bom/YieldStoreKey.hpp>
00013 #include <stdair/bom/YieldStoreTypes.hpp>
00014
00015 namespace stdair {
00016
00017     class YieldStore : public BomAbstract {
00018     template <typename BOM> friend class FacBom;
00019     friend class FacBomManager;
00020
00021     public :
00022         // Type definitions
00023         typedef YieldStoreKey Key_T;
00024
00025     public:
00026         // /////////// Display support methods ///////////
00027         void toStream (std::ostream& ioOut) const { ioOut << toString(); }
00028
00029         BomAbstract* const getParent() const { return _parent; }
00030
00031         void fromStream (std::istream& ioIn) { }
00032
00033         std::string toString() const;
00034
00035         const std::string describeKey() const { return _key.toString(); }
00036
00037     public:
00038         // /////////// Getters ///////////
00039         const Key_T& getKey() const { return _key; }
```

```

00050
00052     const AirlineCode_T& getAirlineCode () const {
00053         return _key.getAirlineCode();
00054     }
00055
00056 protected:
00058     YieldStore (const Key_T&);
00059     YieldStore (const YieldStore&);
00061     ~YieldStore();
00062
00063 protected:
00064     // Attributes
00066     Key_T _key;
00067     BomAbstract* _parent;
00068 };
00069
00070 }
00071 #endif // __STDAIR_BOM_YIELDSTORE_HPP
00072

```

35.457 stdair/bom/YieldStoreKey.cpp File Reference

```
#include <stdair/bom/YieldStoreKey.hpp>
```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.458 YieldStoreKey.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // StdAir
00005 #include <stdair/bom/YieldStoreKey.hpp>
00006
00007 namespace stdair {
00008
00009     // //////////////////////////////////////
00010     YieldStoreKey::YieldStoreKey (const AirlineCode_T& iAirlineCode)
00011         : _airlineCode (iAirlineCode) {
00012     }
00013     // //////////////////////////////////////
00014     YieldStoreKey::YieldStoreKey (const YieldStoreKey& iKey)
00015         : _airlineCode (iKey._airlineCode) {
00016     }
00017
00018     // //////////////////////////////////////
00019     YieldStoreKey::~YieldStoreKey () {
00020     }
00021
00022     // //////////////////////////////////////
00023     void YieldStoreKey::toStream (std::ostream& ioOut) const {
00024         ioOut << "YieldStoreKey: " << toString() << std::endl;
00025     }

```

```

00026
00027 // //////////////////////////////////////
00028 void YieldStoreKey::fromStream (std::istream& ioIn) {
00029 }
00030
00031 // //////////////////////////////////////
00032 const std::string YieldStoreKey::toString() const {
00033     std::ostringstream oStr;
00034     oStr << _airlineCode;
00035     return oStr.str();
00036 }
00037
00038 }

```

35.459 stdair/bom/YieldStoreKey.hpp File Reference

```

#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>

```

Classes

- struct [stdair::YieldStoreKey](#)

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.460 YieldStoreKey.hpp

```

00001 #ifndef __STDAIR_BOM_YIELDSTOREKEY_HPP
00002 #define __STDAIR_BOM_YIELDSTOREKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_inventory_types.hpp>
00009 #include <stdair/bom/KeyAbstract.hpp>
00010
00011 namespace stdair {
00012
00013     struct YieldStoreKey : public KeyAbstract {
00014
00015     private:
00016         // ////////////////////////////////// Default constructor //////////////////////////////////
00017         YieldStoreKey () { };
00018
00019     public:
00020         // ////////////////////////////////// Construction //////////////////////////////////
00021         YieldStoreKey (const AirlineCode_T& iAirlineCode);
00022         YieldStoreKey (const YieldStoreKey&);
00023         ~YieldStoreKey ();

```

```

00027
00028     // //////////// Getters ////////////
00030     const AirlineCode_T& getAirlineCode() const {
00031         return _airlineCode;
00032     }
00033
00034     // //////////// Display support methods ////////////
00037     void toStream (std::ostream& ioOut) const;
00038
00041     void fromStream (std::istream& ioIn);
00042
00048     const std::string toString() const;
00049
00050 private:
00051     // Attributes
00053     AirlineCode_T _airlineCode;
00054 };
00055
00056 }
00057 #endif // __STDAIR_BOM_YIELDSTOREKEY_HPP

```

35.461 stdair/bom/YieldStoreTypes.hpp File Reference

```

#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>

```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

Typedefs

- typedef `std::list< YieldStore * >` `stdair::YieldStoreList_T`
- typedef `std::map< const MapKey_T, YieldStore * >` `stdair::YieldStoreMap_T`

35.462 YieldStoreTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_YIELDSTORETYPES_HPP
00003 #define __STDAIR_BOM_YIELDSTORETYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // StdAir
00012 #include <stdair/bom/key_types.hpp>
00013

```

```

00014 namespace stdair {
00015
00016     // Forward declarations.
00017     class YieldStore;
00018
00020     typedef std::list<YieldStore*> YieldStoreList_T;
00021
00023     typedef std::map<const MapKey_T, YieldStore*> YieldStoreMap_T;
00024
00025 }
00026 #endif // __STDAIR_BOM_YIELDSTORETYPES_HPP
00027

```

35.463 stdair/command/CmdAbstract.cpp File Reference

```
#include <stdair/command/CmdAbstract.hpp>
```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.464 CmdAbstract.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // StdAir
00005 #include <stdair/command/CmdAbstract.hpp>
00006
00007 namespace stdair {
00008
00009 }

```

35.465 stdair/command/CmdAbstract.hpp File Reference

Classes

- class [stdair::CmdAbstract](#)

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.466 CmdAbstract.hpp

```

00001 #ifndef __STDAIR_CMD_CMDABSTRACT_HPP
00002 #define __STDAIR_CMD_CMDABSTRACT_HPP

```

```
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007
00008 namespace stdair {
00009
00011     class CmdAbstract {
00012     public:
00013
00014     };
00015
00016 }
00017 #endif // __STDAIR_CMD_CMDABSTRACT_HPP
```

35.467 stdair/command/CmdBomManager.cpp File Reference

35.468 CmdBomManager.cpp

```
00001
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <cassert>
00010 #include <sstream>
00011 // StdAir
00012 #include <stdair/basic/BasConst_General.hpp>
00013 #include <stdair/basic/BasConst_DefaultObject.hpp>
00014 #include <stdair/basic/BasConst_Request.hpp>
00015 #include <stdair/basic/BasConst_Inventory.hpp>
00016 #include <stdair/bom/BomRetriever.hpp>
00017 #include <stdair/bom/BomRoot.hpp>
00018 #include <stdair/bom/Inventory.hpp>
00019 #include <stdair/bom/FlightDate.hpp>
00020 #include <stdair/bom/LegDate.hpp>
00021 #include <stdair/bom/LegCabin.hpp>
00022 #include <stdair/bom/SegmentDate.hpp>
00023 #include <stdair/bom/SegmentCabin.hpp>
00024 #include <stdair/bom/FareFamily.hpp>
00025 #include <stdair/bom/BookingClass.hpp>
00026 #include <stdair/bom/AirportPair.hpp>
00027 #include <stdair/bom/PosChannel.hpp>
00028 #include <stdair/bom/DatePeriod.hpp>
00029 #include <stdair/bom/TimePeriod.hpp>
00030 #include <stdair/bom/FareFeatures.hpp>
00031 #include <stdair/bom/YieldFeatures.hpp>
00032 #include <stdair/bom/AirlineClassList.hpp>
00033 #include <stdair/bom/BomManager.hpp>
00034 #include <stdair/bom/TravelSolutionStruct.hpp>
00035 #include <stdair/bom/BookingRequestStruct.hpp>
00036 #include <stdair/factory/FacBomManager.hpp>
00037 #include <stdair/factory/FacBom.hpp>
00038 #include <stdair/command/CmdBomManager.hpp>
00039 #include <stdair/service/Logger.hpp>
00040 #include <stdair/bom/OnDDate.hpp>
00041 #include <stdair/bom/SegmentPeriod.hpp>
00042 #include <stdair/bom/FlightPeriod.hpp>
00043
00044 namespace stdair {
```

```

00045
00046 // //////////////////////////////////////
00047 void CmdBomManager::buildSampleBom (BomRoot& ioBomRoot) {
00048
00049     // DEBUG
00050     STDAIR_LOG_DEBUG ("StdAir is building the BOM tree from built-in "
00051         << "specifications.");
00052
00053     // ===== Basic Bom Tree =====
00054     // Build the inventory (flight-dates) and the schedule (flight period) parts.
00055
00056     buildSampleInventorySchedule (ioBomRoot);
00057
00058     // Build the pricing (fare rules) and revenue accounting (yields) parts.
00059     buildSamplePricing (ioBomRoot);
00060
00061     // ===== Partnership Bom Tree =====
00062     // Build the inventory (flight-dates) and the schedule (flight period) parts.
00063
00064     buildPartnershipsSampleInventoryAndRM (ioBomRoot);
00065
00066     // Build the pricing (fare rules) and revenue accounting (yields) parts.
00067     buildPartnershipsSamplePricing (ioBomRoot);
00068
00069     // Build a dummy inventory, needed by RMOL.
00070     buildCompleteDummyInventory (ioBomRoot);
00071 }
00072
00073 // //////////////////////////////////////
00074 void CmdBomManager::buildSampleInventorySchedule (BomRoot& ioBomRoot) {
00075
00076     // Inventory
00077     // Step 0.1: Inventory level
00078     // Create an Inventory for BA
00079     const InventoryKey lBAKey ("BA");
00080     Inventory& lBAInv = FacBom<Inventory>::instance().create (lBAKey);
00081     FacBomManager::addToListAndMap (ioBomRoot, lBAInv);
00082     FacBomManager::linkWithParent (ioBomRoot, lBAInv);
00083
00084     // Create an Inventory for AF
00085     const InventoryKey lAFKey ("AF");
00086     Inventory& lAFInv = FacBom<Inventory>::instance().create (lAFKey);
00087     FacBomManager::addToListAndMap (ioBomRoot, lAFInv);
00088     FacBomManager::linkWithParent (ioBomRoot, lAFInv);
00089
00090     // BA
00091     // Step 0.2: Flight-date level
00092     // Create a FlightDate (BA9/10-JUN-2011) for BA's Inventory
00093     FlightNumber_T lFlightNumber = 9;
00094     Date_T lDate (2011, 6, 10);
00095     FlightDateKey lFlightDateKey (lFlightNumber, lDate);
00096
00097     FlightDate& lBA9_20110610_FD =
00098         FacBom<FlightDate>::instance().create (lFlightDateKey);
00099     FacBomManager::addToListAndMap (lBAInv, lBA9_20110610_FD);
00100     FacBomManager::linkWithParent (lBAInv, lBA9_20110610_FD);
00101
00102     // Display the flight-date
00103     STDAIR_LOG_DEBUG ("FlightDate: " << lBA9_20110610_FD.toString());
00104
00105     // Step 0.3: Segment-date level
00106     // Create a first SegmentDate (LHR-SYD) for BA's Inventory

```

```

00105     // See http://www.britishairways.com/travel/flightinformation/public/fr_fr?&C
        arrier=BA&FlightNumber=0009&from=LHR&to=SYD&depDate=100611&SellingClass=O
00106     const AirportCode_T lLHR ("LHR");
00107     const AirportCode_T lSYD ("SYD");
00108     const DateOffset_T l1Day (1);
00109     const DateOffset_T l2Days (2);
00110     const Duration_T l2135 (21, 45, 0);
00111     const Duration_T l0610 (6, 10, 0);
00112     const Duration_T l2205 (22, 05, 0);
00113     SegmentDateKey lSegmentDateKey (lLHR, lSYD);
00114
00115     SegmentDate& lLHRSYDSegment =
00116         FacBom<SegmentDate>::instance().create (lSegmentDateKey);
00117     FacBomManager::addToListAndMap (lBA9_20110610_FD, lLHRSYDSegment);
00118     FacBomManager::linkWithParent (lBA9_20110610_FD, lLHRSYDSegment);
00119
00120     // Fill the SegmentDate content
00121     lLHRSYDSegment.setBoardingDate (lDate);
00122     lLHRSYDSegment.setOffDate (lDate + l2Days);
00123     lLHRSYDSegment.setBoardingTime (l2135);
00124     lLHRSYDSegment.setOffTime (l0610);
00125     lLHRSYDSegment.setElapsedTime (l2135);
00126
00127     // Display the segment-date
00128     // STDAIR_LOG_DEBUG ("SegmentDate: " << lLHRSYDSegment);
00129
00130     // Create a second SegmentDate (LHR-BKK) for BA's Inventory
00131     // See http://www.britishairways.com/travel/flightinformation/public/fr_fr?&C
        arrier=BA&FlightNumber=0009&from=LHR&to=BKK&depDate=100611&SellingClass=O
00132     const AirportCode_T lBKK ("BKK");
00133     const Duration_T l1540 (15, 40, 0);
00134     const Duration_T l1105 (11, 5, 0);
00135     lSegmentDateKey = SegmentDateKey (lLHR, lBKK);
00136
00137     SegmentDate& lLHRBKKSegment =
00138         FacBom<SegmentDate>::instance().create (lSegmentDateKey);
00139     FacBomManager::addToListAndMap (lBA9_20110610_FD, lLHRBKKSegment);
00140     FacBomManager::linkWithParent (lBA9_20110610_FD, lLHRBKKSegment);
00141
00142     // Fill the SegmentDate content
00143     lLHRBKKSegment.setBoardingDate (lDate);
00144     lLHRBKKSegment.setOffDate (lDate + l1Day);
00145     lLHRBKKSegment.setBoardingTime (l2135);
00146     lLHRBKKSegment.setOffTime (l1540);
00147     lLHRBKKSegment.setElapsedTime (l1105);
00148
00149     // Display the segment-date
00150     // STDAIR_LOG_DEBUG ("SegmentDate: " << lLHRBKKSegment);
00151
00152     // Create a third SegmentDate (BKK-SYD) for BA's Inventory
00153     // See http://www.britishairways.com/travel/flightinformation/public/fr_fr?&C
        arrier=BA&FlightNumber=0009&from=BKK&to=SYD&depDate=110611&SellingClass=O
00154     const Duration_T l1705 (17, 5, 0);
00155     const Duration_T l0905 (9, 5, 0);
00156     lSegmentDateKey = SegmentDateKey (lBKK, lSYD);
00157
00158     SegmentDate& lBKKSYSYDSegment =
00159         FacBom<SegmentDate>::instance().create (lSegmentDateKey);
00160     FacBomManager::addToListAndMap (lBA9_20110610_FD, lBKKSYSYDSegment);
00161     FacBomManager::linkWithParent (lBA9_20110610_FD, lBKKSYSYDSegment);
00162
00163     // Fill the SegmentDate content

```

```

00164     lBKKSYSYDSegment.setBoardingDate (lDate + 11Day);
00165     lBKKSYSYDSegment.setOffDate (lDate + 12Days);
00166     lBKKSYSYDSegment.setBoardingTime (11705);
00167     lBKKSYSYDSegment.setOffTime (11540);
00168     lBKKSYSYDSegment.setElapsedTime (10905);
00169
00170     // Display the segment-date
00171     // STDAIR_LOG_DEBUG ("SegmentDate: " << lBKKSYSYDSegment);
00172
00173     // Step 0.4: Leg-date level
00174     // Create a first LegDate (LHR) for BA's Inventory
00175     LegDateKey lLegDateKey (lLHR);
00176
00177     LegDate& lLHRLeg = FacBom<LegDate>::instance().create (lLegDateKey);
00178     FacBomManager::addToListAndMap (lBA9_20110610_FD, lLHRLeg);
00179     FacBomManager::linkWithParent (lBA9_20110610_FD, lLHRLeg);
00180
00181     // Fill the LegDate content
00182     lLHRLeg.setOffPoint (lBKK);
00183     lLHRLeg.setBoardingDate (lDate);
00184     lLHRLeg.setOffDate (lDate + 11Day);
00185     lLHRLeg.setBoardingTime (12135);
00186     lLHRLeg.setOffTime (11540);
00187     lLHRLeg.setElapsedTime (11105);
00188
00189     // Display the leg-date
00190     // STDAIR_LOG_DEBUG ("LegDate: " << lLHRLeg.toString());
00191
00192     // Create a second LegDate (BKK)
00193     lLegDateKey = LegDateKey (lBKK);
00194
00195     LegDate& lBKKLeg = FacBom<LegDate>::instance().create (lLegDateKey);
00196     FacBomManager::addToListAndMap (lBA9_20110610_FD, lBKKLeg);
00197     FacBomManager::linkWithParent (lBA9_20110610_FD, lBKKLeg);
00198
00199     // Display the leg-date
00200     // STDAIR_LOG_DEBUG ("LegDate: " << lBKKLeg.toString());
00201
00202     // Fill the LegDate content
00203     lBKKLeg.setOffPoint (lSYD);
00204     lBKKLeg.setBoardingDate (lDate + 11Day);
00205     lBKKLeg.setOffDate (lDate + 12Days);
00206     lBKKLeg.setBoardingTime (11705);
00207     lBKKLeg.setOffTime (11540);
00208     lBKKLeg.setElapsedTime (10905);
00209
00210     // Link the segment-dates with the leg-dates
00211     FacBomManager::addToListAndMap (lLHRLeg, lLHRSYDSegment);
00212     FacBomManager::addToListAndMap (lLHRLeg, lLHRBKKSegment);
00213     FacBomManager::addToListAndMap (lBKKLeg, lLHRSYDSegment);
00214     FacBomManager::addToListAndMap (lBKKLeg, lBKKSYSYDSegment);
00215     FacBomManager::addToListAndMap (lLHRSYDSegment, lLHRLeg);
00216     FacBomManager::addToListAndMap (lLHRBKKSegment, lLHRLeg);
00217     FacBomManager::addToListAndMap (lLHRSYDSegment, lBKKLeg);
00218     FacBomManager::addToListAndMap (lBKKSYSYDSegment, lBKKLeg);
00219
00220
00221     // Step 0.5: segment-cabin level
00222     // Create a SegmentCabin (Y) for the Segment LHR-BKK of BA's Inventory
00223     const CabinCode_T lY ("Y");
00224     SegmentCabinKey lYSegmentCabinKey (lY);
00225

```

```

00226     SegmentCabin& lLHRBKKSegmentYCabin =
00227         FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
00228     FacBomManager::addToListAndMap (lLHRBKKSegment, lLHRBKKSegmentYCabin);
00229     FacBomManager::linkWithParent (lLHRBKKSegment, lLHRBKKSegmentYCabin);
00230
00231     // Display the segment-cabin
00232     // STDAIR_LOG_DEBUG ("SegmentCabin: " << lLHRBKKSegmentYCabin.toString());
00233
00234     // Create a SegmentCabin (Y) of the Segment BKK-SYD;
00235     SegmentCabin& lBKKSYSYDSegmentYCabin =
00236         FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
00237     FacBomManager::addToListAndMap (lBKKSYSYDSegment, lBKKSYSYDSegmentYCabin);
00238     FacBomManager::linkWithParent (lBKKSYSYDSegment, lBKKSYSYDSegmentYCabin);
00239
00240
00241     // Display the segment-cabin
00242     // STDAIR_LOG_DEBUG ("SegmentCabin: " << lBKKSYSYDSegmentYCabin.toString());
00243
00244     // Create a SegmentCabin (Y) of the Segment LHR-SYD;
00245     SegmentCabin& lLHRSYDSegmentYCabin =
00246         FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
00247     FacBomManager::addToListAndMap (lLHRSYDSegment, lLHRSYDSegmentYCabin);
00248     FacBomManager::linkWithParent (lLHRSYDSegment, lLHRSYDSegmentYCabin);
00249
00250     // Display the segment-cabin
00251     // STDAIR_LOG_DEBUG ("SegmentCabin: " << lLHRSYDSegmentYCabin.toString());
00252
00253     // Step 0.6: leg-cabin level
00254     // Create a LegCabin (Y) for the Leg LHR-BKK on BA's Inventory
00255     LegCabinKey lYLegCabinKey (lY);
00256
00257     LegCabin& lLHRLegYCabin =
00258         FacBom<LegCabin>::instance().create (lYLegCabinKey);
00259     FacBomManager::addToListAndMap (lLHRLeg, lLHRLegYCabin);
00260     FacBomManager::linkWithParent (lLHRLeg, lLHRLegYCabin);
00261
00262     // Display the leg-cabin
00263     // STDAIR_LOG_DEBUG ("LegCabin: " << lLHRLegYCabin.toString());
00264
00265     // Create a LegCabin (Y) for the Leg BKK-SYD
00266     LegCabin& lBKKLegYCabin =
00267         FacBom<LegCabin>::instance().create (lYLegCabinKey);
00268     FacBomManager::addToListAndMap (lBKKLeg, lBKKLegYCabin);
00269     FacBomManager::linkWithParent (lBKKLeg, lBKKLegYCabin);
00270     // Display the leg-cabin
00271     // STDAIR_LOG_DEBUG ("LegCabin: " << lBKKLegYCabin.toString());
00272
00283     FacBomManager::addToListAndMap (lLHRLegYCabin, lLHRSYDSegmentYCabin,
00284                                     lLHRSYDSegmentYCabin.getFullerKey());
00285     FacBomManager::addToListAndMap (lLHRLegYCabin, lLHRBKKSegmentYCabin,
00286                                     lLHRBKKSegmentYCabin.getFullerKey());
00287     FacBomManager::addToListAndMap (lBKKLegYCabin, lLHRSYDSegmentYCabin,
00288                                     lLHRSYDSegmentYCabin.getFullerKey());
00289     FacBomManager::addToListAndMap (lBKKLegYCabin, lBKKSYSYDSegmentYCabin,
00290                                     lBKKSYSYDSegmentYCabin.getFullerKey());
00291
00302     FacBomManager::addToListAndMap (lLHRSYDSegmentYCabin, lLHRLegYCabin,
00303                                     lLHRLegYCabin.getFullerKey());
00304     FacBomManager::addToListAndMap (lLHRBKKSegmentYCabin, lLHRLegYCabin,
00305                                     lLHRLegYCabin.getFullerKey());
00306     FacBomManager::addToListAndMap (lLHRSYDSegmentYCabin, lBKKLegYCabin,
00307                                     lBKKLegYCabin.getFullerKey());

```

```

00308     FacBomManager::addToListAndMap (lBKKSYSYDSegmentYCabin, lBKKLegYCabin,
00309                                     lBKKLegYCabin.getFullerKey());
00310
00311
00312     // Step 0.7: fare family level
00313     // Create a FareFamily (1) for the Segment LHR-BKK, cabin Y on BA's Inv
00314     const FamilyCode_T l1 ("EcoSaver");
00315     FareFamilyKey l1FareFamilyKey (l1);
00316
00317     FareFamily& lLHRBKKSegmentYCabin1Family =
00318         FacBom<FareFamily>::instance().create (l1FareFamilyKey);
00319     FacBomManager::addToListAndMap (lLHRBKKSegmentYCabin,
00320                                     lLHRBKKSegmentYCabin1Family);
00321     FacBomManager::linkWithParent (lLHRBKKSegmentYCabin,
00322                                     lLHRBKKSegmentYCabin1Family);
00323
00324     // Display the booking class
00325     // STDAIR_LOG_DEBUG ("FareFamily: "
00326     //                    << lLHRBKKSegmentYCabin1Family.toString());
00327
00328     // Create a FareFamily (1) for the Segment BKK-SYD, cabin Y on BA's Inv
00329     FareFamily& lBKKSYSYDSegmentYCabin1Family =
00330         FacBom<FareFamily>::instance().create (l1FareFamilyKey);
00331     FacBomManager::addToListAndMap (lBKKSYSYDSegmentYCabin,
00332                                     lBKKSYSYDSegmentYCabin1Family);
00333     FacBomManager::linkWithParent (lBKKSYSYDSegmentYCabin,
00334                                     lBKKSYSYDSegmentYCabin1Family);
00335
00336     // Display the booking class
00337     // STDAIR_LOG_DEBUG ("FareFamily: "
00338     //                    << lLHRBKKSegmentYCabin1Family.toString());
00339
00340     // Create a FareFamily (1) for the Segment LHR-SYD, cabin Y on BA's Inv
00341     FareFamily& lLHRSYDSegmentYCabin1Family =
00342         FacBom<FareFamily>::instance().create (l1FareFamilyKey);
00343     FacBomManager::addToListAndMap (lLHRSYDSegmentYCabin,
00344                                     lLHRSYDSegmentYCabin1Family);
00345     FacBomManager::linkWithParent (lLHRSYDSegmentYCabin,
00346                                     lLHRSYDSegmentYCabin1Family);
00347
00348     // Display the booking class
00349     // STDAIR_LOG_DEBUG ("FareFamily: "
00350     //                    << lLHRBKKSegmentYCabin1Family.toString());
00351
00352
00353     // Step 0.8: booking class level
00354     // Create a BookingClass (Q) for the Segment LHR-BKK, cabin Y,
00355     // fare family 1 on BA's Inv
00356     const ClassCode_T lQ ("Q");
00357     BookingClassKey lQBookingClassKey (lQ);
00358
00359     BookingClass& lLHRBKKSegmentYCabin1FamilyQClass =
00360         FacBom<BookingClass>::instance().create (lQBookingClassKey);
00361     FacBomManager::addToListAndMap (lLHRBKKSegmentYCabin1Family,
00362                                     lLHRBKKSegmentYCabin1FamilyQClass);
00363     FacBomManager::linkWithParent (lLHRBKKSegmentYCabin1Family,
00364                                     lLHRBKKSegmentYCabin1FamilyQClass);
00365
00366     FacBomManager::addToListAndMap (lLHRBKKSegmentYCabin,
00367                                     lLHRBKKSegmentYCabin1FamilyQClass);
00368     FacBomManager::addToListAndMap (lLHRBKKSegment,
00369                                     lLHRBKKSegmentYCabin1FamilyQClass);

```

```

00370
00371 // Display the booking class
00372 // STDAIR_LOG_DEBUG ("BookingClass: "
00373 //                  << lLHRBKKSegmentYCabinlFamilyQClass.toString());
00374
00375 // Create a BookingClass (Q) for the Segment BKK-SYD, cabin Y,
00376 // fare family 1 on BA's Inv
00377 BookingClass& lBKKSYDSegmentYCabinlFamilyQClass =
00378     FacBom<BookingClass>::instance().create (lQBookingClassKey);
00379 FacBomManager::addToListAndMap (lBKKSYDSegmentYCabinlFamily,
00380     lBKKSYDSegmentYCabinlFamilyQClass);
00381 FacBomManager::linkWithParent (lBKKSYDSegmentYCabinlFamily,
00382     lBKKSYDSegmentYCabinlFamilyQClass);
00383
00384 FacBomManager::addToListAndMap (lBKKSYDSegmentYCabin,
00385     lBKKSYDSegmentYCabinlFamilyQClass);
00386 FacBomManager::addToListAndMap (lBKKSYDSegment,
00387     lBKKSYDSegmentYCabinlFamilyQClass);
00388
00389 // Display the booking class
00390 // STDAIR_LOG_DEBUG ("BookingClass: "
00391 //                  << lLHRBKKSegmentYCabinlFamilyQClass.toString());
00392
00393 // Create a BookingClass (Q) for the Segment LHR-SYD, cabin Y,
00394 // fare family 1 on BA's Inv
00395 BookingClass& lLHRSYDSegmentYCabinlFamilyQClass =
00396     FacBom<BookingClass>::instance().create (lQBookingClassKey);
00397 FacBomManager::addToListAndMap (lLHRSYDSegmentYCabinlFamily,
00398     lLHRSYDSegmentYCabinlFamilyQClass);
00399 FacBomManager::linkWithParent (lLHRSYDSegmentYCabinlFamily,
00400     lLHRSYDSegmentYCabinlFamilyQClass);
00401
00402 FacBomManager::addToListAndMap (lLHRSYDSegmentYCabin,
00403     lLHRSYDSegmentYCabinlFamilyQClass);
00404 FacBomManager::addToListAndMap (lLHRSYDSegment,
00405     lLHRSYDSegmentYCabinlFamilyQClass);
00406
00407 // Display the booking class
00408 // STDAIR_LOG_DEBUG ("BookingClass: "
00409 //                  << lLHRBKKSegmentYCabinlFamilyQClass.toString());
00410
00411
00412 // ///// AF /////
00413 // Step 0.2: Flight-date level
00414 // Create a FlightDate (AF084/20-MAR-2011) for AF's Inventory
00415 lFlightNumber = 84;
00416 lDate = Date_T (2011, 3, 20);
00417 lFlightDateKey = FlightDateKey (lFlightNumber, lDate);
00418
00419 FlightDate& lAF084_20110320_FD =
00420     FacBom<FlightDate>::instance().create (lFlightDateKey);
00421 FacBomManager::addToListAndMap (lAFInv, lAF084_20110320_FD);
00422 FacBomManager::linkWithParent (lAFInv, lAF084_20110320_FD);
00423
00424 // Display the flight-date
00425 // STDAIR_LOG_DEBUG ("FlightDate: " << lAF084_20110320_FD.toString());
00426
00427 // Step 0.3: Segment-date level
00428 // Create a SegmentDate (CDG-SFO) for AF's Inventory
00429 const AirportCode_T lCDG ("CDG");
00430 const AirportCode_T lSFO ("SFO");
00431 const Duration_T l1040 (10, 40, 0);

```

```

00432     const Duration_T l1250 (12, 50, 0);
00433     const Duration_T l1110 (11, 10, 0);
00434     lSegmentDateKey = SegmentDateKey (lCDG, lSFO);
00435
00436     SegmentDate& lCDGSFOSegment =
00437         FacBom<SegmentDate>::instance().create (lSegmentDateKey);
00438     FacBomManager::addToListAndMap (lAF084_20110320_FD, lCDGSFOSegment);
00439     FacBomManager::linkWithParent (lAF084_20110320_FD, lCDGSFOSegment);
00440
00441     // Display the segment-date
00442     // STDAIR_LOG_DEBUG ("SegmentDate: " << lCDGSFOSegment.toString());
00443
00444     // Fill the SegmentDate content
00445     lCDGSFOSegment.setBoardingDate (lDate);
00446     lCDGSFOSegment.setOffDate (lDate);
00447     lCDGSFOSegment.setBoardingTime (l1040);
00448     lCDGSFOSegment.setOffTime (l1250);
00449     lCDGSFOSegment.setElapsedTime (l1110);
00450
00451     // Step 0.4: Leg-date level
00452     // Create a LegDate (CDG) for AF's Inventory
00453     lLegDateKey = LegDateKey (lCDG);
00454
00455     LegDate& lCDGLeg = FacBom<LegDate>::instance().create (lLegDateKey);
00456     FacBomManager::addToListAndMap (lAF084_20110320_FD, lCDGLeg);
00457     FacBomManager::linkWithParent (lAF084_20110320_FD, lCDGLeg);
00458
00459     // Fill the LegDate content
00460     lCDGLeg.setOffPoint (lSFO);
00461     lCDGLeg.setBoardingDate (lDate);
00462     lCDGLeg.setOffDate (lDate);
00463     lCDGLeg.setBoardingTime (l1040);
00464     lCDGLeg.setOffTime (l1250);
00465     lCDGLeg.setElapsedTime (l1110);
00466
00467     // Display the leg-date
00468     // STDAIR_LOG_DEBUG ("LegDate: " << lCDGLeg.toString());
00469
00470     // Link the segment-dates with the leg-dates
00471     FacBomManager::addToListAndMap (lCDGLeg, lCDGSFOSegment);
00472     FacBomManager::addToListAndMap (lCDGSFOSegment, lCDGLeg);
00473
00474     // Step 0.5: segment-cabin level
00475     // Create a SegmentCabin (Y) for the Segment CDG-SFO of AF's Inventory
00476     SegmentCabin& lCDGSFOSegmentYCabin =
00477         FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
00478     FacBomManager::addToListAndMap (lCDGSFOSegment, lCDGSFOSegmentYCabin);
00479     FacBomManager::linkWithParent (lCDGSFOSegment, lCDGSFOSegmentYCabin);
00480
00481     // Display the segment-cabin
00482     // STDAIR_LOG_DEBUG ("SegmentCabin: " << lCDGSFOSegmentYCabin.toString());
00483
00484     // Step 0.6: leg-cabin level
00485     // Create a LegCabin (Y) for the Leg CDG-SFO on AF's Inventory
00486     LegCabin& lCDGLegYCabin =
00487         FacBom<LegCabin>::instance().create (lYLegCabinKey);
00488     FacBomManager::addToListAndMap (lCDGLeg, lCDGLegYCabin);
00489     FacBomManager::linkWithParent (lCDGLeg, lCDGLegYCabin);
00490
00491     // Display the leg-cabin
00492     // STDAIR_LOG_DEBUG ("LegCabin: " << lLHRLegYCabin.toString());

```

```

00494
00495 // Link the segment-dates with the leg-dates
00496 FacBomManager::addToListAndMap (lCDGLegYCabin, lCDGSFOSegmentYCabin,
00497                                 lCDGSFOSegmentYCabin.getFullerKey());
00498 FacBomManager::addToListAndMap (lCDGSFOSegmentYCabin, lCDGLegYCabin,
00499                                 lCDGLegYCabin.getFullerKey());
00500
00501
00502 // Step 0.7: fare family level
00503 // Create a fareFamily (1) for the Segment CDG-SFO, cabin Y on AF's Inv
00504 FareFamily& lCDGSFOSegmentYCabinlFamily =
00505     FacBom<FareFamily>::instance().create (1lFareFamilyKey);
00506 FacBomManager::addToListAndMap (lCDGSFOSegmentYCabin,
00507                                 lCDGSFOSegmentYCabinlFamily);
00508 FacBomManager::linkWithParent (lCDGSFOSegmentYCabin,
00509                                 lCDGSFOSegmentYCabinlFamily);
00510
00511 // Display the fare family
00512 // STDAIR_LOG_DEBUG ("fareFamily: "
00513 //
00514 //                     << lCDGSFOSegmentYCabinlFamily.toString());
00515
00516 // Step 0.8: booking class level Create a BookingClass (Q) for the
00517 // Segment CDG-SFO, cabin Y, fare family 1 on AF's Inv
00518 BookingClass& lCDGSFOSegmentYCabinlFamilyQClass =
00519     FacBom<BookingClass>::instance().create (1QBookingClassKey);
00520 FacBomManager::addToListAndMap (lCDGSFOSegmentYCabinlFamily,
00521                                 lCDGSFOSegmentYCabinlFamilyQClass);
00522 FacBomManager::linkWithParent (lCDGSFOSegmentYCabinlFamily,
00523                                 lCDGSFOSegmentYCabinlFamilyQClass);
00524
00525 FacBomManager::addToListAndMap (lCDGSFOSegmentYCabin,
00526                                 lCDGSFOSegmentYCabinlFamilyQClass);
00527 FacBomManager::addToListAndMap (lCDGSFOSegment,
00528                                 lCDGSFOSegmentYCabinlFamilyQClass);
00529
00530 // Display the booking class
00531 // STDAIR_LOG_DEBUG ("BookingClass: "
00532 //                     << lCDGSFOSegmentYCabinlFamilyQClass.toString());
00533
00534 /*=====
00535 =====
00536 =====
00537 */
00537 // Schedule:
00538 // BA:
00539 // Step 1: flight period level
00540 // Create a flight period for BA9:
00541 const DoWStruct lDoWSrtuct ("1111111");
00542 const Date_T lBA9DateRangeStart (2010, boost::gregorian::Jun, 6);
00543 const Date_T lBA9DateRangeEnd (2010, boost::gregorian::Jun, 7);
00544 const DatePeriod_T lBA9DatePeriod (lBA9DateRangeStart, lBA9DateRangeEnd);
00545 const PeriodStruct lBA9PeriodStruct (lBA9DatePeriod, lDoWSrtuct);
00546
00547 lFlightNumber = FlightNumber_T (9);
00548
00549 FlightPeriodKey lBA9FlightPeriodKey (lFlightNumber, lBA9PeriodStruct);
00550
00551 FlightPeriod& lBA9FlightPeriod =
00552     FacBom<FlightPeriod>::instance().create (lBA9FlightPeriodKey);

```

```

00553     FacBomManager::addToListAndMap (lBAInv, lBA9FlightPeriod);
00554     FacBomManager::linkWithParent (lBAInv, lBA9FlightPeriod);
00555
00556     // Step 2: segment period level
00557     // Create a segment period for SIN-BKK:
00558
00559     SegmentPeriodKey lLHRSYDSegmentPeriodKey (lLHR, lSYD);
00560
00561     SegmentPeriod& lLHRSYDSegmentPeriod =
00562         FacBom<SegmentPeriod>::instance().create (lLHRSYDSegmentPeriodKey);
00563     FacBomManager::addToListAndMap (lBA9FlightPeriod, lLHRSYDSegmentPeriod);
00564     FacBomManager::linkWithParent (lBA9FlightPeriod, lLHRSYDSegmentPeriod);
00565
00566     lLHRSYDSegmentPeriod.setBoardingTime (l2135);
00567     lLHRSYDSegmentPeriod.setOffTime (l1540);
00568     lLHRSYDSegmentPeriod.setElapsedTime (l1105);
00569     ClassList_String_T lYM ("YM");
00570     lLHRSYDSegmentPeriod.addCabinBookingClassList (lY,lYM);
00571
00572     // AF:
00573     // Step 1: flight period level
00574     // Create a flight period for AF84:
00575     const Date_T lAF84DateRangeStart (2011, boost::gregorian::Mar, 20);
00576     const Date_T lAF84DateRangeEnd (2011, boost::gregorian::Mar, 21);
00577     const DatePeriod_T lAF84DatePeriod (lAF84DateRangeStart, lAF84DateRangeEnd);
00578     const PeriodStruct lAF84PeriodStruct (lAF84DatePeriod, lDoWSrtuct);
00579
00580     lFlightNumber = FlightNumber_T (84);
00581
00582     FlightPeriodKey lAF84FlightPeriodKey (lFlightNumber, lAF84PeriodStruct);
00583
00584     FlightPeriod& lAF84FlightPeriod =
00585         FacBom<FlightPeriod>::instance().create (lAF84FlightPeriodKey);
00586     FacBomManager::addToListAndMap (lAFInv, lAF84FlightPeriod);
00587     FacBomManager::linkWithParent (lAFInv, lAF84FlightPeriod);
00588
00589     // Step 2: segment period level
00590     // Create a segment period for SIN-BKK:
00591
00592     SegmentPeriodKey lCDGSFOSegmentPeriodKey (lCDG, lSFO);
00593
00594     SegmentPeriod& lCDGSFOSegmentPeriod =
00595         FacBom<SegmentPeriod>::instance().create (lCDGSFOSegmentPeriodKey);
00596     FacBomManager::addToListAndMap (lAF84FlightPeriod, lCDGSFOSegmentPeriod);
00597     FacBomManager::linkWithParent (lAF84FlightPeriod, lCDGSFOSegmentPeriod);
00598
00599     lCDGSFOSegmentPeriod.setBoardingTime (l1040);
00600     lCDGSFOSegmentPeriod.setOffTime (l1250);
00601     lCDGSFOSegmentPeriod.setElapsedTime (l1110);
00602     lCDGSFOSegmentPeriod.addCabinBookingClassList (lY,lYM);
00603
00604     /*=====
00605     =====
00606     =====
00607     =====*/
00607     // O&D
00608     // Create an O&D Date (BA;9,2010-Jun-06;LHR,SYD) for BA's Inventory
00609     OnDString_T lBALHRSYDOnDStr = "BA;9,2010-Jun-06;LHR,SYD";
00610     OnDStringList_T lBAOnDStrList;
00611     lBAOnDStrList.push_back (lBALHRSYDOnDStr);

```

```

00612
00613     OnDDateKey lBAOnDDateKey (lBAOnDStrList);
00614     OnDDate& lBA_LHRSYD_OnDDate =
00615         FacBom<OnDDate>::instance().create (lBAOnDDateKey);
00616     // Link to the inventory
00617     FacBomManager::addToListAndMap (lBAInv, lBA_LHRSYD_OnDDate);
00618     FacBomManager::linkWithParent (lBAInv, lBA_LHRSYD_OnDDate);
00619
00620     // Add the segment
00621     FacBomManager::addToListAndMap (lBA_LHRSYD_OnDDate, lLHRSYDSegment);
00622
00623     // Add total forecast info for cabin Y.
00624     const MeanStdDevPair_T lMean60StdDev6 (60.0, 6.0);
00625     const WTP_T lWTP750 = 750.0;
00626     const WTPDemandPair_T lWTP750Mean60StdDev6 (lWTP750, lMean60StdDev6);
00627     lBA_LHRSYD_OnDDate.setTotalForecast (lY, lWTP750Mean60StdDev6);
00628
00629     // Create an O&D Date (AF;84,2011-Mar-21;CDG,SFO) for AF's Inventory
00630     OnDString_T lAFLHRSYDOnDStr = "AF;9,2011-Mar-20;CDG,SFO";
00631     OnDStringList_T lAFOnDStrList;
00632     lAFOnDStrList.push_back (lAFLHRSYDOnDStr);
00633
00634     OnDDateKey lAFOnDDateKey (lAFOnDStrList);
00635     OnDDate& lAF_LHRSYD_OnDDate =
00636         FacBom<OnDDate>::instance().create (lAFOnDDateKey);
00637     // Link to the inventory
00638     FacBomManager::addToListAndMap (lAFInv, lAF_LHRSYD_OnDDate);
00639     FacBomManager::linkWithParent (lAFInv, lAF_LHRSYD_OnDDate);
00640
00641     // Add the segment
00642     FacBomManager::addToListAndMap (lAF_LHRSYD_OnDDate, lLHRSYDSegment);
00643
00644     // Add total forecast info for cabin Y.
00645     lAF_LHRSYD_OnDDate.setTotalForecast (lY, lWTP750Mean60StdDev6);
00646
00647 }
00648 // //////////////////////////////////////
00649 void CmdBomManager::buildCompleteDummyInventory (BomRoot& ioBomRoot) {
00650
00651     // Build a dummy inventory, containing a dummy flight-date with a
00652     // single segment-cabin and a single leg-cabin.
00653     const CabinCapacity_T lCapacity = DEFAULT_CABIN_CAPACITY;
00654     buildDummyInventory (ioBomRoot, lCapacity);
00655
00656     // Retrieve the (sample) segment-cabin.
00657     SegmentCabin& lDummySegmentCabin =
00658         BomRetriever::retrieveDummySegmentCabin (ioBomRoot);
00659
00660     // Retrieve the (sample) leg-cabin.
00661     LegCabin& lDummyLegCabin =
00662         BomRetriever::retrieveDummyLegCabin (ioBomRoot);
00663
00664     // Add some booking classes to the dummy segment-cabin and some
00665     // virtual ones to the dummy leg-cabin.
00666     // First booking class yield and demand information.
00667     Yield_T lYield = 100;
00668     MeanValue_T lMean = 20;
00669     StdDevValue_T lStdDev = 9;
00670     BookingClassKey lBCKKey (DEFAULT_CLASS_CODE);
00671
00672     BookingClass& lDummyBookingClass =
00673         FacBom<BookingClass>::instance().create (lBCKKey);

```

```

00674     lDummyBookingClass.setYield (lYield);
00675     lDummyBookingClass.setMean (lMean);
00676     lDummyBookingClass.setStdDev (lStdDev);
00677     // Add a booking class to the segment-cabin.
00678     FacBomManager::addToList (lDummySegmentCabin, lDummyBookingClass);
00679
00680     VirtualClassStruct lDummyVirtualClass (lDummyBookingClass);
00681     lDummyVirtualClass.setYield (lYield);
00682     lDummyVirtualClass.setMean (lMean);
00683     lDummyVirtualClass.setStdDev (lStdDev);
00684     // Add the corresponding virtual class to the leg-cabin.
00685     lDummyLegCabin.addVirtualClass (lDummyVirtualClass);
00686
00687     // Second booking class yield and demand information.
00688     lYield = 70;
00689     lMean = 45;
00690     lStdDev= 12;
00691     lDummyBookingClass.setYield (lYield);
00692     lDummyBookingClass.setMean (lMean);
00693     lDummyBookingClass.setStdDev (lStdDev);
00694     // Add a booking class to the segment-cabin.
00695     FacBomManager::addToList (lDummySegmentCabin, lDummyBookingClass);
00696
00697     lDummyVirtualClass.setYield (lYield);
00698     lDummyVirtualClass.setMean (lMean);
00699     lDummyVirtualClass.setStdDev (lStdDev);
00700     // Add the corresponding virtual class to the leg-cabin.
00701     lDummyLegCabin.addVirtualClass (lDummyVirtualClass);
00702
00703     // Third booking class yield and demand information.
00704     lYield = 42;
00705     lMean = 80;
00706     lStdDev= 16;
00707     lDummyBookingClass.setYield (lYield);
00708     lDummyBookingClass.setMean (lMean);
00709     lDummyBookingClass.setStdDev (lStdDev);
00710     // Add a booking class to the segment-cabin.
00711     FacBomManager::addToList (lDummySegmentCabin, lDummyBookingClass);
00712
00713     lDummyVirtualClass.setYield (lYield);
00714     lDummyVirtualClass.setMean (lMean);
00715     lDummyVirtualClass.setStdDev (lStdDev);
00716     // Add the corresponding virtual class to the leg-cabin.
00717     lDummyLegCabin.addVirtualClass (lDummyVirtualClass);
00718
00719 }
00720
00721 // //////////////////////////////////////
00722 void CmdBomManager::buildDummyInventory (BomRoot& ioBomRoot,
00723     const CabinCapacity_T& iCapacity) {
00724     // Inventory
00725     const InventoryKey lInventoryKey (DEFAULT_AIRLINE_CODE);
00726     Inventory& lInv = FacBom<Inventory>::instance().create (lInventoryKey);
00727     FacBomManager::addToListAndMap (ioBomRoot, lInv);
00728     FacBomManager::linkWithParent (ioBomRoot, lInv);
00729
00730     // Flight-date
00731     FlightDateKey lFlightDateKey (DEFAULT_FLIGHT_NUMBER, DEFAULT_DEPARTURE_DATE);
00732     FlightDate& lFlightDate =
00733         FacBom<FlightDate>::instance().create (lFlightDateKey);
00734     FacBomManager::addToListAndMap (lInv, lFlightDate);
00735     FacBomManager::linkWithParent (lInv, lFlightDate);

```

```

00736
00737 // Leg-date
00738 LegDateKey lLegDateKey (DEFAULT_ORIGIN);
00739 LegDate& lLeg = FacBom<LegDate>::instance().create (lLegDateKey);
00740 FacBomManager::addToListAndMap (lFlightDate, lLeg);
00741 FacBomManager::linkWithParent (lFlightDate, lLeg);
00742
00743 // Fill the LegDate content
00744 lLeg.setOffPoint (DEFAULT_DESTINATION);
00745 lLeg.setBoardingDate (DEFAULT_DEPARTURE_DATE);
00746 lLeg.setOffDate (DEFAULT_DEPARTURE_DATE);
00747 lLeg.setBoardingTime (Duration_T (14, 0, 0));
00748 lLeg.setOffTime (Duration_T (16, 0, 0));
00749 lLeg.setElapsedTime (Duration_T (8, 0, 0));
00750
00751 // Leg-cabin
00752 LegCabinKey lLegCabinKey (DEFAULT_CABIN_CODE);
00753 LegCabin& lLegCabin = FacBom<LegCabin>::instance().create (lLegCabinKey);
00754 FacBomManager::addToListAndMap (lLeg, lLegCabin);
00755 FacBomManager::linkWithParent (lLeg, lLegCabin);
00756
00757 lLegCabin.setCapacities (iCapacity);
00758 lLegCabin.setAvailabilityPool (iCapacity);
00759
00760 // Segment-date
00761 SegmentDateKey lSegmentDateKey (DEFAULT_ORIGIN, DEFAULT_DESTINATION);
00762 SegmentDate& lSegment =
00763     FacBom<SegmentDate>::instance().create (lSegmentDateKey);
00764 FacBomManager::addToListAndMap (lFlightDate, lSegment);
00765 FacBomManager::linkWithParent (lFlightDate, lSegment);
00766
00767 // Links between the segment-date and the leg-date
00768 FacBomManager::addToListAndMap (lLeg, lSegment);
00769 FacBomManager::addToListAndMap (lSegment, lLeg);
00770
00771 // Fill the SegmentDate content
00772 lSegment.setBoardingDate (DEFAULT_DEPARTURE_DATE);
00773 lSegment.setOffDate (DEFAULT_DEPARTURE_DATE);
00774 lSegment.setBoardingTime (Duration_T (14, 0, 0));
00775 lSegment.setOffTime (Duration_T (16, 0, 0));
00776 lSegment.setElapsedTime (Duration_T (8, 0, 0));
00777
00778 // Segment-cabin
00779 SegmentCabinKey lSegmentCabinKey (DEFAULT_CABIN_CODE);
00780 SegmentCabin& lSegmentCabin =
00781     FacBom<SegmentCabin>::instance().create (lSegmentCabinKey);
00782 FacBomManager::addToListAndMap (lSegment, lSegmentCabin);
00783 FacBomManager::linkWithParent (lSegment, lSegmentCabin);
00784
00785 // Links between the segment-cabin and the leg-cabin
00786 FacBomManager::addToListAndMap (lLegCabin, lSegmentCabin,
00787     lSegmentCabin.getFullerKey());
00788 FacBomManager::addToListAndMap (lSegmentCabin, lLegCabin,
00789     lLegCabin.getFullerKey());
00790
00791 // Create a FareFamily (1) for the Segment LHR-BKK, cabin Y on BA's Inv
00792 const FamilyCode_T l1 ("EcoSaver");
00793 FareFamilyKey l1FareFamilyKey (l1);
00794
00795 FareFamily& lSegmentYCabin1Family =
00796     FacBom<FareFamily>::instance().create (l1FareFamilyKey);
00797 FacBomManager::addToListAndMap (lSegmentCabin, lSegmentYCabin1Family);

```

```

00798     FacBomManager::linkWithParent (lSegmentCabin, lSegmentYCabinlFamily);
00799
00800     // Create a booking-class
00801     const ClassCode_T lQ ("Q");
00802     BookingClassKey lQBookingClassKey (lQ);
00803
00804     BookingClass& lSegmentYCabinlFamilyQClass =
00805         FacBom<BookingClass>::instance().create (lQBookingClassKey);
00806     FacBomManager::addToListAndMap (lSegmentYCabinlFamily,
00807                                     lSegmentYCabinlFamilyQClass);
00808     FacBomManager::linkWithParent (lSegmentYCabinlFamily,
00809                                     lSegmentYCabinlFamilyQClass);
00810
00811     FacBomManager::addToListAndMap (lSegmentCabin, lSegmentYCabinlFamilyQClass);
00812     FacBomManager::addToListAndMap (lSegment, lSegmentYCabinlFamilyQClass);
00813
00814 }
00815
00816 // //////////////////////////////////////
00817 void CmdBomManager::buildSamplePricing (BomRoot& ioBomRoot) {
00818
00819     // Set the airport-pair primary key.
00820     const AirportPairKey lAirportPairKey (AIRPORT_LHR, AIRPORT_SYD);
00821
00822     // Create the AirportPairKey object and link it to the BOM tree root.
00823     AirportPair& lAirportPair =
00824         FacBom<AirportPair>::instance().create (lAirportPairKey);
00825     FacBomManager::addToListAndMap (ioBomRoot, lAirportPair);
00826     FacBomManager::linkWithParent (ioBomRoot, lAirportPair);
00827
00828     // Set the fare date-period primary key.
00829     const Date_T lDateRangeStart (2011, boost::gregorian::Jan, 15);
00830     const Date_T lDateRangeEnd (2011, boost::gregorian::Dec, 31);
00831     const DatePeriod_T lDateRange (lDateRangeStart, lDateRangeEnd);
00832     const DatePeriodKey lDatePeriodKey (lDateRange);
00833
00834     // Create the DatePeriodKey object and link it to the PosChannel object.
00835     DatePeriod& lDatePeriod =
00836         FacBom<DatePeriod>::instance().create (lDatePeriodKey);
00837     FacBomManager::addToListAndMap (lAirportPair, lDatePeriod);
00838     FacBomManager::linkWithParent (lAirportPair, lDatePeriod);
00839
00840     // Set the point-of-sale-channel primary key.
00841     const PosChannelKey lPosChannelKey (POS_LHR, CHANNEL_DN);
00842
00843     // Create the PositionKey object and link it to the AirportPair object.
00844     PosChannel& lPosChannel =
00845         FacBom<PosChannel>::instance().create (lPosChannelKey);
00846     FacBomManager::addToListAndMap (lDatePeriod, lPosChannel);
00847     FacBomManager::linkWithParent (lDatePeriod, lPosChannel);
00848
00849     // Set the fare time-period primary key.
00850     const Time_T lTimeRangeStart (0, 0, 0);
00851     const Time_T lTimeRangeEnd (23, 0, 0);
00852     const TimePeriodKey lTimePeriodKey (lTimeRangeStart, lTimeRangeEnd);
00853
00854     // Create the TimePeriodKey and link it to the DatePeriod object.
00855     TimePeriod& lTimePeriod =
00856         FacBom<TimePeriod>::instance().create (lTimePeriodKey);
00857     FacBomManager::addToListAndMap (lPosChannel, lTimePeriod);
00858     FacBomManager::linkWithParent (lPosChannel, lTimePeriod);
00859

```

```

00860 // Pricing -- Generate the FareRule
00861 const FareFeaturesKey lFareFeaturesKey (TRIP_TYPE_ROUND_TRIP,
00862                                         NO_ADVANCE_PURCHASE,
00863                                         SATURDAY_STAY,
00864                                         CHANGE_FEES,
00865                                         NON_REFUNDABLE,
00866                                         NO_STAY_DURATION);
00867
00868 // Create the FareFeaturesKey and link it to the TimePeriod object.
00869 FareFeatures& lFareFeatures =
00870     FacBom<FareFeatures>::instance().create (lFareFeaturesKey);
00871 FacBomManager::addToListAndMap (lTimePeriod, lFareFeatures);
00872 FacBomManager::linkWithParent (lTimePeriod, lFareFeatures);
00873
00874 // Revenue Accounting -- Generate the YieldRule
00875 const YieldFeaturesKey lYieldFeaturesKey (TRIP_TYPE_ROUND_TRIP,
00876                                           CABIN_Y);
00877
00878 // Create the YieldFeaturesKey and link it to the TimePeriod object.
00879 YieldFeatures& lYieldFeatures =
00880     FacBom<YieldFeatures>::instance().create (lYieldFeaturesKey);
00881 FacBomManager::addToListAndMap (lTimePeriod, lYieldFeatures);
00882 FacBomManager::linkWithParent (lTimePeriod, lYieldFeatures);
00883
00884 // Generate Segment Features and link them to their respective
00885 // fare and yield rules.
00886 AirlineCodeList_T lAirlineCodeList;
00887 lAirlineCodeList.push_back (AIRLINE_CODE_BA);
00888 ClassList_StringList_T lClassCodeList;
00889 lClassCodeList.push_back (CLASS_CODE_Y);
00890 const AirlineClassListKey lAirlineClassListKey (lAirlineCodeList,
00891                                                  lClassCodeList);
00892
00893 // Create the AirlineClassList
00894 AirlineClassList& lAirlineClassList =
00895     FacBom<AirlineClassList>::instance().create (lAirlineClassListKey);
00896 // Link the AirlineClassList to the FareFeatures object
00897 lAirlineClassList.setFare (900);
00898 FacBomManager::addToListAndMap (lFareFeatures, lAirlineClassList);
00899 FacBomManager::linkWithParent (lFareFeatures, lAirlineClassList);
00900
00901 // Link the AirlineClassList to the YieldFeatures object
00902 lAirlineClassList.setYield (900);
00903 FacBomManager::addToListAndMap (lYieldFeatures, lAirlineClassList);
00904 // \todo (gsabatier): the following calls overrides the parent for
00905 //     lAirlineClassList. Check that it is what is actually wanted.
00906 FacBomManager::linkWithParent (lYieldFeatures, lAirlineClassList);
00907 }
00908
00909 // //////////////////////////////////////
00910 void CmdBomManager::
00911 buildSampleTravelSolutionForPricing (TravelSolutionList_T& ioTravelSolutionList
00912 ) {
00913     // Clean the list
00914     ioTravelSolutionList.clear();
00915
00916     //
00917     const std::string lBA9_SegmentDateKey ("BA, 9, 2011-06-10, LHR, SYD, 21:45");
00918
00919     // Add the segment date key to the travel solution

```

```

00920     TravelSolutionStruct lTS;
00921     lTS.addSegment (lBA9_SegmentDateKey);
00922
00923     // Add the travel solution to the list
00924     ioTravelSolutionList.push_back (lTS);
00925 }
00926
00927 // //////////////////////////////////////
00928 void CmdBomManager::
00929 buildSampleTravelSolutions (TravelSolutionList_T& ioTravelSolutionList) {
00930
00931     // Clean the list
00932     ioTravelSolutionList.clear();
00933
00934     //
00935     const std::string lBA9_SegmentDateKey ("BA, 9, 2011-06-10, LHR, SYD, 21:45");
00936
00937     // Add the segment date key to the travel solution
00938     TravelSolutionStruct lTS;
00939     lTS.addSegment (lBA9_SegmentDateKey);
00940
00941     // Fare option
00942     const ClassCode_T lClassPath (CLASS_CODE_Q);
00943     const Fare_T lFare (900);
00944     const ChangeFees_T lChangeFee (CHANGE_FEES);
00945     const NonRefundable_T lIsNonRefundable (NON_REFUNDABLE);
00946     const SaturdayStay_T lSaturdayStay (SATURDAY_STAY);
00947     const FareOptionStruct lFareOption (lClassPath, lFare, lChangeFee,
00948                                         isNonRefundable, lSaturdayStay);
00949
00950     // Add (a copy of) the fare option
00951     lTS.addFareOption (lFareOption);
00952
00953     // Map of class availabilities: set the availability for the Q
00954     // booking class (the one corresponding to the fare option) to 8.
00955     ClassAvailabilityMap_T lClassAvailabilityMap;
00956     const Availability_T lAvl (8);
00957     const bool hasInsertBeenSuccessful = lClassAvailabilityMap.
00958         insert (ClassAvailabilityMap_T::value_type (lClassPath, lAvl)).second;
00959     assert (hasInsertBeenSuccessful == true);
00960     // Add the map to the dedicated list held by the travel solution
00961     lTS.addClassAvailabilityMap (lClassAvailabilityMap);
00962
00963     // Add the travel solution to the list
00964     ioTravelSolutionList.push_back (lTS);
00965 }
00966
00967 // //////////////////////////////////////
00968 BookingRequestStruct CmdBomManager::buildSampleBookingRequest() {
00969     // Origin
00970     const AirportCode_T lOrigin (AIRPORT_LHR);
00971
00972     // Destination
00973     const AirportCode_T lDestination (AIRPORT_SYD);
00974
00975     // Point of Sale (POS)
00976     const CityCode_T lPOS (POS_LHR);
00977
00978     // Preferred departure date (10-JUN-2011)
00979     const Date_T lPreferredDepartureDate (2011, boost::gregorian::Jun, 10);
00980

```

```

00981 // Preferred departure time (08:00)
00982 const Duration_T lPreferredDepartureTime (8, 0, 0);
00983
00984 // Date of the request (15-MAY-2011)
00985 const Date_T lRequestDate (2011, boost::gregorian::May, 15);
00986
00987 // Time of the request (10:00)
00988 const Duration_T lRequestTime (10, 0, 0);
00989
00990 // Date-time of the request (made of the date and time above)
00991 const DateTime_T lRequestDateTime (lRequestDate, lRequestTime);
00992
00993 // Preferred cabin (also named class of service sometimes)
00994 const CabinCode_T lPreferredCabin (CABIN_ECO);
00995
00996 // Number of persons in the party
00997 const PartySize_T lPartySize (3);
00998
00999 // Channel (direct/indirect, on-line/off-line)
01000 const ChannelLabel_T lChannel (CHANNEL_DN);
01001
01002 // Type of the trip (one-way, inbound/outbound of a return trip)
01003 const TripType_T lTripType (TRIP_TYPE_INBOUND);
01004
01005 // Duration of the stay (expressed as a number of days)
01006 const DayDuration_T lStayDuration (DEFAULT_STAY_DURATION);
01007
01008 // Frequent flyer tier (member, silver, gold, platinum, senator, etc)
01009 const FrequentFlyer_T lFrequentFlyerType (FREQUENT_FLYER_MEMBER);
01010
01011 // Maximum willing-to-pay (WTP, expressed in monetary unit, e.g., EUR)
01012 const WTP_T lWTP (DEFAULT_WTP);
01013
01014 // Value of time, for the customer (expressed in monetary unit per
01015 // unit of time, e.g., EUR/hour)
01016 const PriceValue_T lValueOfTime (DEFAULT_VALUE_OF_TIME);
01017
01018 // Creation of the booking request structure
01019 BookingRequestStruct oBookingRequest (lOrigin, lDestination, lPOS,
01020                                     lPreferredDepartureDate,
01021                                     lRequestDateTime,
01022                                     lPreferredCabin,
01023                                     lPartySize, lChannel,
01024                                     lTripType, lStayDuration,
01025                                     lFrequentFlyerType,
01026                                     lPreferredDepartureTime,
01027                                     lWTP, lValueOfTime);
01028
01029 return oBookingRequest;
01030 }
01031
01032 ///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
01033 BookingRequestStruct CmdBomManager::buildSampleBookingRequestForCRS() {
01034 // Origin
01035 const AirportCode_T lOrigin (AIRPORT_SIN);
01036
01037 // Destination
01038 const AirportCode_T lDestination (AIRPORT_BKK);
01039
01040 // Point of Sale (POS)
01041 const CityCode_T lPOS (POS_SIN);
01042

```

```

01043 // Preferred departure date (30-JAN-2010)
01044 const Date_T lPreferredDepartureDate (2010, boost::gregorian::Jan, 30);
01045
01046 // Preferred departure time (10:00)
01047 const Duration_T lPreferredDepartureTime (10, 0, 0);
01048
01049 // Date of the request (22-JAN-2010)
01050 const Date_T lRequestDate (2010, boost::gregorian::Jan, 22);
01051
01052 // Time of the request (10:00)
01053 const Duration_T lRequestTime (10, 0, 0);
01054
01055 // Date-time of the request (made of the date and time above)
01056 const DateTime_T lRequestDateTime (lRequestDate, lRequestTime);
01057
01058 // Preferred cabin (also named class of service sometimes)
01059 const CabinCode_T lPreferredCabin (CABIN_ECO);
01060
01061 // Number of persons in the party
01062 const PartySize_T lPartySize (3);
01063
01064 // Channel (direct/indirect, on-line/off-line)
01065 const ChannelLabel_T lChannel (CHANNEL_IN);
01066
01067 // Type of the trip (one-way, inbound/outbound of a return trip)
01068 const TripType_T lTripType (TRIP_TYPE_INBOUND);
01069
01070 // Duration of the stay (expressed as a number of days)
01071 const DayDuration_T lStayDuration (DEFAULT_STAY_DURATION);
01072
01073 // Frequent flyer tier (member, silver, gold, platinum, senator, etc)
01074 const FrequentFlyer_T lFrequentFlyerType (FREQUENT_FLYER_MEMBER);
01075
01076 // Maximum willing-to-pay (WTP, expressed in monetary unit, e.g., EUR)
01077 const WTP_T lWTP (DEFAULT_WTP);
01078
01079 // Value of time, for the customer (expressed in monetary unit per
01080 // unit of time, e.g., EUR/hour)
01081 const PriceValue_T lValueOfTime (DEFAULT_VALUE_OF_TIME);
01082
01083 // Creation of the booking request structure
01084 BookingRequestStruct oBookingRequest (lOrigin,
01085                                       lDestination,
01086                                       lPOS,
01087                                       lPreferredDepartureDate,
01088                                       lRequestDateTime,
01089                                       lPreferredCabin,
01090                                       lPartySize, lChannel,
01091                                       lTripType, lStayDuration,
01092                                       lFrequentFlyerType,
01093                                       lPreferredDepartureTime,
01094                                       lWTP, lValueOfTime);
01095
01096 return oBookingRequest;
01097 }
01098
01099 // //////////////////////////////////////
01100 void CmdBomManager::buildPartnershipsSampleInventoryAndRM (BomRoot& ioBomRoot)
01101 {
01102 // Step 0.1: Inventory level
01103 // Create an Inventory for SQ

```

```

01104     const InventoryKey lSQKey ("SQ");
01105     Inventory& lSQInv = FacBom<Inventory>::instance().create (lSQKey);
01106     FacBomManager::addToListAndMap (ioBomRoot, lSQInv);
01107     FacBomManager::linkWithParent (ioBomRoot, lSQInv);
01108
01109     // Create an Inventory for CX
01110     const InventoryKey lCXKey ("CX");
01111     Inventory& lCXInv = FacBom<Inventory>::instance().create (lCXKey);
01112     FacBomManager::addToListAndMap (ioBomRoot, lCXInv);
01113     FacBomManager::linkWithParent (ioBomRoot, lCXInv);
01114
01115
01116     // ===== SQ =====
01117     // Step 0.2: Flight-date level
01118     // Create a FlightDate (SQ11/08-FEB-2010) for SQ's Inventory
01119     FlightNumber_T lFlightNumber = 11;
01120     Date_T lDate (2010, 2, 8);
01121     FlightDateKey lFlightDateKey (lFlightNumber, lDate);
01122
01123     FlightDate& lSQ11_20100208_FD =
01124         FacBom<FlightDate>::instance().create (lFlightDateKey);
01125     FacBomManager::addToListAndMap (lSQInv, lSQ11_20100208_FD);
01126     FacBomManager::linkWithParent (lSQInv, lSQ11_20100208_FD);
01127
01128     // Create a (mkt) FlightDate (SQ1200/08-FEB-2010) for SQ's Inventory
01129     FlightNumber_T lMktFlightNumber = 1200;
01130     //lDate = Date_T (2010, 2, 8);
01131     FlightDateKey lMktFlightDateKey (lMktFlightNumber, lDate);
01132
01133     FlightDate& lSQ1200_20100208_FD =
01134         FacBom<FlightDate>::instance().create (lMktFlightDateKey);
01135     FacBomManager::addToListAndMap (lSQInv, lSQ1200_20100208_FD);
01136     FacBomManager::linkWithParent (lSQInv, lSQ1200_20100208_FD);
01137
01138     // Display the flight-date
01139     // STDAIR_LOG_DEBUG ("FlightDate: " << lBA9_20110610_FD.toString());
01140
01141     // Step 0.3: Segment-date level
01142     // Create a first SegmentDate (SIN-BKK) for SQ's Inventory
01143     const AirportCode_T lSIN ("SIN");
01144     const AirportCode_T lBKK ("BKK");
01145     const DateOffset_T l1Day (1);
01146     const DateOffset_T l2Days (2);
01147     const Duration_T l0820 (8, 20, 0);
01148     const Duration_T l1100 (11, 0, 0);
01149     const Duration_T l10340 (3, 40, 0);
01150     SegmentDateKey lSegmentDateKey (lSIN, lBKK);
01151
01152     SegmentDate& lSINBKKSegment =
01153         FacBom<SegmentDate>::instance().create (lSegmentDateKey);
01154     FacBomManager::addToListAndMap (lSQ11_20100208_FD, lSINBKKSegment);
01155     FacBomManager::linkWithParent (lSQ11_20100208_FD, lSINBKKSegment);
01156
01157     // Fill the SegmentDate content
01158     lSINBKKSegment.setBoardingDate (lDate);
01159     lSINBKKSegment.setOffDate (lDate);
01160     lSINBKKSegment.setBoardingTime (l0820);
01161     lSINBKKSegment.setOffTime (l1100);
01162     lSINBKKSegment.setElapsedTime (l10340);
01163
01164     // Create a second (mkt) SegmentDate (BKK-HKG) for SQ's Inventory
01165     const AirportCode_T lHKG ("HKG");

```

```

01166     const Duration_T l1200 (12, 0, 0);
01167     const Duration_T l1540 (15, 40, 0);
01168     const Duration_T l0240 (2, 40, 0);
01169     SegmentDateKey lMktSegmentDateKey (lBKK, lHKG);
01170
01171     SegmentDate& lMktBKKHKGSegment =
01172         FacBom<SegmentDate>::instance().create (lMktSegmentDateKey);
01173     FacBomManager::addToListAndMap (lSQ1200_20100208_FD, lMktBKKHKGSegment);
01174     FacBomManager::linkWithParent (lSQ1200_20100208_FD, lMktBKKHKGSegment);
01175
01176     // Fill the (mkt) SegmentDate content
01177     lMktBKKHKGSegment.setBoardingDate (lDate);
01178     lMktBKKHKGSegment.setOffDate (lDate);
01179     lMktBKKHKGSegment.setBoardingTime (l1200);
01180     lMktBKKHKGSegment.setOffTime (l1540);
01181     lMktBKKHKGSegment.setElapsedTime (l0240);
01182
01183     // Step 0.4: Leg-date level
01184     // Create a first LegDate (SIN) for SQ's Inventory
01185     LegDateKey lLegDateKey (lSIN);
01186
01187     LegDate& lSINLeg = FacBom<LegDate>::instance().create (lLegDateKey);
01188     FacBomManager::addToListAndMap (lSQ11_20100208_FD, lSINLeg);
01189     FacBomManager::linkWithParent (lSQ11_20100208_FD, lSINLeg);
01190
01191     // Fill the LegDate content
01192     lSINLeg.setOffPoint (lBKK);
01193     lSINLeg.setBoardingDate (lDate);
01194     lSINLeg.setOffDate (lDate);
01195     lSINLeg.setBoardingTime (l0820);
01196     lSINLeg.setOffTime (l1100);
01197     lSINLeg.setElapsedTime (l0340);
01198
01199
01200     // Link the segment-dates with the leg-dates
01201     FacBomManager::addToListAndMap (lSINLeg, lSINBKKSegment);
01202     FacBomManager::addToListAndMap (lSINBKKSegment, lSINLeg);
01203
01204     // Step 0.5: segment-cabin level
01205     // Create a SegmentCabin (Y) for the Segment SIN-BKK of SQ's Inventory
01206     const CabinCode_T lY ("Y");
01207     SegmentCabinKey lYSegmentCabinKey (lY);
01208
01209     SegmentCabin& lSINBKKSegmentYCabin =
01210         FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
01211     FacBomManager::addToListAndMap (lSINBKKSegment, lSINBKKSegmentYCabin);
01212     FacBomManager::linkWithParent (lSINBKKSegment, lSINBKKSegmentYCabin);
01213
01214     // Create a SegmentCabin (Y) for the (mkt) Segment BKK-HKG of SQ's Inventory
01215     SegmentCabin& lMktBKKHKGSegmentYCabin =
01216         FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
01217     FacBomManager::addToListAndMap (lMktBKKHKGSegment, lMktBKKHKGSegmentYCabin);
01218     FacBomManager::linkWithParent (lMktBKKHKGSegment, lMktBKKHKGSegmentYCabin);
01219
01220
01221     // Step 0.6: leg-cabin level
01222     // Create a LegCabin (Y) for the Leg SIN-BKK on SQ's Inventory
01223     LegCabinKey lYLegCabinKey (lY);
01224
01225     LegCabin& lSINLegYCabin =
01226         FacBom<LegCabin>::instance().create (lYLegCabinKey);
01227     FacBomManager::addToListAndMap (lSINLeg, lSINLegYCabin);

```

```

01228     FacBomManager::linkWithParent (lSINLeg, lSINLegYCabin);
01229
01230     CabinCapacity_T lCapacity (100);
01231     lSINLegYCabin.setCapacities (lCapacity);
01232     lSINLegYCabin.setAvailabilityPool (lCapacity);
01233
01245     FacBomManager::addToListAndMap (lSINLegYCabin, lSINBKKSegmentYCabin,
01246                                     lSINBKKSegmentYCabin.getFullerKey());
01247
01258     FacBomManager::addToListAndMap (lSINBKKSegmentYCabin, lSINLegYCabin,
01259                                     lSINLegYCabin.getFullerKey());
01260
01261
01262     // Step 0.7: fare family level
01263     // Create a FareFamily (1) for the Segment SIN-BKK, cabin Y on SQ's Inv
01264     const FamilyCode_T l1 ("EcoSaver");
01265     FareFamilyKey l1FareFamilyKey (l1);
01266
01267     FareFamily& lSINBKKSegmentYCabinlFamily =
01268         FacBom<FareFamily>::instance().create (l1FareFamilyKey);
01269     FacBomManager::addToListAndMap (lSINBKKSegmentYCabin,
01270                                     lSINBKKSegmentYCabinlFamily);
01271     FacBomManager::linkWithParent (lSINBKKSegmentYCabin,
01272                                     lSINBKKSegmentYCabinlFamily);
01273
01274     // Create a FareFamily (1) for the (mkt) Segment BKK-HKG, cabin Y on SQ's Inv
01275
01276     FareFamily& lMktBKKHKGSegmentYCabinlFamily =
01277         FacBom<FareFamily>::instance().create (l1FareFamilyKey);
01278     FacBomManager::addToListAndMap (lMktBKKHKGSegmentYCabin,
01279                                     lMktBKKHKGSegmentYCabinlFamily);
01280     FacBomManager::linkWithParent (lMktBKKHKGSegmentYCabin,
01281                                     lMktBKKHKGSegmentYCabinlFamily);
01282
01283     // Step 0.8: booking class level
01284     // Create a BookingClass (Y) for the Segment SIN-BKK, cabin Y,
01285     // fare family 1 on SQ's Inv
01286     BookingClassKey lYBookingClassKey (lY);
01287
01288     BookingClass& lSINBKKSegmentYCabinlFamilyYClass =
01289         FacBom<BookingClass>::instance().create (lYBookingClassKey);
01290     FacBomManager::addToListAndMap (lSINBKKSegmentYCabinlFamily,
01291                                     lSINBKKSegmentYCabinlFamilyYClass);
01292     FacBomManager::linkWithParent (lSINBKKSegmentYCabinlFamily,
01293                                     lSINBKKSegmentYCabinlFamilyYClass);
01294
01295     FacBomManager::addToListAndMap (lSINBKKSegmentYCabin,
01296                                     lSINBKKSegmentYCabinlFamilyYClass);
01297     FacBomManager::addToListAndMap (lSINBKKSegment,
01298                                     lSINBKKSegmentYCabinlFamilyYClass);
01299
01300     lSINBKKSegmentYCabinlFamilyYClass.setYield(700);
01301
01302     // Create a BookingClass (Y) for the (mkt) Segment BKK-HKG, cabin Y,
01303     // fare family 1 on SQ's Inv
01304     BookingClass& lMktBKKHKGSegmentYCabinlFamilyYClass =
01305         FacBom<BookingClass>::instance().create (lYBookingClassKey);
01306     FacBomManager::addToListAndMap (lMktBKKHKGSegmentYCabinlFamily,
01307                                     lMktBKKHKGSegmentYCabinlFamilyYClass);
01308     FacBomManager::linkWithParent (lMktBKKHKGSegmentYCabinlFamily,
01309                                     lMktBKKHKGSegmentYCabinlFamilyYClass);
01310

```

```

01310     FacBomManager::addToListAndMap (lMktBKKHKGSegmentYCabin,
01311                                     lMktBKKHKGSegmentYCabinlFamilyYClass);
01312     FacBomManager::addToListAndMap (lMktBKKHKGSegment,
01313                                     lMktBKKHKGSegmentYCabinlFamilyYClass);
01314
01315     lMktBKKHKGSegmentYCabinlFamilyYClass.setYield(700);
01316
01317
01318     // Create a BookingClass (M) for the Segment SIN-BKK, cabin Y,
01319     // fare family 1 on SQ's Inv
01320     const ClassCode_T lM ("M");
01321     BookingClassKey lMBookingClassKey (lM);
01322
01323     BookingClass& lSINBKKSegmentYCabinlFamilyMClass =
01324         FacBom<BookingClass>::instance().create (lMBookingClassKey);
01325     FacBomManager::addToListAndMap (lSINBKKSegmentYCabinlFamily,
01326                                     lSINBKKSegmentYCabinlFamilyMClass);
01327     FacBomManager::linkWithParent (lSINBKKSegmentYCabinlFamily,
01328                                     lSINBKKSegmentYCabinlFamilyMClass);
01329
01330     FacBomManager::addToListAndMap (lSINBKKSegmentYCabin,
01331                                     lSINBKKSegmentYCabinlFamilyMClass);
01332     FacBomManager::addToListAndMap (lSINBKKSegment,
01333                                     lSINBKKSegmentYCabinlFamilyMClass);
01334
01335     lSINBKKSegmentYCabinlFamilyMClass.setYield(500);
01336
01337     // Create a BookingClass (M) for the (mkt) Segment BKK-HKG, cabin Y,
01338     // fare family 1 on SQ's Inv
01339     BookingClass& lMktBKKHKGSegmentYCabinlFamilyMClass =
01340         FacBom<BookingClass>::instance().create (lMBookingClassKey);
01341     FacBomManager::addToListAndMap (lMktBKKHKGSegmentYCabinlFamily,
01342                                     lMktBKKHKGSegmentYCabinlFamilyMClass);
01343     FacBomManager::linkWithParent (lMktBKKHKGSegmentYCabinlFamily,
01344                                     lMktBKKHKGSegmentYCabinlFamilyMClass);
01345
01346     FacBomManager::addToListAndMap (lMktBKKHKGSegmentYCabin,
01347                                     lMktBKKHKGSegmentYCabinlFamilyMClass);
01348     FacBomManager::addToListAndMap (lMktBKKHKGSegment,
01349                                     lMktBKKHKGSegmentYCabinlFamilyMClass);
01350
01351     lMktBKKHKGSegmentYCabinlFamilyMClass.setYield(500);
01352
01353     /* ===== */
01354     // Step 0.9: Partner Inventory
01355     // Create a partner Inventory CX for SQ
01356     const InventoryKey lPartnerCXKey ("CX");
01357     Inventory& lPartnerCXInv = FacBom<Inventory>::instance().create (lPartnerCXKey
01358 y);
01359     FacBomManager::addToListAndMap (lSQInv, lPartnerCXInv);
01360     FacBomManager::linkWithParent (lSQInv, lPartnerCXInv);
01361
01362     // Step 0.9.2 : Flight-date level
01363     lFlightNumber = 12;
01364     lFlightDateKey = FlightDateKey (lFlightNumber, lDate);
01365
01366     FlightDate& lPartnerCX12_20100208_FD =
01367         FacBom<FlightDate>::instance().create (lFlightDateKey);
01368     FacBomManager::addToListAndMap (lPartnerCXInv, lPartnerCX12_20100208_FD);
01369     FacBomManager::linkWithParent (lPartnerCXInv, lPartnerCX12_20100208_FD);
01370

```

```

01370     lFlightNumber = 1100;
01371     lMktFlightDateKey = FlightDateKey (lFlightNumber, lDate);
01372
01373     FlightDate& lPartnerCX1100_20100208_FD =
01374         FacBom<FlightDate>::instance().create (lMktFlightDateKey);
01375     FacBomManager::addToListAndMap (lPartnerCXInv, lPartnerCX1100_20100208_FD);
01376     FacBomManager::linkWithParent (lPartnerCXInv, lPartnerCX1100_20100208_FD);
01377
01378     // Step 0.9.3: Segment-date level
01379     lSegmentDateKey = SegmentDateKey (lBKK, lHKG);
01380
01381     SegmentDate& lPartnerBKKHKGSegment =
01382         FacBom<SegmentDate>::instance().create (lSegmentDateKey);
01383     FacBomManager::addToListAndMap (lPartnerCX12_20100208_FD, lPartnerBKKHKGSegme
01384 nt);
01385     FacBomManager::linkWithParent (lPartnerCX12_20100208_FD, lPartnerBKKHKGSegmen
01386 t);
01387     lPartnerBKKHKGSegment.setBoardingDate (lDate);
01388     lPartnerBKKHKGSegment.setOffDate (lDate);
01389     lPartnerBKKHKGSegment.setBoardingTime (11200);
01390     lPartnerBKKHKGSegment.setOffTime (11540);
01391     lPartnerBKKHKGSegment.setElapsedTime (10240);
01392
01393     lMktSegmentDateKey = SegmentDateKey (lSIN, lBKK);
01394
01395     SegmentDate& lPartnerMktSINBKKSegment =
01396         FacBom<SegmentDate>::instance().create (lMktSegmentDateKey);
01397     FacBomManager::addToListAndMap (lPartnerCX1100_20100208_FD, lPartnerMktSINBKK
01398 Segment);
01399     FacBomManager::linkWithParent (lPartnerCX1100_20100208_FD, lPartnerMktSINBKK
01400 Segment);
01401     lPartnerMktSINBKKSegment.setBoardingDate (lDate);
01402     lPartnerMktSINBKKSegment.setOffDate (lDate);
01403     lPartnerMktSINBKKSegment.setBoardingTime (10820);
01404     lPartnerMktSINBKKSegment.setOffTime (11100);
01405     lPartnerMktSINBKKSegment.setElapsedTime (10340);
01406
01407     // Step 0.9.4: Leg-date level
01408     lLegDateKey = LegDateKey (lBKK);
01409
01410     LegDate& lPartnerBKKLeg = FacBom<LegDate>::instance().create (lLegDateKey);
01411     FacBomManager::addToListAndMap (lPartnerCX12_20100208_FD, lPartnerBKKLeg);
01412     FacBomManager::linkWithParent (lPartnerCX12_20100208_FD, lPartnerBKKLeg);
01413
01414     lPartnerBKKLeg.setOffPoint (lHKG);
01415     lPartnerBKKLeg.setBoardingDate (lDate);
01416     lPartnerBKKLeg.setOffDate (lDate);
01417     lPartnerBKKLeg.setBoardingTime (11200);
01418     lPartnerBKKLeg.setOffTime (11540);
01419     lPartnerBKKLeg.setElapsedTime (10240);
01420
01421     FacBomManager::addToListAndMap (lPartnerBKKLeg, lPartnerBKKHKGSegment);
01422     FacBomManager::addToListAndMap (lPartnerBKKHKGSegment, lPartnerBKKLeg);
01423
01424     // Step 9.0.5: segment-cabin level
01425
01426     SegmentCabin& lPartnerBKKHKGSegmentYCabin =
01427         FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
01428     FacBomManager::addToListAndMap (lPartnerBKKHKGSegment, lPartnerBKKHKGSegmentY
01429 Cabin);

```

```

01427     FacBomManager::linkWithParent (lPartnerBKKHKGSegment, lPartnerBKKHKGSegmentYC
abin);
01428
01429     SegmentCabin& lPartnerMktSINBKKSegmentYCabin =
01430         FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
01431     FacBomManager::addToListAndMap (lPartnerMktSINBKKSegment, lPartnerMktSINBKKSe
gmentYCabin);
01432     FacBomManager::linkWithParent (lPartnerMktSINBKKSegment, lPartnerMktSINBKKSeg
mentYCabin);
01433
01434     // Step 9.0.6: leg-cabin level
01435
01436     LegCabin& lPartnerBKKLegYCabin =
01437         FacBom<LegCabin>::instance().create (lYLegCabinKey);
01438     FacBomManager::addToListAndMap (lPartnerBKKLeg, lPartnerBKKLegYCabin);
01439     FacBomManager::linkWithParent (lPartnerBKKLeg, lPartnerBKKLegYCabin);
01440
01441     lCapacity = CabinCapacity_T(999);
01442     lPartnerBKKLegYCabin.setCapacities (lCapacity);
01443     lPartnerBKKLegYCabin.setAvailabilityPool (lCapacity);
01444
01445     FacBomManager::addToListAndMap (lPartnerBKKLegYCabin, lPartnerBKKHKGSegmentYC
abin,
01446                                     lPartnerBKKHKGSegmentYCabin.getFullerKey());
01447     FacBomManager::addToListAndMap (lPartnerBKKHKGSegmentYCabin, lPartnerBKKLegYC
abin,
01448                                     lPartnerBKKLegYCabin.getFullerKey());
01449
01450     // Step 9.0.7: fare family level
01451
01452     FareFamily& lPartnerBKKHKGSegmentYCabinlFamily =
01453         FacBom<FareFamily>::instance().create (lFareFamilyKey);
01454     FacBomManager::addToListAndMap (lPartnerBKKHKGSegmentYCabin,
01455                                     lPartnerBKKHKGSegmentYCabinlFamily);
01456     FacBomManager::linkWithParent (lPartnerBKKHKGSegmentYCabin,
01457                                     lPartnerBKKHKGSegmentYCabinlFamily);
01458
01459     FareFamily& lPartnerMktSINBKKSegmentYCabinlFamily =
01460         FacBom<FareFamily>::instance().create (lFareFamilyKey);
01461     FacBomManager::addToListAndMap (lPartnerMktSINBKKSegmentYCabin,
01462                                     lPartnerMktSINBKKSegmentYCabinlFamily);
01463     FacBomManager::linkWithParent (lPartnerMktSINBKKSegmentYCabin,
01464                                     lPartnerMktSINBKKSegmentYCabinlFamily);
01465
01466     // Step 9.0.8: booking class level
01467
01468     BookingClass& lPartnerBKKHKGSegmentYCabinlFamilyYClass =
01469         FacBom<BookingClass>::instance().create (lYBookingClassKey);
01470     FacBomManager::addToListAndMap (lPartnerBKKHKGSegmentYCabinlFamily,
01471                                     lPartnerBKKHKGSegmentYCabinlFamilyYClass);
01472     FacBomManager::linkWithParent (lPartnerBKKHKGSegmentYCabinlFamily,
01473                                     lPartnerBKKHKGSegmentYCabinlFamilyYClass);
01474
01475     FacBomManager::addToListAndMap (lPartnerBKKHKGSegmentYCabin,
01476                                     lPartnerBKKHKGSegmentYCabinlFamilyYClass);
01477     FacBomManager::addToListAndMap (lPartnerBKKHKGSegment,
01478                                     lPartnerBKKHKGSegmentYCabinlFamilyYClass);
01479
01480     BookingClass& lPartnerMktSINBKKSegmentYCabinlFamilyYClass =
01481         FacBom<BookingClass>::instance().create (lYBookingClassKey);
01482     FacBomManager::addToListAndMap (lPartnerMktSINBKKSegmentYCabinlFamily,
01483                                     lPartnerMktSINBKKSegmentYCabinlFamilyYClass);

```

```

01484     FacBomManager::linkWithParent (lPartnerMktSINBKKSegmentYCabinlFamily,
01485                                     lPartnerMktSINBKKSegmentYCabinlFamilyYClass);
01486
01487     FacBomManager::addToListAndMap (lPartnerMktSINBKKSegmentYCabin,
01488                                     lPartnerMktSINBKKSegmentYCabinlFamilyYClass);
01489
01489     FacBomManager::addToListAndMap (lPartnerMktSINBKKSegment,
01490                                     lPartnerMktSINBKKSegmentYCabinlFamilyYClass);
01491
01492     BookingClass& lPartnerBKKHKGSegmentYCabinlFamilyMClass =
01493         FacBom<BookingClass>::instance().create (lMBookingClassKey);
01494     FacBomManager::addToListAndMap (lPartnerBKKHKGSegmentYCabinlFamily,
01495                                     lPartnerBKKHKGSegmentYCabinlFamilyMClass);
01496     FacBomManager::linkWithParent (lPartnerBKKHKGSegmentYCabinlFamily,
01497                                     lPartnerBKKHKGSegmentYCabinlFamilyMClass);
01498
01499     FacBomManager::addToListAndMap (lPartnerBKKHKGSegmentYCabin,
01500                                     lPartnerBKKHKGSegmentYCabinlFamilyMClass);
01501     FacBomManager::addToListAndMap (lPartnerBKKHKGSegment,
01502                                     lPartnerBKKHKGSegmentYCabinlFamilyMClass);
01503
01504     BookingClass& lPartnerMktSINBKKSegmentYCabinlFamilyMClass =
01505         FacBom<BookingClass>::instance().create (lMBookingClassKey);
01506     FacBomManager::addToListAndMap (lPartnerMktSINBKKSegmentYCabinlFamily,
01507                                     lPartnerMktSINBKKSegmentYCabinlFamilyMClass);
01508
01508     FacBomManager::linkWithParent (lPartnerMktSINBKKSegmentYCabinlFamily,
01509                                     lPartnerMktSINBKKSegmentYCabinlFamilyMClass);
01510
01511     FacBomManager::addToListAndMap (lPartnerMktSINBKKSegmentYCabin,
01512                                     lPartnerMktSINBKKSegmentYCabinlFamilyMClass);
01513
01513     FacBomManager::addToListAndMap (lPartnerMktSINBKKSegment,
01514                                     lPartnerMktSINBKKSegmentYCabinlFamilyMClass);
01515
01515     // Step 9.0.9: link SQ inventory objects to Partner CX inventory objects
01517
01518     FacBomManager::addToList (lSINBKKSegment, lPartnerMktSINBKKSegment);
01519
01520     lMktBKKHKGSegment.linkWithOperating (lPartnerBKKHKGSegment);
01521
01522     /* =====
01523     ===== */
01524
01524     // Step 1.0: O&D level
01525     // Create an O&D Date (SQ11/08-FEB-2010/SIN-BKK-SQ1200/08-FEB-2010/BKK-HKG) f
01526     or SQ's Inventory
01527     OnDString_T lSQSINBKKOnDStr = "SQ;11,2010-Feb-08;SIN,BKK";
01528     OnDString_T lMktSQBKKHKGOnDStr = "SQ;1200,2010-Feb-08;BKK,HKG";
01529     OnDStringList_T lOnDStringList;
01530     lOnDStringList.push_back (lSQSINBKKOnDStr);
01531     lOnDStringList.push_back (lMktSQBKKHKGOnDStr);
01532
01532     OnDDateKey lOnDDateKey (lOnDStringList);
01533     OnDDate& lSQ_SINHKG_OnDDate =
01534         FacBom<OnDDate>::instance().create (lOnDDateKey);
01535     // Link to the inventory
01536     FacBomManager::addToListAndMap (lSQInv, lSQ_SINHKG_OnDDate);
01537     FacBomManager::linkWithParent (lSQInv, lSQ_SINHKG_OnDDate);

```

```

01538
01539     // Add the segments
01540     FacBomManager::addToListAndMap (lSQ_SINHKG_OnDDate, lSINBKKSegment);
01541     FacBomManager::addToListAndMap (lSQ_SINHKG_OnDDate, lMktBKKHKGSegment);
01542
01543     // Add total forecast info for cabin Y.
01544     const MeanStdDevPair_T lMean60StdDev6 (60.0, 6.0);
01545     const WTP_T lWTP750 = 750.0;
01546     const WTPDemandPair_T lWTP750Mean60StdDev6 (lWTP750, lMean60StdDev6);
01547     lSQ_SINHKG_OnDDate.setTotalForecast (lY, lWTP750Mean60StdDev6);
01548
01549     // Add demand info (optional).
01550     // 2 legs here, so 2 CabinClassPair to add in the list.
01551     // First leg: cabin Y, class M.
01552     CabinClassPair_T lCC_YM1 (lY, lM);
01553     // Second leg: cabin Y, class M too.
01554     CabinClassPair_T lCC_YM2 (lY, lM);
01555     CabinClassPairList_T lCabinClassPairList;
01556     lCabinClassPairList.push_back(lCC_YM1);
01557     lCabinClassPairList.push_back(lCC_YM2);
01558     const MeanStdDevPair_T lMean20StdDev2 (20.0, 2.0);
01559     const Yield_T lYield850 = 850.0;
01560     const YieldDemandPair_T lYield850Mean20StdDev2 (lYield850, lMean20StdDev2);
01561     lSQ_SINHKG_OnDDate.setDemandInformation (lCabinClassPairList, lYield850Mean20
StdDev2);
01562
01563     CabinClassPair_T lCC_YY1 (lY, lY);
01564     CabinClassPair_T lCC_YY2 (lY, lY);
01565     lCabinClassPairList.clear();
01566     lCabinClassPairList.push_back(lCC_YY1);
01567     lCabinClassPairList.push_back(lCC_YY2);
01568     const MeanStdDevPair_T lMean10StdDev1 (10.0, 1.0);
01569     const Yield_T lYield1200 = 1200.0;
01570     const YieldDemandPair_T lYield1200Mean10StdDev1 (lYield1200, lMean10StdDev1);
01571
01572     lSQ_SINHKG_OnDDate.setDemandInformation (lCabinClassPairList, lYield1200Mean1
0StdDev1);
01572
01573     // Create an O&D Date (SQ11/08-FEB-2010/SIN-BKK) for SQ's Inventory
01574     lOnDStringList.clear();
01575     lOnDStringList.push_back (lSQ_SINBKKOnDStr);
01576
01577     lOnDDateKey = OnDDateKey(lOnDStringList);
01578     OnDDate& lSQ_SINBKK_OnDDate =
01579         FacBom<OnDDate>::instance().create (lOnDDateKey);
01580     // Link to the inventory
01581     FacBomManager::addToListAndMap (lSQ_Inv, lSQ_SINBKK_OnDDate);
01582     FacBomManager::linkWithParent (lSQ_Inv, lSQ_SINBKK_OnDDate);
01583
01584     // Add the segments
01585     FacBomManager::addToListAndMap (lSQ_SINBKK_OnDDate, lSINBKKSegment);
01586
01587     // Add total forecast info for cabin Y.
01588     const WTP_T lWTP400 = 400.0;
01589     const WTPDemandPair_T lWTP400Mean60StdDev6 (lWTP400, lMean60StdDev6);
01590     lSQ_SINBKK_OnDDate.setTotalForecast (lY, lWTP400Mean60StdDev6);
01591
01592     // Add demand info (optional).
01593     lCabinClassPairList.clear();
01594     lCabinClassPairList.push_back(lCC_YM1);
01595     const MeanStdDevPair_T lMean20StdDev1 (20.0, 1.0);
01596     const Yield_T lYield500 = 500.0;

```

```

01597     const YieldDemandPair_T lYield500Mean20StdDev1 (lYield500, lMean20StdDev1);
01598     lSQ_SINBKK_OnDDate.setDemandInformation (lCabinClassPairList, lYield500Mean20
StdDev1);
01599
01600     lCabinClassPairList.clear();
01601     lCabinClassPairList.push_back(lCC_YY1);
01602     const Yield_T lYield700 = 700.0;
01603     const YieldDemandPair_T lYield700Mean20StdDev1 (lYield700, lMean10StdDev1 );
01604     lSQ_SINBKK_OnDDate.setDemandInformation (lCabinClassPairList, lYield700Mean20
StdDev1);
01605
01606     /*****
***
01607     // Create an O&D Date (SQ1200/08-FEB-2010/BKK-HKG) for SQ's Inventory
01608     lFullKeyList.clear();
01609     lFullKeyList.push_back (lMktSQBKKHKGFullKeyStr);
01610
01611     lOnDDateKey = OnDDateKey(lFullKeyList);
01612     OnDDate& lMktSQ_BKKHKG_OnDDate =
01613         FacBom<OnDDate>::instance().create (lOnDDateKey);
01614     // Link to the inventory
01615     FacBomManager::addToListAndMap (lSQInv, lMktSQ_BKKHKG_OnDDate);
01616     FacBomManager::linkWithParent (lSQInv, lMktSQ_BKKHKG_OnDDate);
01617
01618     // Add the segments
01619     FacBomManager::addToListAndMap (lMktSQ_BKKHKG_OnDDate, lMktBKKHKGSegment);
01620
01621     // Demand info is not added for purely marketed O&Ds
01622     // Add demand info
01623     // lCabinClassPairList.clear();
01624     // lCabinClassPairList.push_back(lCC_YM2);
01625     // lMktSQ_BKKHKG_OnDDate.setDemandInformation (lCabinClassPairList, 500.0, 20
.0, 1.0);
01626     *****/
01627
01628
01629     // ===== CX =====
01630     // Step 0.2: Flight-date level
01631     // Create a FlightDate (CX12/08-FEB-2010) for CX's Inventory
01632     lFlightNumber = 12;
01633     //lDate = Date_T (2010, 2, 8);
01634     lFlightDateKey = FlightDateKey (lFlightNumber, lDate);
01635
01636     FlightDate& lCX12_20100208_FD =
01637         FacBom<FlightDate>::instance().create (lFlightDateKey);
01638     FacBomManager::addToListAndMap (lCXInv, lCX12_20100208_FD);
01639     FacBomManager::linkWithParent (lCXInv, lCX12_20100208_FD);
01640
01641     // Create a (mkt) FlightDate (CX1100/08-FEB-2010) for CX's Inventory
01642     lFlightNumber = 1100;
01643     //lDate = Date_T (2010, 2, 8);
01644     lMktFlightDateKey = FlightDateKey (lFlightNumber, lDate);
01645
01646     FlightDate& lCX1100_20100208_FD =
01647         FacBom<FlightDate>::instance().create (lMktFlightDateKey);
01648     FacBomManager::addToListAndMap (lCXInv, lCX1100_20100208_FD);
01649     FacBomManager::linkWithParent (lCXInv, lCX1100_20100208_FD);
01650
01651     // Display the flight-date
01652     // STDAIR_LOG_DEBUG ("FlightDate: " << lAF084_20110320_FD.toString());
01653

```

```

01654 // Step 0.3: Segment-date level
01655 // Create a SegmentDate BKK-HKG for CX's Inventory
01656
01657 lSegmentDateKey = SegmentDateKey (lBKK, lHKG);
01658
01659 SegmentDate& lBKKHKGSegment =
01660     FacBom<SegmentDate>::instance().create (lSegmentDateKey);
01661 FacBomManager::addToListAndMap (lCX12_20100208_FD, lBKKHKGSegment);
01662 FacBomManager::linkWithParent (lCX12_20100208_FD, lBKKHKGSegment);
01663
01664 // Fill the SegmentDate content
01665 lBKKHKGSegment.setBoardingDate (lDate);
01666 lBKKHKGSegment.setOffDate (lDate);
01667 lBKKHKGSegment.setBoardingTime (l1200);
01668 lBKKHKGSegment.setOffTime (l1540);
01669 lBKKHKGSegment.setElapsedTime (l0240);
01670
01671 // Create a second (mkt) SegmentDate (SIN-BKK) for CX's Inventory
01672 lMktSegmentDateKey = SegmentDateKey (lSIN, lBKK);
01673
01674 SegmentDate& lMktSINBKKSegment =
01675     FacBom<SegmentDate>::instance().create (lMktSegmentDateKey);
01676 FacBomManager::addToListAndMap (lCX1100_20100208_FD, lMktSINBKKSegment);
01677 FacBomManager::linkWithParent (lCX1100_20100208_FD, lMktSINBKKSegment);
01678
01679 // Fill the (mkt) SegmentDate content
01680 lMktSINBKKSegment.setBoardingDate (lDate);
01681 lMktSINBKKSegment.setOffDate (lDate);
01682 lMktSINBKKSegment.setBoardingTime (l0820);
01683 lMktSINBKKSegment.setOffTime (l1100);
01684 lMktSINBKKSegment.setElapsedTime (l0340);
01685
01686 // Step 0.4: Leg-date level
01687 // Create a LegDate (BKK) for CX's Inventory
01688 lLegDateKey = LegDateKey (lBKK);
01689
01690 LegDate& lBKKLeg = FacBom<LegDate>::instance().create (lLegDateKey);
01691 FacBomManager::addToListAndMap (lCX12_20100208_FD, lBKKLeg);
01692 FacBomManager::linkWithParent (lCX12_20100208_FD, lBKKLeg);
01693
01694 // Fill the LegDate content
01695 lBKKLeg.setOffPoint (lHKG);
01696 lBKKLeg.setBoardingDate (lDate);
01697 lBKKLeg.setOffDate (lDate);
01698 lBKKLeg.setBoardingTime (l1200);
01699 lBKKLeg.setOffTime (l1540);
01700 lBKKLeg.setElapsedTime (l0240);
01701
01702 // Display the leg-date
01703 // STDAIR_LOG_DEBUG ("LegDate: " << lCDGLeg.toString());
01704
01705 // Link the segment-dates with the leg-dates
01706 FacBomManager::addToListAndMap (lBKKLeg, lBKKHKGSegment);
01707 FacBomManager::addToListAndMap (lBKKHKGSegment, lBKKLeg);
01708
01709 // Step 0.5: segment-cabin level
01710 // Create a SegmentCabin (Y) for the Segment BKK-HKG of CX's Inventory
01711 SegmentCabin& lBKKHKGSegmentYCabin =
01712     FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
01713 FacBomManager::addToListAndMap (lBKKHKGSegment, lBKKHKGSegmentYCabin);
01714 FacBomManager::linkWithParent (lBKKHKGSegment, lBKKHKGSegmentYCabin);
01715

```

```

01716 // Create a SegmentCabin (Y) for the (mkt) Segment SIN-BKK of CX's Inventory
01717 SegmentCabin& lMktSINBKKSegmentYCabin =
01718     FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
01719 FacBomManager::addToListAndMap (lMktSINBKKSegment, lMktSINBKKSegmentYCabin);
01720 FacBomManager::linkWithParent (lMktSINBKKSegment, lMktSINBKKSegmentYCabin);
01721
01722 // Step 0.6: leg-cabin level
01723 // Create a LegCabin (Y) for the Leg BKK-HKG on CX's Inventory
01724 LegCabin& lBKKLegYCabin =
01725     FacBom<LegCabin>::instance().create (lYLegCabinKey);
01726 FacBomManager::addToListAndMap (lBKKLeg, lBKKLegYCabin);
01727 FacBomManager::linkWithParent (lBKKLeg, lBKKLegYCabin);
01728
01729 lCapacity = CabinCapacity_T(100);
01730 lBKKLegYCabin.setCapacities (lCapacity);
01731 lBKKLegYCabin.setAvailabilityPool (lCapacity);
01732
01733 // Link the segment-dates with the leg-dates
01734 FacBomManager::addToListAndMap (lBKKLegYCabin, lBKKHKGSegmentYCabin,
01735     lBKKHKGSegmentYCabin.getFullerKey());
01736 FacBomManager::addToListAndMap (lBKKHKGSegmentYCabin, lBKKLegYCabin,
01737     lBKKLegYCabin.getFullerKey());
01738
01739 // Step 0.7: fare family level
01740 // Create a fareFamily (1) for the Segment BKK-HKG, cabin Y on CX's Inv
01741 FareFamily& lBKKHKGSegmentYCabinlFamily =
01742     FacBom<FareFamily>::instance().create (l1FareFamilyKey);
01743 FacBomManager::addToListAndMap (lBKKHKGSegmentYCabin,
01744     lBKKHKGSegmentYCabinlFamily);
01745 FacBomManager::linkWithParent (lBKKHKGSegmentYCabin,
01746     lBKKHKGSegmentYCabinlFamily);
01747
01748 // Create a FareFamily (1) for the (mkt) Segment SIN-BKK, cabin Y on CX's Inv
01749 FareFamily& lMktSINBKKSegmentYCabinlFamily =
01750     FacBom<FareFamily>::instance().create (l1FareFamilyKey);
01751 FacBomManager::addToListAndMap (lMktSINBKKSegmentYCabin,
01752     lMktSINBKKSegmentYCabinlFamily);
01753 FacBomManager::linkWithParent (lMktSINBKKSegmentYCabin,
01754     lMktSINBKKSegmentYCabinlFamily);
01755
01756 // Step 0.8: booking class level
01757 // Create a BookingClass (Y) for the
01758 // Segment BKK-HKG, cabin Y, fare family 1 on CX's Inv
01759 BookingClass& lBKKHKGSegmentYCabinlFamilyYClass =
01760     FacBom<BookingClass>::instance().create (lYBookingClassKey);
01761 FacBomManager::addToListAndMap (lBKKHKGSegmentYCabinlFamily,
01762     lBKKHKGSegmentYCabinlFamilyYClass);
01763 FacBomManager::linkWithParent (lBKKHKGSegmentYCabinlFamily,
01764     lBKKHKGSegmentYCabinlFamilyYClass);
01765
01766 FacBomManager::addToListAndMap (lBKKHKGSegmentYCabin,
01767     lBKKHKGSegmentYCabinlFamilyYClass);
01768 FacBomManager::addToListAndMap (lBKKHKGSegment,
01769     lBKKHKGSegmentYCabinlFamilyYClass);
01770
01771 lBKKHKGSegmentYCabinlFamilyYClass.setYield(700);
01772
01773 // Create a BookingClass (Y) for the (mkt) Segment SIN-BKK, cabin Y,
01774 // fare family 1 on CX's Inv
01775 BookingClass& lMktSINBKKSegmentYCabinlFamilyYClass =

```

```

01777     FacBom<BookingClass>::instance().create (lYBookingClassKey);
01778     FacBomManager::addToListAndMap (lMktSINBKKSegmentYCabinlFamily,
01779                                     lMktSINBKKSegmentYCabinlFamilyYClass);
01780     FacBomManager::linkWithParent (lMktSINBKKSegmentYCabinlFamily,
01781                                    lMktSINBKKSegmentYCabinlFamilyYClass);
01782
01783     FacBomManager::addToListAndMap (lMktSINBKKSegmentYCabin,
01784                                     lMktSINBKKSegmentYCabinlFamilyYClass);
01785     FacBomManager::addToListAndMap (lMktSINBKKSegment,
01786                                     lMktSINBKKSegmentYCabinlFamilyYClass);
01787
01788     lMktSINBKKSegmentYCabinlFamilyYClass.setYield(700);
01789
01790     //Create a BookingClass (M) for the
01791     // Segment BKK-HKG, cabin Y, fare family 1 on CX's Inv
01792     BookingClass& lBKKHKGSegmentYCabinlFamilyMClass =
01793         FacBom<BookingClass>::instance().create (lMBookingClassKey);
01794     FacBomManager::addToListAndMap (lBKKHKGSegmentYCabinlFamily,
01795                                     lBKKHKGSegmentYCabinlFamilyMClass);
01796     FacBomManager::linkWithParent (lBKKHKGSegmentYCabinlFamily,
01797                                    lBKKHKGSegmentYCabinlFamilyMClass);
01798
01799     FacBomManager::addToListAndMap (lBKKHKGSegmentYCabin,
01800                                     lBKKHKGSegmentYCabinlFamilyMClass);
01801     FacBomManager::addToListAndMap (lBKKHKGSegment,
01802                                     lBKKHKGSegmentYCabinlFamilyMClass);
01803
01804     lBKKHKGSegmentYCabinlFamilyMClass.setYield(500);
01805
01806     // Create a BookingClass (M) for the (mkt) Segment SIN-BKK, cabin Y,
01807     // fare family 1 on CX's Inv
01808     BookingClass& lMktSINBKKSegmentYCabinlFamilyMClass =
01809         FacBom<BookingClass>::instance().create (lMBookingClassKey);
01810     FacBomManager::addToListAndMap (lMktSINBKKSegmentYCabinlFamily,
01811                                     lMktSINBKKSegmentYCabinlFamilyMClass);
01812     FacBomManager::linkWithParent (lMktSINBKKSegmentYCabinlFamily,
01813                                    lMktSINBKKSegmentYCabinlFamilyMClass);
01814
01815     FacBomManager::addToListAndMap (lMktSINBKKSegmentYCabin,
01816                                     lMktSINBKKSegmentYCabinlFamilyMClass);
01817     FacBomManager::addToListAndMap (lMktSINBKKSegment,
01818                                     lMktSINBKKSegmentYCabinlFamilyMClass);
01819
01820     lMktSINBKKSegmentYCabinlFamilyMClass.setYield(500);
01821
01822     /* ===== */
01823     // Step 0.9: Partner Inventory
01824     // Create a partner Inventory SQ for CX
01825     const InventoryKey lPartnerSQKey ("SQ");
01826     Inventory& lPartnerSQInv = FacBom<Inventory>::instance().create (lPartnerSQKey
01827 y);
01827     FacBomManager::addToListAndMap (lCXInv, lPartnerSQInv);
01828     FacBomManager::linkWithParent (lCXInv, lPartnerSQInv);
01829
01830     // Step 0.9.2 : Flight-date level
01831     lFlightNumber = 11;
01832     lFlightDateKey = FlightDateKey (lFlightNumber, lDate);
01833
01834     FlightDate& lPartnerSQ11_20100208_FD =
01835         FacBom<FlightDate>::instance().create (lFlightDateKey);
01836     FacBomManager::addToListAndMap (lPartnerSQInv, lPartnerSQ11_20100208_FD);

```

```

01837     FacBomManager::linkWithParent (lPartnerSQInv, lPartnerSQ11_20100208_FD);
01838
01839     lFlightNumber = 1200;
01840     lMktFlightDateKey = FlightDateKey (lFlightNumber, lDate);
01841
01842     FlightDate& lPartnerSQL200_20100208_FD =
01843         FacBom<FlightDate>::instance().create (lMktFlightDateKey);
01844     FacBomManager::addToListAndMap (lPartnerSQInv, lPartnerSQL200_20100208_FD);
01845     FacBomManager::linkWithParent (lPartnerSQInv, lPartnerSQL200_20100208_FD);
01846
01847     // Step 0.9.3: Segment-date level
01848     lSegmentDateKey = SegmentDateKey (lSIN, lBKK);
01849
01850     SegmentDate& lPartnerSINBKKSegment =
01851         FacBom<SegmentDate>::instance().create (lSegmentDateKey);
01852     FacBomManager::addToListAndMap (lPartnerSQ11_20100208_FD, lPartnerSINBKKSegme
01853 nt);
01854     FacBomManager::linkWithParent (lPartnerSQ11_20100208_FD, lPartnerSINBKKSegmen
01855 t);
01856
01857     lPartnerSINBKKSegment.setBoardingDate (lDate);
01858     lPartnerSINBKKSegment.setOffDate (lDate);
01859     lPartnerSINBKKSegment.setBoardingTime (10820);
01860     lPartnerSINBKKSegment.setOffTime (11100);
01861     lPartnerSINBKKSegment.setElapsedTime (10340);
01862
01863     lMktSegmentDateKey = SegmentDateKey (lBKK, lHKG);
01864
01865     SegmentDate& lPartnerMktBKKHKGSegment =
01866         FacBom<SegmentDate>::instance().create (lMktSegmentDateKey);
01867     FacBomManager::addToListAndMap (lPartnerSQL200_20100208_FD, lPartnerMktBKKHKG
01868 Segment);
01869     FacBomManager::linkWithParent (lPartnerSQL200_20100208_FD, lPartnerMktBKKHKG
01870 Segment);
01871
01872     lPartnerMktBKKHKGSegment.setBoardingDate (lDate);
01873     lPartnerMktBKKHKGSegment.setOffDate (lDate);
01874     lPartnerMktBKKHKGSegment.setBoardingTime (11200);
01875     lPartnerMktBKKHKGSegment.setOffTime (11540);
01876     lPartnerMktBKKHKGSegment.setElapsedTime (10240);
01877
01878     // Step 0.9.4: Leg-date level
01879     lLegDateKey = LegDateKey (lSIN);
01880
01881     LegDate& lPartnerSINLeg = FacBom<LegDate>::instance().create (lLegDateKey);
01882     FacBomManager::addToListAndMap (lPartnerSQ11_20100208_FD, lPartnerSINLeg);
01883     FacBomManager::linkWithParent (lPartnerSQ11_20100208_FD, lPartnerSINLeg);
01884
01885     lPartnerSINLeg.setOffPoint (lBKK);
01886     lPartnerSINLeg.setBoardingDate (lDate);
01887     lPartnerSINLeg.setOffDate (lDate);
01888     lPartnerSINLeg.setBoardingTime (10820);
01889     lPartnerSINLeg.setOffTime (11100);
01890     lPartnerSINLeg.setElapsedTime (10340);
01891
01892     FacBomManager::addToListAndMap (lPartnerSINLeg, lPartnerSINBKKSegment);
01893     FacBomManager::addToListAndMap (lPartnerSINBKKSegment, lPartnerSINLeg);
01894
01895     // Step 9.0.5: segment-cabin level
01896
01897     SegmentCabin& lPartnerSINBKKSegmentYCabin =
01898         FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);

```

```

01895     FacBomManager::addToListAndMap (lPartnerSINBKKSegment, lPartnerSINBKKSegmentY
Cabin);
01896     FacBomManager::linkWithParent (lPartnerSINBKKSegment, lPartnerSINBKKSegmentYC
abin);
01897
01898     SegmentCabin& lPartnerMktBKKHKGSegmentYCabin =
01899         FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
01900     FacBomManager::addToListAndMap (lPartnerMktBKKHKGSegment, lPartnerMktBKKHKGSe
gmentYCabin);
01901     FacBomManager::linkWithParent (lPartnerMktBKKHKGSegment, lPartnerMktBKKHKGSe
gmentYCabin);
01902
01903     // Step 9.0.6: leg-cabin level
01904
01905     LegCabin& lPartnerSINLegYCabin =
01906         FacBom<LegCabin>::instance().create (lYLegCabinKey);
01907     FacBomManager::addToListAndMap (lPartnerSINLeg, lPartnerSINLegYCabin);
01908     FacBomManager::linkWithParent (lPartnerSINLeg, lPartnerSINLegYCabin);
01909
01910     lCapacity = CabinCapacity_T(999);
01911     lPartnerSINLegYCabin.setCapacities (lCapacity);
01912     lPartnerSINLegYCabin.setAvailabilityPool (lCapacity);
01913
01914     FacBomManager::addToListAndMap (lPartnerSINLegYCabin, lPartnerSINBKKSegmentYC
abin,
01915                                     lPartnerSINBKKSegmentYCabin.getFullerKey());
01916     FacBomManager::addToListAndMap (lPartnerSINBKKSegmentYCabin, lPartnerSINLegYC
abin,
01917                                     lPartnerSINLegYCabin.getFullerKey());
01918
01919     // Step 9.0.7: fare family level
01920
01921     FareFamily& lPartnerSINBKKSegmentYCabinlFamily =
01922         FacBom<FareFamily>::instance().create (lFareFamilyKey);
01923     FacBomManager::addToListAndMap (lPartnerSINBKKSegmentYCabin,
01924                                     lPartnerSINBKKSegmentYCabinlFamily);
01925     FacBomManager::linkWithParent (lPartnerSINBKKSegmentYCabin,
01926                                     lPartnerSINBKKSegmentYCabinlFamily);
01927
01928     FareFamily& lPartnerMktBKKHKGSegmentYCabinlFamily =
01929         FacBom<FareFamily>::instance().create (lFareFamilyKey);
01930     FacBomManager::addToListAndMap (lPartnerMktBKKHKGSegmentYCabin,
01931                                     lPartnerMktBKKHKGSegmentYCabinlFamily);
01932     FacBomManager::linkWithParent (lPartnerMktBKKHKGSegmentYCabin,
01933                                     lPartnerMktBKKHKGSegmentYCabinlFamily);
01934
01935     // Step 9.0.8: booking class level
01936
01937     BookingClass& lPartnerSINBKKSegmentYCabinlFamilyYClass =
01938         FacBom<BookingClass>::instance().create (lYBookingClassKey);
01939     FacBomManager::addToListAndMap (lPartnerSINBKKSegmentYCabinlFamily,
01940                                     lPartnerSINBKKSegmentYCabinlFamilyYClass);
01941     FacBomManager::linkWithParent (lPartnerSINBKKSegmentYCabinlFamily,
01942                                     lPartnerSINBKKSegmentYCabinlFamilyYClass);
01943
01944     FacBomManager::addToListAndMap (lPartnerSINBKKSegmentYCabin,
01945                                     lPartnerSINBKKSegmentYCabinlFamilyYClass);
01946     FacBomManager::addToListAndMap (lPartnerSINBKKSegment,
01947                                     lPartnerSINBKKSegmentYCabinlFamilyYClass);
01948
01949     BookingClass& lPartnerMktBKKHKGSegmentYCabinlFamilyYClass =

```

```

01951         FacBom<BookingClass>::instance().create (lYBookingClassKey);
01952         FacBomManager::addToListAndMap (lPartnerMktBKKHKGSegmentYCabinlFamily,
01953                                         lPartnerMktBKKHKGSegmentYCabinlFamilyYClass);

01954         FacBomManager::linkWithParent (lPartnerMktBKKHKGSegmentYCabinlFamily,
01955                                         lPartnerMktBKKHKGSegmentYCabinlFamilyYClass);
01956
01957         FacBomManager::addToListAndMap (lPartnerMktBKKHKGSegmentYCabin,
01958                                         lPartnerMktBKKHKGSegmentYCabinlFamilyYClass);

01959         FacBomManager::addToListAndMap (lPartnerMktBKKHKGSegment,
01960                                         lPartnerMktBKKHKGSegmentYCabinlFamilyYClass);

01961
01962         BookingClass& lPartnerSINBKKSegmentYCabinlFamilyMClass =
01963             FacBom<BookingClass>::instance().create (lMBookingClassKey);
01964         FacBomManager::addToListAndMap (lPartnerSINBKKSegmentYCabinlFamily,
01965                                         lPartnerSINBKKSegmentYCabinlFamilyMClass);
01966         FacBomManager::linkWithParent (lPartnerSINBKKSegmentYCabinlFamily,
01967                                         lPartnerSINBKKSegmentYCabinlFamilyMClass);
01968
01969         FacBomManager::addToListAndMap (lPartnerSINBKKSegmentYCabin,
01970                                         lPartnerSINBKKSegmentYCabinlFamilyMClass);
01971         FacBomManager::addToListAndMap (lPartnerSINBKKSegment,
01972                                         lPartnerSINBKKSegmentYCabinlFamilyMClass);
01973
01974         BookingClass& lPartnerMktBKKHKGSegmentYCabinlFamilyMClass =
01975             FacBom<BookingClass>::instance().create (lMBookingClassKey);
01976         FacBomManager::addToListAndMap (lPartnerMktBKKHKGSegmentYCabinlFamily,
01977                                         lPartnerMktBKKHKGSegmentYCabinlFamilyMClass);

01978         FacBomManager::linkWithParent (lPartnerMktBKKHKGSegmentYCabinlFamily,
01979                                         lPartnerMktBKKHKGSegmentYCabinlFamilyMClass);
01980
01981         FacBomManager::addToListAndMap (lPartnerMktBKKHKGSegmentYCabin,
01982                                         lPartnerMktBKKHKGSegmentYCabinlFamilyMClass);

01983         FacBomManager::addToListAndMap (lPartnerMktBKKHKGSegment,
01984                                         lPartnerMktBKKHKGSegmentYCabinlFamilyMClass);

01985
01986         // Step 9.0.9: link CX inventory objects to Partner SQ inventory objects
01987         FacBomManager::addToList (lBKKHKGSegment, lPartnerMktBKKHKGSegment);
01988
01989         lMktSINBKKSegment.linkWithOperating (lPartnerSINBKKSegment);
01990
01991         /* ===== */
01992         // Step 1.0: O&D level
01993         // Create an O&D Date (CX1100/08-FEB-2010/SIN-BKK-CX12/08-FEB-2010/BKK-HKG) f
01994         or CX's Inventory
01995         OnDString_T lMktCXSinBKKOnDStr = "CX;1100,2010-Feb-08;SIN,BKK";
01996         OnDString_T lCXBKKHKGOnDStr = "CX;12,2010-Feb-08;BKK,HKG";
01997         lOnDStringList.clear();
01998         lOnDStringList.push_back (lMktCXSinBKKOnDStr);
01999         lOnDStringList.push_back (lCXBKKHKGOnDStr);
02000
02001         lOnDDateKey = OnDDateKey(lOnDStringList);
02002         OnDDate& lCX_SINHKG_OnDDate =
02003             FacBom<OnDDate>::instance().create (lOnDDateKey);
02004         // Link to the inventory

```

```

02005     FacBomManager::addToListAndMap (lCXInv, lCX_SINHKG_OnDDate);
02006     FacBomManager::linkWithParent (lCXInv, lCX_SINHKG_OnDDate);
02007
02008     // Add the segments
02009     FacBomManager::addToListAndMap (lCX_SINHKG_OnDDate, lMktSINBKKSegment);
02010     FacBomManager::addToListAndMap (lCX_SINHKG_OnDDate, lBKKHKGSegment);
02011
02012     // Add total forecast info for cabin Y.
02013     lCX_SINHKG_OnDDate.setTotalForecast (lY, lWTP750Mean60StdDev6);
02014
02015     // Add demand info
02016     lCabinClassPairList.clear();
02017     lCabinClassPairList.push_back(lCC_YM1);
02018     lCabinClassPairList.push_back(lCC_YM2);
02019     lCX_SINHKG_OnDDate.setDemandInformation (lCabinClassPairList, lYield850Mean20
StdDev2);
02020
02021     lCabinClassPairList.clear();
02022     lCabinClassPairList.push_back(lCC_YY1);
02023     lCabinClassPairList.push_back(lCC_YY2);
02024     lCX_SINHKG_OnDDate.setDemandInformation (lCabinClassPairList, lYield1200Mean1
0StdDev1);
02025
02026     /*****
02027     // Create an O&D Date (CX1100/08-FEB-2010/SIN-BKK) for CX's Inventory
02028     lFullKeyList.clear();
02029     lFullKeyList.push_back (lMktCX_SINBKKFullKeyStr);
02030
02031     lOnDDateKey = OnDDateKey(lFullKeyList);
02032     OnDDate& lMktCX_SINBKK_OnDDate =
02033         FacBom<OnDDate>::instance().create (lOnDDateKey);
02034     // Link to the inventory
02035     FacBomManager::addToListAndMap (lCXInv, lMktCX_SINBKK_OnDDate);
02036     FacBomManager::linkWithParent (lCXInv, lMktCX_SINBKK_OnDDate);
02037
02038     // Add the segments
02039     FacBomManager::addToListAndMap (lMktCX_SINBKK_OnDDate, lMktSINBKKSegment);
02040
02041     // Demand info is not added for purely marketed O&Ds
02042     // Add demand info
02043     // lCabinClassPairList.clear();
02044     // lCabinClassPairList.push_back(lCC_YM1);
02045     // lMktCX_SINBKK_OnDDate.setDemandInformation (lCabinClassPairList, 500.0, 20
.0, 1.0);
02046     *****/
02047
02048     // Create an O&D Date (CX12/08-FEB-2010/BKK-HKG) for CX's Inventory
02049     lOnDStringList.clear();
02050     lOnDStringList.push_back (lCXBKKHKGOnDStr);
02051
02052     lOnDDateKey = OnDDateKey(lOnDStringList);
02053     OnDDate& lCX_BKKHKG_OnDDate =
02054         FacBom<OnDDate>::instance().create (lOnDDateKey);
02055     // Link to the inventory
02056     FacBomManager::addToListAndMap (lCXInv, lCX_BKKHKG_OnDDate);
02057     FacBomManager::linkWithParent (lCXInv, lCX_BKKHKG_OnDDate);
02058
02059     // Add the segments
02060     FacBomManager::addToListAndMap (lCX_BKKHKG_OnDDate, lBKKHKGSegment);
02061

```

```

02062     // Add total forecast info for cabin Y.
02063     lCX_BKKHKG_OnDDate.setTotalForecast (lY, lWTP400Mean60StdDev6);
02064
02065     // Add demand info
02066     lCabinClassPairList.clear();
02067     lCabinClassPairList.push_back(lCC_YM2);
02068     lCX_BKKHKG_OnDDate.setDemandInformation (lCabinClassPairList, lYield500Mean20
StdDev1);
02069
02070     lCabinClassPairList.clear();
02071     lCabinClassPairList.push_back(lCC_YY2);
02072     const YieldDemandPair_T lYield700Mean10StdDev1 (lYield700, lMean10StdDev1 );
02073     lCX_BKKHKG_OnDDate.setDemandInformation (lCabinClassPairList, lYield700Mean10
StdDev1);
02074
02075     /*=====
=====
02076     =====
=====
02077     =====*/
02078     // Schedule:
02079     // SQ:
02080     // Step 1: flight period level
02081     // Create a flight period for SQ11:
02082     const DoWStruct lDoWSrtuct ("1111111");
02083     const Date_T lDateRangeStart (2010, boost::gregorian::Feb, 8);
02084     const Date_T lDateRangeEnd (2010, boost::gregorian::Feb, 9);
02085     const DatePeriod_T lDatePeriod (lDateRangeStart, lDateRangeEnd);
02086     const PeriodStruct lPeriodStruct (lDatePeriod, lDoWSrtuct);
02087
02088     lFlightNumber = FlightNumber_T (11);
02089
02090     FlightPeriodKey lFlightPeriodKey (lFlightNumber, lPeriodStruct);
02091
02092     FlightPeriod& lSQ11FlightPeriod =
02093         FacBom<FlightPeriod>::instance().create (lFlightPeriodKey);
02094     FacBomManager::addToListAndMap (lSQInv, lSQ11FlightPeriod);
02095     FacBomManager::linkWithParent (lSQInv, lSQ11FlightPeriod);
02096
02097     // Step 2: segment period level
02098     // Create a segment period for SIN-BKK:
02099
02100     SegmentPeriodKey lSegmentPeriodKey (lSIN, lBKK);
02101
02102     SegmentPeriod& lSINBKKSegmentPeriod =
02103         FacBom<SegmentPeriod>::instance().create (lSegmentPeriodKey);
02104     FacBomManager::addToListAndMap (lSQ11FlightPeriod, lSINBKKSegmentPeriod);
02105     FacBomManager::linkWithParent (lSQ11FlightPeriod, lSINBKKSegmentPeriod);
02106
02107     lSINBKKSegmentPeriod.setBoardingTime (l0820);
02108     lSINBKKSegmentPeriod.setOffTime (l1100);
02109     lSINBKKSegmentPeriod.setElapsedTime (l0340);
02110     ClassList_String_T lYM ("YM");
02111     lSINBKKSegmentPeriod.addCabinBookingClassList (lY, lYM);
02112
02113     // CX:
02114     // Step 1: flight period level
02115     // Create a flight period for CX12:
02116     lFlightNumber = FlightNumber_T (12);
02117
02118     lFlightPeriodKey = FlightPeriodKey(lFlightNumber, lPeriodStruct);

```

```

02119
02120     FlightPeriod& lCX12FlightPeriod =
02121         FacBom<FlightPeriod>::instance().create (lFlightPeriodKey);
02122     FacBomManager::addToListAndMap (lCXInv, lCX12FlightPeriod);
02123     FacBomManager::linkWithParent (lCXInv, lCX12FlightPeriod);
02124
02125     // Step 2: segment period level
02126     // Create a segment period for BKK-HKG:
02127
02128     lSegmentPeriodKey = SegmentPeriodKey (lBKK, lHKG);
02129
02130     SegmentPeriod& lBKKHKGSegmentPeriod =
02131         FacBom<SegmentPeriod>::instance().create (lSegmentPeriodKey);
02132     FacBomManager::addToListAndMap (lCX12FlightPeriod, lBKKHKGSegmentPeriod);
02133     FacBomManager::linkWithParent (lCX12FlightPeriod, lBKKHKGSegmentPeriod);
02134
02135     lBKKHKGSegmentPeriod.setBoardingTime (l1200);
02136     lBKKHKGSegmentPeriod.setOffTime (l1540);
02137     lBKKHKGSegmentPeriod.setElapsedTime (l0240);
02138     lBKKHKGSegmentPeriod.addCabinBookingClassList (lY,lYM);
02139
02140 }
02141
02142 // ////////////////////////////////////////
02143 void CmdBomManager::buildPartnershipsSamplePricing (BomRoot& ioBomRoot) {
02144
02145     /*=====*/
02146     // First airport pair SIN-BKK.
02147     // Set the airport-pair primary key.
02148     AirportPairKey lAirportPairKey ("SIN", "BKK");
02149
02150     // Create the AirportPairKey object and link it to the ioBomRoot object.
02151     AirportPair& lSINBKKAirportPair =
02152         FacBom<AirportPair>::instance().create (lAirportPairKey);
02153     FacBomManager::addToListAndMap (ioBomRoot, lSINBKKAirportPair);
02154     FacBomManager::linkWithParent (ioBomRoot, lSINBKKAirportPair);
02155
02156     // Set the fare date-period primary key.
02157     const Date_T lDateRangeStart (2010, boost::gregorian::Jan, 15);
02158     const Date_T lDateRangeEnd (2010, boost::gregorian::Dec, 31);
02159     const DatePeriod_T lDateRange (lDateRangeStart, lDateRangeEnd);
02160     const DatePeriodKey lDatePeriodKey (lDateRange);
02161
02162     // Create the DatePeriodKey object and link it to the PosChannel object.
02163     DatePeriod& lSINBKKDatePeriod =
02164         FacBom<DatePeriod>::instance().create (lDatePeriodKey);
02165     FacBomManager::addToListAndMap (lSINBKKAirportPair, lSINBKKDatePeriod);
02166     FacBomManager::linkWithParent (lSINBKKAirportPair, lSINBKKDatePeriod);
02167
02168     // Set the point-of-sale-channel primary key.
02169     PosChannelKey lPosChannelKey ("SIN","IN");
02170
02171     // Create the PositionKey object and link it to the AirportPair object.
02172     PosChannel& lSINPosChannel =
02173         FacBom<PosChannel>::instance().create (lPosChannelKey);
02174     FacBomManager::addToListAndMap (lSINBKKDatePeriod, lSINPosChannel);
02175     FacBomManager::linkWithParent (lSINBKKDatePeriod, lSINPosChannel);
02176
02177     // Set the fare time-period primary key.
02178     const Time_T lTimeRangeStart (0, 0, 0);
02179
02180

```

```

02181     const Time_T lTimeRangeEnd (23, 0, 0);
02182     const TimePeriodKey lFareTimePeriodKey (lTimeRangeStart,
02183                                             lTimeRangeEnd);
02184
02185     // Create the TimePeriodKey and link it to the DatePeriod object.
02186     TimePeriod& lSINBKKFareTimePeriod =
02187         FacBom<TimePeriod>::instance().create (lFareTimePeriodKey);
02188     FacBomManager::addToListAndMap (lSINPosChannel, lSINBKKFareTimePeriod);
02189     FacBomManager::linkWithParent (lSINPosChannel, lSINBKKFareTimePeriod);
02190
02191     // Generate the FareRule
02192     const FareFeaturesKey lFareFeaturesKey (TRIP_TYPE_ONE_WAY,
02193                                             NO_ADVANCE_PURCHASE,
02194                                             SATURDAY_STAY,
02195                                             CHANGE_FEES,
02196                                             NON_REFUNDABLE,
02197                                             NO_STAY_DURATION);
02198
02199     // Create the FareFeaturesKey and link it to the TimePeriod object.
02200     FareFeatures& lSINBKKFareFeatures =
02201         FacBom<FareFeatures>::instance().create (lFareFeaturesKey);
02202     FacBomManager::addToListAndMap (lSINBKKFareTimePeriod, lSINBKKFareFeatures);
02203     FacBomManager::linkWithParent (lSINBKKFareTimePeriod, lSINBKKFareFeatures);
02204
02205     // Generate Segment Features and link them to their FareRule.
02206     AirlineCodeList_T lSQAirlineCodeList;
02207     lSQAirlineCodeList.push_back ("SQ");
02208
02209     ClassList_StringList_T lYClassCodeList;
02210     lYClassCodeList.push_back ("Y");
02211     const AirlineClassListKey lSQAirlineYClassListKey (lSQAirlineCodeList,
02212                                                         lYClassCodeList);
02213
02214     ClassList_StringList_T lMClassCodeList;
02215     lMClassCodeList.push_back ("M");
02216     const AirlineClassListKey lSQAirlineMClassListKey (lSQAirlineCodeList,
02217                                                         lMClassCodeList);
02218
02219     // Create the AirlineClassListKey and link it to the FareFeatures object.
02220     AirlineClassList& lSQAirlineYClassList =
02221         FacBom<AirlineClassList>::instance().create (lSQAirlineYClassListKey);
02222     lSQAirlineYClassList.setFare(700);
02223     FacBomManager::addToListAndMap (lSINBKKFareFeatures, lSQAirlineYClassList);
02224     FacBomManager::linkWithParent (lSINBKKFareFeatures, lSQAirlineYClassList);
02225
02226     AirlineClassList& lSQAirlineMClassList =
02227         FacBom<AirlineClassList>::instance().create (lSQAirlineMClassListKey);
02228     lSQAirlineMClassList.setFare(500);
02229     FacBomManager::addToListAndMap (lSINBKKFareFeatures, lSQAirlineMClassList);
02230     FacBomManager::linkWithParent (lSINBKKFareFeatures, lSQAirlineMClassList);
02231
02232     /*=====
=====*/
02233     // Second airport pair BKK-HKG.
02234     // Set the airport-pair primary key.
02235     lAirportPairKey = AirportPairKey ("BKK", "HKG");
02236
02237     // Create the AirportPairKey object and link it to the ioBomRoot object.
02238     AirportPair& lBKKHKGAirportPair =
02239         FacBom<AirportPair>::instance().create (lAirportPairKey);

```

```

02240     FacBomManager::addToListAndMap (ioBomRoot, lBKKHKGAirportPair);
02241     FacBomManager::linkWithParent (ioBomRoot, lBKKHKGAirportPair);
02242
02243     // Set the fare date-period primary key.
02244     // Use the same as previously.
02245
02246     // Create the DatePeriodKey object and link it to the PosChannel object.
02247     DatePeriod& lBKKHKGDatePeriod =
02248         FacBom<DatePeriod>::instance().create (lDatePeriodKey);
02249     FacBomManager::addToListAndMap (lBKKHKGAirportPair, lBKKHKGDatePeriod);
02250     FacBomManager::linkWithParent (lBKKHKGAirportPair, lBKKHKGDatePeriod);
02251
02252     // Set the point-of-sale-channel primary key.
02253     lPosChannelKey = PosChannelKey("BKK","IN");
02254
02255     // Create the PositionKey object and link it to the AirportPair object.
02256     PosChannel& lBKKPosChannel =
02257         FacBom<PosChannel>::instance().create (lPosChannelKey);
02258     FacBomManager::addToListAndMap (lBKKHKGDatePeriod, lBKKPosChannel);
02259     FacBomManager::linkWithParent (lBKKHKGDatePeriod, lBKKPosChannel);
02260
02261     // Set the fare time-period primary key.
02262     // Use the same as previously.
02263
02264     // Create the TimePeriodKey and link it to the DatePeriod object.
02265     TimePeriod& lBKKHKGTimePeriod =
02266         FacBom<TimePeriod>::instance().create (lFareTimePeriodKey);
02267     FacBomManager::addToListAndMap (lBKKPosChannel, lBKKHKGTimePeriod);
02268     FacBomManager::linkWithParent (lBKKPosChannel, lBKKHKGTimePeriod);
02269
02270     // Generate the FareRule
02271     // Use the same key as previously.
02272
02273     // Create the FareFeaturesKey and link it to the TimePeriod object.
02274     FareFeatures& lBKKHKGTimePeriod =
02275         FacBom<FareFeatures>::instance().create (lFareFeaturesKey);
02276     FacBomManager::addToListAndMap (lBKKHKGTimePeriod, lBKKHKGTimePeriod);
02277     FacBomManager::linkWithParent (lBKKHKGTimePeriod, lBKKHKGTimePeriod);
02278
02279     // Generate Segment Features and link them to their FareRule.
02280     AirlineCodeList_T lCXAirlineCodeList;
02281     lCXAirlineCodeList.push_back ("CX");
02282
02283     const AirlineClassListKey lCXAirlineYClassListKey (lCXAirlineCodeList,
02284                                                         lYClassCodeList);
02285
02286     const AirlineClassListKey lCXAirlineMClassListKey (lCXAirlineCodeList,
02287                                                         lMClassCodeList);
02288
02289     // Create the AirlineClassListKey and link it to the FareFeatures object.
02290     AirlineClassList& lCXAirlineYClassList =
02291         FacBom<AirlineClassList>::instance().create (lCXAirlineYClassListKey);
02292     lCXAirlineYClassList.setFare(700);
02293     FacBomManager::addToListAndMap (lBKKHKGTimePeriod, lCXAirlineYClassList);
02294     FacBomManager::linkWithParent (lBKKHKGTimePeriod, lCXAirlineYClassList);
02295
02296     AirlineClassList& lCXAirlineMClassList =
02297         FacBom<AirlineClassList>::instance().create (lCXAirlineMClassListKey);
02298     lCXAirlineMClassList.setFare(500);
02299     FacBomManager::addToListAndMap (lBKKHKGTimePeriod, lCXAirlineMClassList);

```

```

02300     FacBomManager::linkWithParent (lBKKHKGFareFeatures, lCXAirlineMClassList);
02301
02302     /*=====*/
02303     // Third airport pair SIN-HKG.
02304     // Set the airport-pair primary key.
02305     lAirportPairKey = AirportPairKey ("SIN", "HKG");
02306
02307     // Create the AirportPairKey object and link it to the ioBomRoot object.
02308     AirportPair& lSINHKGAirportPair =
02309         FacBom<AirportPair>::instance().create (lAirportPairKey);
02310     FacBomManager::addToListAndMap (ioBomRoot, lSINHKGAirportPair);
02311     FacBomManager::linkWithParent (ioBomRoot, lSINHKGAirportPair);
02312
02313     // Set the fare date-period primary key.
02314     // Use the same as previously.
02315
02316     // Create the DatePeriodKey object and link it to the PosChannel object.
02317     DatePeriod& lSINHKGDatePeriod =
02318         FacBom<DatePeriod>::instance().create (lDatePeriodKey);
02319     FacBomManager::addToListAndMap (lSINHKGAirportPair, lSINHKGDatePeriod);
02320     FacBomManager::linkWithParent (lSINHKGAirportPair, lSINHKGDatePeriod);
02321
02322     // Set the point-of-sale-channel primary key.
02323     lPosChannelKey = PosChannelKey ("SIN", "IN");
02324
02325     // Create the PositionKey object and link it to the AirportPair object.
02326     PosChannel& lOnDSINPosChannel =
02327         FacBom<PosChannel>::instance().create (lPosChannelKey);
02328     FacBomManager::addToListAndMap (lSINHKGDatePeriod, lOnDSINPosChannel);
02329     FacBomManager::linkWithParent (lSINHKGDatePeriod, lOnDSINPosChannel);
02330
02331     // Set the fare time-period primary key.
02332     // Use the same as previously.
02333
02334     // Create the TimePeriodKey and link it to the DatePeriod object.
02335     TimePeriod& lSINHKGTimePeriod =
02336         FacBom<TimePeriod>::instance().create (lFareTimePeriodKey);
02337     FacBomManager::addToListAndMap (lOnDSINPosChannel, lSINHKGTimePeriod);
02338     FacBomManager::linkWithParent (lOnDSINPosChannel, lSINHKGTimePeriod);
02339
02340     // Generate the FareRule
02341     // Use the same key as previously.
02342
02343     // Create the FareFeaturesKey and link it to the TimePeriod object.
02344     FareFeatures& lSINHKGFAreFeatures =
02345         FacBom<FareFeatures>::instance().create (lFareFeaturesKey);
02346     FacBomManager::addToListAndMap (lSINHKGTimePeriod, lSINHKGFAreFeatures);
02347     FacBomManager::linkWithParent (lSINHKGTimePeriod, lSINHKGFAreFeatures);
02348
02349     // Generate Segment Features and link them to their FareRule.
02350     AirlineCodeList_T lSQ_CXAirlineCodeList;
02351     lSQ_CXAirlineCodeList.push_back ("SQ");
02352     lSQ_CXAirlineCodeList.push_back ("CX");
02353
02354     ClassList_StringList_T lY_YClassCodeList;
02355     lY_YClassCodeList.push_back ("Y");
02356     lY_YClassCodeList.push_back ("Y");
02357     const AirlineClassListKey lSQ_CXAirlineYClassListKey (lSQ_CXAirlineCodeList,
02358                                                             lY_YClassCodeList);

```

```

02359
02360     ClassList_StringList_T lM_MClassCodeList;
02361     lM_MClassCodeList.push_back ("M");
02362     lM_MClassCodeList.push_back ("M");
02363     const AirlineClassListKey lSQ_CXAirlineMClassListKey (lSQ_CXAirlineCodeList,
02364                                                         lM_MClassCodeList);
02365
02366     // Create the AirlineClassListKey and link it to the FareFeatures object.
02367     AirlineClassList& lSQ_CXAirlineYClassList =
02368         FacBom<AirlineClassList>::instance().create (lSQ_CXAirlineYClassListKey);
02369     lSQ_CXAirlineYClassList.setFare(1200);
02370     FacBomManager::addToListAndMap (lSINHKGFAreFeatures, lSQ_CXAirlineYClassList)
02371 ;
02372
02373     AirlineClassList& lSQ_CXAirlineMClassList =
02374         FacBom<AirlineClassList>::instance().create (lSQ_CXAirlineMClassListKey);
02375     lSQ_CXAirlineMClassList.setFare(850);
02376     FacBomManager::addToListAndMap (lSINHKGFAreFeatures, lSQ_CXAirlineMClassList)
02377 ;
02378     FacBomManager::linkWithParent (lSINHKGFAreFeatures, lSQ_CXAirlineMClassList);
02379
02380
02381
02382     /*=====*/
02383
02384     // Use the same airport pair, and date period for adding SQ SIN-BKK yields.
02385
02386     // Set the point-of-sale-channel primary key.
02387     lPosChannelKey = PosChannelKey(DEFAULT_POS, DEFAULT_CHANNEL);
02388
02389     // Create the PositionKey object and link it to the AirportPair object.
02390     PosChannel& lRAC_SINBKKPosChannel =
02391         FacBom<PosChannel>::instance().create (lPosChannelKey);
02392     FacBomManager::addToListAndMap (lSINBKKDatePeriod, lRAC_SINBKKPosChannel);
02393     FacBomManager::linkWithParent (lSINBKKDatePeriod, lRAC_SINBKKPosChannel);
02394
02395     // Set the yield time-period primary key.
02396     const TimePeriodKey lYieldTimePeriodKey (lTimeRangeStart,
02397                                             lTimeRangeEnd);
02398
02399     // Create the TimePeriodKey and link it to the DatePeriod object.
02400     TimePeriod& lSINBKKYieldTimePeriod =
02401         FacBom<TimePeriod>::instance().create (lYieldTimePeriodKey);
02402     FacBomManager::addToListAndMap (lRAC_SINBKKPosChannel, lSINBKKYieldTimePeriod)
02403 ;
02404     FacBomManager::linkWithParent (lRAC_SINBKKPosChannel, lSINBKKYieldTimePeriod)
02405 ;
02406
02407     // Generate the YieldRule
02408     const YieldFeaturesKey lYieldFeaturesKey (TRIP_TYPE_ONE_WAY,
02409                                             CABIN_Y);
02410
02411     // Create the YieldFeaturesKey and link it to the TimePeriod object.
02412     YieldFeatures& lSINBKKYieldFeatures =
02413         FacBom<YieldFeatures>::instance().create (lYieldFeaturesKey);
02414     FacBomManager::addToListAndMap (lSINBKKYieldTimePeriod, lSINBKKYieldFeatures)
02415 ;
02416     FacBomManager::linkWithParent (lSINBKKYieldTimePeriod, lSINBKKYieldFeatures);

```

```

02414
02415     // Generate Segment Features and link them to their YieldRule.
02416     // Use the same key as previously.
02417
02418     // Create the AirlineClassListKey and link it to the YieldFeatures object.
02419     AirlineClassList& lRAC_SQAirlineYClassList =
02420         FacBom<AirlineClassList>::instance().create (lSQAirlineYClassListKey);
02421     lRAC_SQAirlineYClassList.setYield(700);
02422     FacBomManager::addToListAndMap (lSINBKKYieldFeatures, lRAC_SQAirlineYClassList
02423 );
02424     FacBomManager::linkWithParent (lSINBKKYieldFeatures, lRAC_SQAirlineYClassList
02425 );
02426     AirlineClassList& lRAC_SQAirlineMClassList =
02427         FacBom<AirlineClassList>::instance().create (lSQAirlineMClassListKey);
02428     lRAC_SQAirlineMClassList.setYield(500);
02429     FacBomManager::addToListAndMap (lSINBKKYieldFeatures, lRAC_SQAirlineMClassList
02430 );
02431     FacBomManager::linkWithParent (lSINBKKYieldFeatures, lRAC_SQAirlineMClassList
02432 );
02433     /*=====
02434     // Use the same airport pair, and date period for adding CX BKK-HKG yields.
02435     // Set the point-of-sale-channel primary key.
02436     // Use the same as previously.
02437
02438     // Create the PositionKey object and link it to the AirportPair object.
02439     PosChannel& lRAC_BKKHKGPosChannel =
02440         FacBom<PosChannel>::instance().create (lPosChannelKey);
02441     FacBomManager::addToListAndMap (lBKKHKGDatePeriod, lRAC_BKKHKGPosChannel);
02442     FacBomManager::linkWithParent (lBKKHKGDatePeriod, lRAC_BKKHKGPosChannel);
02443
02444     // Set the yield time-period primary key.
02445     // Use the same as previously.
02446
02447     // Create the TimePeriodKey and link it to the DatePeriod object.
02448     TimePeriod& lBKKHKGYieldTimePeriod =
02449         FacBom<TimePeriod>::instance().create (lYieldTimePeriodKey);
02450     FacBomManager::addToListAndMap (lRAC_BKKHKGPosChannel, lBKKHKGYieldTimePeriod
02451 );
02452     FacBomManager::linkWithParent (lRAC_BKKHKGPosChannel, lBKKHKGYieldTimePeriod
02453 );
02454     // Generate the YieldRule
02455     // Use the same key as previously.
02456
02457     // Create the YieldFeaturesKey and link it to the TimePeriod object.
02458     YieldFeatures& lBKKHKGYieldFeatures =
02459         FacBom<YieldFeatures>::instance().create (lYieldFeaturesKey);
02460     FacBomManager::addToListAndMap (lBKKHKGYieldTimePeriod, lBKKHKGYieldFeatures)
02461 );
02462     FacBomManager::linkWithParent (lBKKHKGYieldTimePeriod, lBKKHKGYieldFeatures);
02463
02464     // Generate Segment Features and link them to their YieldRule.
02465     // Use the same key as previously.
02466
02467     // Create the AirlineClassListKey and link it to the YieldFeatures object.

```

```

02466     AirlineClassList& lRAC_CXAirlineYClassList =
02467         FacBom<AirlineClassList>::instance().create (lCXAirlineYClassListKey);
02468     lRAC_CXAirlineYClassList.setYield(700);
02469     FacBomManager::addToListAndMap (lBKKHKGyieldFeatures, lRAC_CXAirlineYClassList);
02470     FacBomManager::linkWithParent (lBKKHKGyieldFeatures, lRAC_CXAirlineYClassList);
02471
02472     AirlineClassList& lRAC_CXAirlineMClassList =
02473         FacBom<AirlineClassList>::instance().create (lCXAirlineMClassListKey);
02474     lRAC_CXAirlineMClassList.setYield(500);
02475     FacBomManager::addToListAndMap (lBKKHKGyieldFeatures, lRAC_CXAirlineMClassList);
02476     FacBomManager::linkWithParent (lBKKHKGyieldFeatures, lRAC_CXAirlineMClassList);
02477
02478     /*=====
02479     =====*/
02480     // Use the same airport pair, and date period for SQ-CX SIN-HKG
02481
02482     // Set the point-of-sale-channel primary key.
02483     // Use the same as previously.
02484
02485     // Create the PositionKey object and link it to the AirportPair object.
02486     PosChannel& lRAC_SINHKGChannel =
02487         FacBom<PosChannel>::instance().create (lPosChannelKey);
02488     FacBomManager::addToListAndMap (lSINHKGDatePeriod, lRAC_SINHKGChannel);
02489     FacBomManager::linkWithParent (lSINHKGDatePeriod, lRAC_SINHKGChannel);
02490
02491     // Set the yield time-period primary key.
02492     // Use the same as previously.
02493
02494     // Create the TimePeriodKey and link it to the DatePeriod object.
02495     TimePeriod& lSINHKGyieldTimePeriod =
02496         FacBom<TimePeriod>::instance().create (lYieldTimePeriodKey);
02497     FacBomManager::addToListAndMap (lRAC_SINHKGChannel, lSINHKGyieldTimePeriod);
02498     FacBomManager::linkWithParent (lRAC_SINHKGChannel, lSINHKGyieldTimePeriod);
02499
02500     // Generate the YieldRule
02501     // Use the same key as previously.
02502
02503     // Create the YieldFeaturesKey and link it to the TimePeriod object.
02504     YieldFeatures& lSINHKGyieldFeatures =
02505         FacBom<YieldFeatures>::instance().create (lYieldFeaturesKey);
02506     FacBomManager::addToListAndMap (lSINHKGyieldTimePeriod, lSINHKGyieldFeatures);
02507     FacBomManager::linkWithParent (lSINHKGyieldTimePeriod, lSINHKGyieldFeatures);
02508
02509     // Generate Segment Features and link them to their YieldRule.
02510     // Use the same key as previously
02511
02512     // Create the AirlineClassListKey and link it to the YieldFeatures object.
02513     AirlineClassList& lRAC_SQ_CXAirlineYClassList =
02514         FacBom<AirlineClassList>::instance().create (lSQ_CXAirlineYClassListKey);
02515     lRAC_SQ_CXAirlineYClassList.setYield(1200);
02516     FacBomManager::addToListAndMap (lSINHKGyieldFeatures, lRAC_SQ_CXAirlineYClassList);
02517     FacBomManager::linkWithParent (lSINHKGyieldFeatures, lRAC_SQ_CXAirlineYClassList);

```

```

02518
02519     AirlineClassList& lRAC_SQ_CXAirlineMClassList =
02520         FacBom<AirlineClassList>::instance().create (lSQ_CXAirlineMClassListKey);
02521     lRAC_SQ_CXAirlineMClassList.setYield(850);
02522     FacBomManager::addToListAndMap (lSINHKGyieldFeatures, lRAC_SQ_CXAirlineMClass
02523         List);
02524     FacBomManager::linkWithParent (lSINHKGyieldFeatures, lRAC_SQ_CXAirlineMClassL
02525         ist);
02526 }
02527 }
02528

```

35.469 stdair/command/CmdBomManager.hpp File Reference

```

#include <iosfwd>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/SampleType.hpp>
#include <stdair/bom/TravelSolutionTypes.hpp>
#include <stdair/command/CmdAbstract.hpp>

```

Classes

- class [stdair::CmdBomManager](#)

Namespaces

- namespace [stdair](#)

Handle on the StdAir library context.

35.470 CmdBomManager.hpp

```

00001 #ifndef __STDAIR_CMD_CMDBOMMANAGER_HPP
00002 #define __STDAIR_CMD_CMDBOMMANAGER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 // StdAir
00010 #include <stdair/stdair_inventory_types.hpp>
00011 #include <stdair/basic/SampleType.hpp>
00012 #include <stdair/bom/TravelSolutionTypes.hpp>
00013 #include <stdair/command/CmdAbstract.hpp>
00014
00015 namespace stdair {
00016
00017     class BomRoot;

```

```

00019     struct BookingRequestStruct;
00020
00025     class CmdBomManager : public CmdAbstract {
00026     //
00027     friend class STDAIR_Service;
00028     private:
00029
00030     // //////////////// BOM initialisation support methods ////////////////
00043     static void buildSampleBom (BomRoot&);
00044
00056     static void buildSampleInventorySchedule (BomRoot&);
00057
00071     static void buildCompleteDummyInventory (BomRoot&);
00072
00090     static void buildDummyInventory (BomRoot&, const CabinCapacity_T&);
00091
00099     static void buildSamplePricing (BomRoot&);
00100
00115     static void buildSampleTravelSolutionForPricing (TravelSolutionList_T&);
00116
00134     static void buildSampleTravelSolutions (TravelSolutionList_T&);
00135
00152     static BookingRequestStruct buildSampleBookingRequest ();
00153
00170     static BookingRequestStruct buildSampleBookingRequestForCRS ();
00171
00182     static void buildPartnershipsSampleInventoryAndRM (BomRoot&);
00183
00191     static void buildPartnershipsSamplePricing (BomRoot&);
00192
00193     };
00194 }
00195 #endif // __STDAIR_CMD_CMDBOMMANAGER_HPP

```

35.471 stdair/command/CmdBomSerialiser.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/list.hpp>
#include <boost/serialization/map.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/SegmentDate.hpp>

```

```
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/Bucket.hpp>
#include <stdair/factory/FacBomManager.hpp>
#include <stdair/factory/FacBom.hpp>
#include <stdair/command/CmdBomSerialiser.hpp>
#include <stdair/service/Logger.hpp>
```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

Functions

- template<class Archive , class BOM_OBJECT1 , class BOM_OBJECT2 >
void [stdair::serialiseHelper](#) (BOM_OBJECT1 &ioObject1, Archive &ioArchive, const unsigned int iFileVersion)
- template void [stdair::BomRoot::serialize](#)< [ba::text_oarchive](#) > (ba::text_oarchive &, unsigned int)
- template void [stdair::BomRoot::serialize](#)< [ba::text_iarchive](#) > (ba::text_iarchive &, unsigned int)
- template void [stdair::Inventory::serialize](#)< [ba::text_oarchive](#) > (ba::text_oarchive &, unsigned int)
- template void [stdair::Inventory::serialize](#)< [ba::text_iarchive](#) > (ba::text_iarchive &, unsigned int)
- template void [stdair::FlightDate::serialize](#)< [ba::text_oarchive](#) > (ba::text_oarchive &, unsigned int)
- template void [stdair::FlightDate::serialize](#)< [ba::text_iarchive](#) > (ba::text_iarchive &, unsigned int)
- template void [stdair::SegmentDate::serialize](#)< [ba::text_oarchive](#) > (ba::text_oarchive &, unsigned int)
- template void [stdair::SegmentDate::serialize](#)< [ba::text_iarchive](#) > (ba::text_iarchive &, unsigned int)
- template void [stdair::SegmentCabin::serialize](#)< [ba::text_oarchive](#) > (ba::text_oarchive &, unsigned int)
- template void [stdair::SegmentCabin::serialize](#)< [ba::text_iarchive](#) > (ba::text_iarchive &, unsigned int)

35.472 CmdBomSerialiser.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/list.hpp>
00011 #include <boost/serialization/map.hpp>
00012 #include <boost/serialization/access.hpp>
00013 // StdAir
00014 #include <stdair/basic/BasConst_General.hpp>
00015 #include <stdair/basic/BasConst_Inventory.hpp>
00016 #include <stdair/bom/BomRoot.hpp>
00017 #include <stdair/bom/Inventory.hpp>
00018 #include <stdair/bom/FlightDate.hpp>
00019 #include <stdair/bom/SegmentDate.hpp>
00020 #include <stdair/bom/SegmentCabin.hpp>
00021 #include <stdair/bom/FareFamily.hpp>
00022 #include <stdair/bom/LegDate.hpp>
00023 #include <stdair/bom/LegCabin.hpp>
00024 #include <stdair/bom/Bucket.hpp>
00025 #include <stdair/factory/FacBomManager.hpp>
00026 #include <stdair/factory/FacBom.hpp>
00027 #include <stdair/command/CmdBomSerialiser.hpp>
00028 #include <stdair/service/Logger.hpp>
00029
00030 namespace stdair {
00031
00032 // //////////////////////////////////////
00033 template <class Archive, class BOM_OBJECT1, class BOM_OBJECT2>
00034 void serialiseHelper (BOM_OBJECT1& ioObject1, Archive& ioArchive,
00035                     const unsigned int iFileVersion) {
00036
00050     BomHolder<BOM_OBJECT2>* lBomHolder_ptr =
00051         FacBomManager::getBomHolderPtr<BOM_OBJECT2> (ioObject1);
00052
00053     if (lBomHolder_ptr == NULL) {
00054         lBomHolder_ptr = &FacBomManager::addBomHolder<BOM_OBJECT2> (ioObject1);
00055     }
00056     assert (lBomHolder_ptr != NULL);
00057
00061     //ioArchive.register_type (static_cast<Inventory*> (NULL));
00062     ioArchive & lBomHolder_ptr->_bomList;
00063     ioArchive & lBomHolder_ptr->_bomMap;
00064
00071     typedef typename BomHolder<BOM_OBJECT2>::BomList_T BomList_T;
00072     const BomList_T& lBomList = lBomHolder_ptr->_bomList;
00073     for (typename BomList_T::const_iterator itObject = lBomList.begin();
00074          itObject != lBomList.end(); ++itObject) {
00075         BOM_OBJECT2* lObject2_ptr = *itObject;
00076         assert (lObject2_ptr != NULL);
00077
00078         if (lObject2_ptr->getParent() == NULL) {
00084             FacBomManager::linkWithParent (ioObject1, *lObject2_ptr);
00085         }
00086     }
00087

```

```

00096     typedef typename BomHolder<BOM_OBJECT2>::BomMap_T BomMap_T;
00097     const BomMap_T& lBomMap = lBomHolder_ptr->_bomMap;
00098     if (lBomList.empty() == true && lBomMap.empty() == false) {
00099
00100         for (typename BomMap_T::const_iterator itObject = lBomMap.begin();
00101              itObject != lBomMap.end(); ++itObject) {
00102             BOM_OBJECT2* lObject2_ptr = itObject->second;
00103             assert (lObject2_ptr != NULL);
00104
00105             if (lObject2_ptr->getParent() == NULL) {
00111                 FacBomManager::linkWithParent (ioObject1, *lObject2_ptr);
00112             }
00113         }
00114     }
00115 }
00116
00117 // //////////////////////////////////////
00118 void BomRoot::serialisationImplementationExport() const {
00119     std::ostream oStr;
00120     boost::archive::text_oarchive oa (oStr);
00121     oa << *this;
00122 }
00123
00124 // //////////////////////////////////////
00125 void BomRoot::serialisationImplementationImport() {
00126     std::istream iStr;
00127     boost::archive::text_iarchive ia (iStr);
00128     ia >> *this;
00129 }
00130
00131 // //////////////////////////////////////
00132 template<class Archive>
00133 void BomRoot::serialize (Archive& ioArchive,
00134                          const unsigned int iFileVersion) {
00135     // Serialise the key (by default, equal to " -- ROOT -- ")
00136     ioArchive & _key;
00137
00138     // Serialise the children of the BomRoot object, i.e., the
00139     // Inventory children
00140     stdair::serialiseHelper<Archive, BomRoot, Inventory> (*this, ioArchive,
00141                                                           iFileVersion);
00142 }
00143
00144 // //////////////////////////////////////
00145 void Inventory::serialisationImplementationExport() const {
00146     std::ostream oStr;
00147     boost::archive::text_oarchive oa (oStr);
00148     oa << *this;
00149 }
00150
00151 // //////////////////////////////////////
00152 void Inventory::serialisationImplementationImport() {
00153     std::istream iStr;
00154     boost::archive::text_iarchive ia (iStr);
00155     ia >> *this;
00156 }
00157
00158 // //////////////////////////////////////
00159 template<class Archive>
00160 void Inventory::serialize (Archive& ioArchive,
00161                           const unsigned int iFileVersion) {
00162     // Serialise the key (airline code)

```

```

00163     ioArchive & _key;
00164
00165     // Serialise the children of the Inventory object, i.e., the
00166     // FlightDate children
00167     stdair::serialiseHelper<Archive, Inventory, FlightDate> (*this, ioArchive,
00168                                                             iFileVersion);
00169 }
00170
00171 // //////////////////////////////////////
00172 void FlightDate::serialisationImplementationExport() const {
00173     std::ostringstream oStr;
00174     boost::archive::text_oarchive oa (oStr);
00175     oa << *this;
00176 }
00177
00178 // //////////////////////////////////////
00179 void FlightDate::serialisationImplementationImport() {
00180     std::istringstream iStr;
00181     boost::archive::text_iarchive ia (iStr);
00182     ia >> *this;
00183 }
00184
00185 // //////////////////////////////////////
00186 template<class Archive>
00187 void FlightDate::serialize (Archive& ioArchive,
00188                             const unsigned int iFileVersion) {
00189     ioArchive & _key;
00190 }
00191
00192 // //////////////////////////////////////
00193 void SegmentDate::serialisationImplementationExport() const {
00194     std::ostringstream oStr;
00195     boost::archive::text_oarchive oa (oStr);
00196     oa << *this;
00197 }
00198
00199 // //////////////////////////////////////
00200 void SegmentDate::serialisationImplementationImport() {
00201     std::istringstream iStr;
00202     boost::archive::text_iarchive ia (iStr);
00203     ia >> *this;
00204 }
00205
00206 // //////////////////////////////////////
00207 template<class Archive>
00208 void SegmentDate::serialize (Archive& ioArchive,
00209                             const unsigned int iFileVersion) {
00210     ioArchive & _key;
00211 }
00212
00213 // //////////////////////////////////////
00214 void SegmentCabin::serialisationImplementationExport() const {
00215     std::ostringstream oStr;
00216     boost::archive::text_oarchive oa (oStr);
00217     oa << *this;
00218 }
00219
00220 // //////////////////////////////////////
00221 void SegmentCabin::serialisationImplementationImport() {
00222     std::istringstream iStr;
00223     boost::archive::text_iarchive ia (iStr);
00224     ia >> *this;

```

```

00225     }
00226
00227     // //////////////////////////////////////
00228     template<class Archive>
00229     void SegmentCabin::serialize (Archive& ioArchive,
00230                                   const unsigned int iFileVersion) {
00231         ioArchive & _key;
00232     }
00233
00234     // //////////////////////////////////////
00235     // Explicit template instantiations
00236     namespace ba = boost::archive;
00237     template void BomRoot::serialize<ba::text_oarchive> (ba::text_oarchive&,
00238                                                         unsigned int);
00239     template void BomRoot::serialize<ba::text_iarchive> (ba::text_iarchive&,
00240                                                         unsigned int);
00241     template void Inventory::serialize<ba::text_oarchive> (ba::text_oarchive&,
00242                                                         unsigned int);
00243     template void Inventory::serialize<ba::text_iarchive> (ba::text_iarchive&,
00244                                                         unsigned int);
00245     template void FlightDate::serialize<ba::text_oarchive> (ba::text_oarchive&,
00246                                                         unsigned int);
00247     template void FlightDate::serialize<ba::text_iarchive> (ba::text_iarchive&,
00248                                                         unsigned int);
00249     template void SegmentDate::serialize<ba::text_oarchive> (ba::text_oarchive&,
00250                                                         unsigned int);
00251     template void SegmentDate::serialize<ba::text_iarchive> (ba::text_iarchive&,
00252                                                         unsigned int);
00253     template void SegmentCabin::serialize<ba::text_oarchive> (ba::text_oarchive&,
00254                                                         unsigned int);
00255     template void SegmentCabin::serialize<ba::text_iarchive> (ba::text_iarchive&,
00256                                                         unsigned int);
00257     // //////////////////////////////////////
00258
00259 }
```

35.473 stdair/command/CmdBomSerialiser.hpp File Reference

```

#include <iosfwd>

#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/TravelSolutionTypes.hpp>
#include <stdair/command/CmdAbstract.hpp>
```

Classes

- class [stdair::CmdBomSerialiser](#)

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.474 CmdBomSerialiser.hpp

```

00001 #ifndef __STDAIR_CMD_CMDBOMSERIALISER_HPP
00002 #define __STDAIR_CMD_CMDBOMSERIALISER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 // StdAir
00010 #include <stdair/stdair_inventory_types.hpp>
00011 #include <stdair/bom/TravelSolutionTypes.hpp>
00012 #include <stdair/command/CmdAbstract.hpp>
00013
00014 namespace stdair {
00015
00016     class BomRoot;
00017     struct BookingRequestStruct;
00018
00019
00020
00021     class CmdBomSerialiser : public CmdAbstract {
00022     public:
00023
00024         // ////////////////////////////////// BOM serialisation support methods //////////////////////////////////
00025         //
00026     };
00027
00028 }
00029 #endif // __STDAIR_CMD_CMDBOMSERIALISER_HPP

```

35.475 stdair/command/DBManagerForAirlines.cpp File Reference

```

#include <cassert>
#include <soci.h>
#include <mysql/soci-mysql.h>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/bom/AirlineStruct.hpp>
#include <stdair/dbadaptor/DbAirline.hpp>
#include <stdair/command/DBManagerForAirlines.hpp>
#include <stdair/service/Logger.hpp>

```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

35.476 DBManagerForAirlines.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // SOCI
00007 #if defined(SOCI_HEADERS_BURIED)
00008 #include <soci/core/soci.h>
00009 #include <soci/backends/mysql/soci-mysql.h>
00010 #else // SOCI_HEADERS_BURIED
00011 #include <soci.h>
00012 #include <mysql/soci-mysql.h>
00013 #endif // SOCI_HEADERS_BURIED
00014 // StdAir
00015 #include <stdair/stdair_basic_types.hpp>
00016 #include <stdair/stdair_exceptions.hpp>
00017 #include <stdair/bom/AirlineStruct.hpp>
00018 #include <stdair/dbadaptor/Dbairline.hpp>
00019 #include <stdair/command/DBManagerForAirlines.hpp>
00020 #include <stdair/service/Logger.hpp>
00021
00022 namespace stdair {
00023
00024 // //////////////////////////////////////
00025 void DBManagerForAirlines::
00026     prepareSelectStatement (DBSession_T& ioSociSession,
00027                             DBRequestStatement_T& ioSelectStatement,
00028                             AirlineStruct& ioAirline) {
00029
00030     try {
00031
00032         // Instanciate a SQL statement (no request is performed at that stage)
00033         ioSelectStatement = (ioSociSession.prepare
00034                             << "select iata_code, name "
00035                             << "from airlines ", soci::into (ioAirline));
00036
00037         // Execute the SQL query
00038         ioSelectStatement.execute();
00039
00040     } catch (std::exception const& lException) {
00041         throw SQLDatabaseException (lException.what());
00042     }
00043 }
00044
00045 // //////////////////////////////////////
00046 void DBManagerForAirlines::
00047     prepareSelectOnAirlineCodeStatement (DBSession_T& ioSociSession,
00048                                           DBRequestStatement_T& ioSelectStatement,
00049                                           const AirlineCode_T& iAirlineCode,
00050                                           AirlineStruct& ioAirline) {
00051
00052     try {
00053
00054         // Instanciate a SQL statement (no request is performed at that stage)
00055         ioSelectStatement = (ioSociSession.prepare
00056                             << "select iata_code, name "
00057                             << "from airlines "
00058                             << "where iata_code = :airline_code ",
00059                             soci::into (ioAirline), soci::use (iAirlineCode));
00060
00061     }
00062 }

```

```

00072         // Execute the SQL query
00073         ioSelectStatement.execute();
00074
00075     } catch (std::exception const& lException) {
00076         throw SQLDatabaseException (lException.what());
00077     }
00078 }
00079
00080 // //////////////////////////////////////
00081 bool DBManagerForAirlines::
00082 iterateOnStatement (DBRequestStatement_T& ioStatement,
00083                     AirlineStruct& ioAirline) {
00084     bool hasStillData = false;
00085
00086     try {
00087         // Retrieve the next row of Airline object
00088         hasStillData = ioStatement.fetch();
00089
00090     } catch (std::exception const& lException) {
00091         throw SQLDatabaseException (lException.what());
00092     }
00093
00094     return hasStillData;
00095 }
00096
00097 // //////////////////////////////////////
00098 void DBManagerForAirlines::updateAirlineInDB (DBSession_T& ioSociSession,
00099                                               const AirlineStruct& iAirline) {
00100     try {
00101         // Begin a transaction on the database
00102         ioSociSession.begin();
00103
00104         // Retrieve the airline code
00105         const std::string& lAirlineCode = iAirline.getAirlineCode();
00106
00107         // Retrieve the airline name
00108         const std::string& lAirlineName = iAirline.getAirlineName();
00109
00110         // Instantiate a SQL statement (no request is performed at that stage)
00111         DBRequestStatement_T lUpdateStatement =
00112             (ioSociSession.prepare
00113              << "update airlines "
00114              << "set name = :name "
00115              << "where iata_code = :iata_code",
00116              soci::use (lAirlineName), soci::use (lAirlineCode));
00117
00118         // Execute the SQL query
00119         lUpdateStatement.execute (true);
00120
00121         // Commit the transaction on the database
00122         ioSociSession.commit();
00123
00124         // Debug
00125         // STDAIR_LOG_DEBUG ("[" << lAirlineCode << "]" " << iAirline);
00126
00127     } catch (std::exception const& lException) {
00128         throw SQLDatabaseException (lException.what());
00129     }
00130 }
00131
00132 // //////////////////////////////////////

```

```

00134 bool DBManagerForAirlines::retrieveAirline (DBSession_T& ioSociSession,
00135                                             const AirlineCode_T& iAirlineCode,
00136                                             AirlineStruct& ioAirline) {
00137     bool oHasRetrievedAirline = false;
00138
00139     try {
00140         // Prepare the SQL request corresponding to the select statement
00141         DBRequestStatement_T lSelectStatement (ioSociSession);
00142         prepareSelectOnAirlineCodeStatement (ioSociSession, lSelectStatement,
00143                                             iAirlineCode, ioAirline);
00144
00145         // const bool shouldDoReset = true;
00146         bool hasStillData = iterateOnStatement (lSelectStatement, ioAirline);
00147         if (hasStillData == true) {
00148             oHasRetrievedAirline = true;
00149         }
00150
00151         // Sanity check
00152         // const bool shouldNotDoReset = false;
00153         hasStillData = iterateOnStatement (lSelectStatement, ioAirline);
00154
00155         // Debug
00156         // STDAIR_LOG_DEBUG ("[" << iDocID << "]" " << ioAirline);
00157
00158     } catch (std::exception const& lException) {
00159         throw SQLDatabaseException (lException.what());
00160     }
00161
00162     return oHasRetrievedAirline;
00163 }
00164
00165 }

```

35.477 stdair/command/DBManagerForAirlines.hpp File Reference

```

#include <stdair/stdair_db.hpp>
#include <stdair/command/CmdAbstract.hpp>

```

Classes

- class [stdair::DBManagerForAirlines](#)

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.478 DBManagerForAirlines.hpp

```

00001 #ifndef __DSIM_CMD_DBMANAGERFORAIRLINES_HPP
00002 #define __DSIM_CMD_DBMANAGERFORAIRLINES_HPP
00003
00004 // //////////////////////////////////////

```

```

00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_db.hpp>
00009 #include <stdair/command/CmdAbstract.hpp>
00010
00011 namespace stdair {
00012
00013     // Forward declarations
00014     struct AirlineStruct;
00015
00016     class DBManagerForAirlines : public CmdAbstract {
00017     public:
00024         static void updateAirlineInDB (DBSession_T&, const AirlineStruct&);
00025
00032         static bool retrieveAirline (DBSession_T&, const AirlineCode_T&,
00033                                     AirlineStruct&);
00034
00035     public:
00041         static void prepareSelectStatement (DBSession_T&, DBRequestStatement_T&,
00042                                             AirlineStruct&);
00043
00048         static bool iterateOnStatement (DBRequestStatement_T&, AirlineStruct&);
00049
00050     private:
00057         static void prepareSelectOnAirlineCodeStatement (DBSession_T&,
00058                                                         DBRequestStatement_T&,
00059                                                         const AirlineCode_T&,
00060                                                         AirlineStruct&);
00061
00062     private:
00063         // ////////////////////////////////// Constructors and Destructors //////////////////////////////////
00064         DBManagerForAirlines () {}
00065         DBManagerForAirlines (const DBManagerForAirlines&) {}
00066         ~DBManagerForAirlines () {}
00071     };
00072
00073 }
00074 #endif // __DSIM_CMD_DBMANAGERFORAIRLINES_HPP

```

35.479 stdair/config/stdair-paths.hpp File Reference

Defines

- #define [PACKAGE](#) "stdair"
- #define [PACKAGE_NAME](#) "STDAIR"
- #define [PACKAGE_VERSION](#) "0.44.0"
- #define [PREFIXDIR](#) "/usr"
- #define [EXEC_PREFIX](#) "/usr"
- #define [BINDIR](#) "/usr/bin"
- #define [LIBDIR](#) "/usr/lib64"
- #define [LIBEXECDIR](#) "/usr/libexec"
- #define [SBINDIR](#) "/usr/sbin"
- #define [SYSCONFDIR](#) "/usr/etc"

- #define INCLUDEDIR "/usr/include"
- #define DATAROOTDIR "/usr/share"
- #define DATADIR "/usr/share"
- #define DOCDIR "/usr/share/doc/stdair-0.44.0"
- #define MANDIR "/usr/share/man"
- #define INFODIR "/usr/share/info"
- #define HTMLDIR "/usr/share/doc/stdair-0.44.0/html"
- #define PDFDIR "/usr/share/doc/stdair-0.44.0/html"
- #define STDAIR_SAMPLE_DIR "/usr/share/stdair/samples"

35.479.1 Define Documentation

35.479.1.1 #define PACKAGE "stdair"

Definition at line 4 of file [stdair-paths.hpp](#).

35.479.1.2 #define PACKAGE_NAME "STDAIR"

Definition at line 5 of file [stdair-paths.hpp](#).

35.479.1.3 #define PACKAGE_VERSION "0.44.0"

Definition at line 6 of file [stdair-paths.hpp](#).

35.479.1.4 #define PREFIXDIR "/usr"

Definition at line 7 of file [stdair-paths.hpp](#).

35.479.1.5 #define EXEC_PREFIX "/usr"

Definition at line 8 of file [stdair-paths.hpp](#).

35.479.1.6 #define BINDIR "/usr/bin"

Definition at line 9 of file [stdair-paths.hpp](#).

35.479.1.7 #define LIBDIR "/usr/lib64"

Definition at line 10 of file [stdair-paths.hpp](#).

35.479.1.8 #define LIBEXECDIR "/usr/libexec"

Definition at line 11 of file [stdair-paths.hpp](#).

35.479.1.9 #define SBINDIR "/usr/sbin"

Definition at line 12 of file [stdair-paths.hpp](#).

35.479.1.10 #define SYSCONFDIR "/usr/etc"

Definition at line 13 of file [stdair-paths.hpp](#).

35.479.1.11 `#define INCLUDEDIR "/usr/include"`

Definition at line 14 of file [stdair-paths.hpp](#).

35.479.1.12 `#define DATAROOTDIR "/usr/share"`

Definition at line 15 of file [stdair-paths.hpp](#).

35.479.1.13 `#define DATADIR "/usr/share"`

Definition at line 16 of file [stdair-paths.hpp](#).

35.479.1.14 `#define DOCDIR "/usr/share/doc/stdair-0.44.0"`

Definition at line 17 of file [stdair-paths.hpp](#).

35.479.1.15 `#define MANDIR "/usr/share/man"`

Definition at line 18 of file [stdair-paths.hpp](#).

35.479.1.16 `#define INFODIR "/usr/share/info"`

Definition at line 19 of file [stdair-paths.hpp](#).

35.479.1.17 `#define HTMLDIR "/usr/share/doc/stdair-0.44.0/html"`

Definition at line 20 of file [stdair-paths.hpp](#).

35.479.1.18 `#define PDFDIR "/usr/share/doc/stdair-0.44.0/html"`

Definition at line 21 of file [stdair-paths.hpp](#).

35.479.1.19 `#define STDAIR_SAMPLE_DIR "/usr/share/stdair/samples"`

Definition at line 22 of file [stdair-paths.hpp](#).

35.480 stdair-paths.hpp

```
00001 #ifndef __STDAIR_PATHS_HPP__
00002 #define __STDAIR_PATHS_HPP__
00003
00004 #define PACKAGE "stdair"
00005 #define PACKAGE_NAME "STDAIR"
00006 #define PACKAGE_VERSION "0.44.0"
00007 #define PREFIXDIR "/usr"
00008 #define EXEC_PREFIX "/usr"
00009 #define BINDIR "/usr/bin"
00010 #define LIBDIR "/usr/lib64"
00011 #define LIBEXECDIR "/usr/libexec"
00012 #define SBINDIR "/usr/sbin"
00013 #define SYSCONFDIR "/usr/etc"
00014 #define INCLUDEDIR "/usr/include"
00015 #define DATAROOTDIR "/usr/share"
00016 #define DATADIR "/usr/share"
```

```

00017 #define DOCDIR "/usr/share/doc/stdair-0.44.0"
00018 #define MANDIR "/usr/share/man"
00019 #define INFODIR "/usr/share/info"
00020 #define HTMLDIR "/usr/share/doc/stdair-0.44.0/html"
00021 #define PDFDIR "/usr/share/doc/stdair-0.44.0/html"
00022 #define STDAIR_SAMPLE_DIR "/usr/share/stdair/samples"
00023
00024 #endif // __STDAIR_PATHS_HPP__

```

35.481 stdair/dbadaptor/DbAbstract.cpp File Reference

```
#include <stdair/dbadaptor/DbAbstract.hpp>
```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.482 DbAbstract.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // StdAir
00005 #include <stdair/dbadaptor/DbAbstract.hpp>
00006
00007 namespace stdair {
00008
00009 }

```

35.483 stdair/dbadaptor/DbAbstract.hpp File Reference

```
#include <iosfwd>
```

Classes

- class [stdair::DbAbstract](#)

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

Functions

- template<class charT, class traits >
std::basic_ostream< charT, traits > & [operator<<](#) (std::basic_ostream< charT, traits > &ioOut, const [stdair::DbAbstract](#) &iDb)

- `template<class charT , class traits >`
`std::basic_istream< charT, traits > & operator>> (std::basic_istream< charT,`
`traits > &ioIn, stdair::DbAbstract &ioDb)`

35.483.1 Function Documentation

35.483.1.1 `template<class charT , class traits > std::basic_ostream<charT, traits>&`
`operator<< (std::basic_ostream< charT, traits > & ioOut, const`
`stdair::DbAbstract & iDb) \[inline\]`

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (p653) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line [41](#) of file [DbAbstract.hpp](#).

References [stdair::DbAbstract::toStream\(\)](#).

35.483.1.2 `template<class charT , class traits > std::basic_istream<charT, traits>&`
`operator>> (std::basic_istream< charT, traits > & ioIn, stdair::DbAbstract &`
`ioDb) \[inline\]`

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (pp655-657) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line [69](#) of file [DbAbstract.hpp](#).

References [stdair::DbAbstract::fromStream\(\)](#).

35.484 DbAbstract.hpp

```
00001 #ifndef __STDAIR_DBA_DBAABSTRACT_HPP
00002 #define __STDAIR_DBA_DBAABSTRACT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009
00010 namespace stdair {
00011
00012     class DbAbstract {
00013     public:
00014
00015         virtual ~DbAbstract() {}
00016
00017         virtual void toStream (std::ostream& ioOut) const {}
00018
00019         virtual void fromStream (std::istream& ioIn) {}
00020
00021     protected:
00022         DbAbstract() {}
00023     };
00024 }
```

```

00031 }
00032
00038 template <class charT, class traits>
00039 inline
00040 std::basic_ostream<charT, traits>&
00041 operator<< (std::basic_ostream<charT, traits>& ioOut,
00042           const stdair::DbairlineAbstract& iDbairline) {
00048     std::basic_ostringstream<charT,traits> ostr;
00049     ostr.copyfmt (ioOut);
00050     ostr.width (0);
00051
00052     // Fill string stream
00053     iDbairline.toStream (ostr);
00054
00055     // Print string stream
00056     ioOut << ostr.str();
00057
00058     return ioOut;
00059 }
00060
00066 template <class charT, class traits>
00067 inline
00068 std::basic_istream<charT, traits>&
00069 operator>> (std::basic_istream<charT, traits>& ioIn,
00070           stdair::DbairlineAbstract& ioDbairline) {
00071     // Fill Dbairline object with input stream
00072     ioDbairline.fromStream (ioIn);
00073     return ioIn;
00074 }
00075
00076 #endif // __STDAIR_DBAIRLINE_ABSTRACT_HPP

```

35.485 stdair/dbadaptor/Dbairline.cpp File Reference

```

#include <exception>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/AirlineStruct.hpp>
#include <stdair/dbadaptor/Dbairline.hpp>
#include <stdair/service/Logger.hpp>

```

Namespaces

- namespace `soci`
- namespace `stdair`

Handle on the StdAir library context.

35.486 Dbairline.cpp

```

00001 // //////////////////////////////////////
00002 // Import section

```

```

00003 // //////////////////////////////////////
00004 // STL
00005 #include <exception>
00006 #include <string>
00007 // Stdair
00008 #include <stdair/stdair_inventory_types.hpp>
00009 #include <stdair/bom/AirlineStruct.hpp>
00010 #include <stdair/dbadaptor/Dbairline.hpp>
00011 #include <stdair/service/Logger.hpp>
00012
00013 namespace soci {
00014
00015 // //////////////////////////////////////
00016 void type_conversion<stdair::AirlineStruct>::
00017 from_base (values const& iAirlineValues, indicator /* ind */,
00018           stdair::AirlineStruct& ioAirline) {
00019     /*
00020      iata_code, name
00021     */
00022     ioAirline.setAirlineCode (iAirlineValues.get<std::string> ("iata_code"));
00023     // The city code will be set to the default value (empty string)
00024     // when the column is null
00025     ioAirline.setAirlineName (iAirlineValues.get<std::string> ("name", ""));
00026 }
00027
00028 // //////////////////////////////////////
00029 void type_conversion<stdair::AirlineStruct>::
00030 to_base (const stdair::AirlineStruct& iAirline, values& ioAirlineValues,
00031         indicator& ioIndicator) {
00032     const indicator lNameIndicator =
00033         iAirline.getAirlineName().empty() ? i_null : i_ok;
00034     ioAirlineValues.set ("iata_code", iAirline.getAirlineCode());
00035     ioAirlineValues.set ("name", iAirline.getAirlineName(), lNameIndicator);
00036     ioIndicator = i_ok;
00037 }
00038
00039 }
00040
00041 namespace stdair {
00042
00043 }

```

35.487 stdair/dbadaptor/Dbairline.hpp File Reference

```
#include <soci/soci.h>
```

Classes

- struct [soci::type_conversion< stdair::AirlineStruct >](#)

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.
- namespace [soci](#)

35.488 Dbairline.hpp

```

00001 #ifndef __STDAIR_DBA_DBAAIRLINE_HPP
00002 #define __STDAIR_DBA_DBAAIRLINE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // SOCI
00008 #if defined(SOCI_HEADERS_BURIED)
00009 #include <soci/core/soci.h>
00010 #else // SOCI_HEADERS_BURIED
00011 #include <soci/soci.h>
00012 #endif // SOCI_HEADERS_BURIED
00013
00014 // Forward declarations
00015 namespace stdair {
00016     struct AirlineStruct;
00017 }
00018
00019 namespace soci {
00020
00024     template <>
00025     struct type_conversion<stdair::AirlineStruct> {
00026
00027         typedef values base_type;
00028
00030         static void from_base (values const& iAirlineValues,
00031                               indicator /* ind */,
00032                               stdair::AirlineStruct& ioAirline);
00033
00034
00036         static void to_base (const stdair::AirlineStruct& iAirline,
00037                              values& ioAirlineValues,
00038                              indicator& ioIndicator);
00039     };
00040 }
00041 #endif // __STDAIR_DBA_DBAAIRLINE_HPP

```

35.489 stdair/factory/FacAbstract.cpp File Reference

```

#include <cassert>

#include <stdair/bom/BomAbstract.hpp>

#include <stdair/factory/FacAbstract.hpp>

```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

35.490 FacAbstract.cpp

```

00001 // //////////////////////////////////////
00002 // Import section

```

```

00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // STDAIR
00007 #include <stdair/bom/BomAbstract.hpp>
00008 #include <stdair/factory/FacAbstract.hpp>
00009
00010 namespace stdair {
00011
00012 // //////////////////////////////////////
00013     FacAbstract::~FacAbstract() {
00014     }
00015
00016 }

```

35.491 stdair/factory/FacAbstract.hpp File Reference

Classes

- class [stdair::FacAbstract](#)

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.492 FacAbstract.hpp

```

00001 #ifndef __STDAIR_FAC_FACABSTRACT_HPP
00002 #define __STDAIR_FAC_FACABSTRACT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007
00008 namespace stdair {
00009     class FacAbstract {
00010     public:
00011         virtual ~FacAbstract();
00012
00013     protected:
00014         FacAbstract() {}
00015     };
00016 }
00017 #endif // __STDAIR_FAC_FACABSTRACT_HPP

```

35.493 stdair/factory/FacBom.hpp File Reference

```

#include <cassert>
#include <string>
#include <list>

```

```
#include <stdair/factory/FacAbstract.hpp>
#include <stdair/service/FacSupervisor.hpp>
#include <stdair/service/Logger.hpp>
```

Classes

- class `stdair::FacBom< BOM >`

Base class for Factory layer.

Namespaces

- namespace `stdair`

Handle on the StdAir library context.

35.494 FacBom.hpp

```
00001 #ifndef __STDAIR_FAC_FACBOM_HPP
00002 #define __STDAIR_FAC_FACBOM_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <cassert>
00009 #include <string>
00010 #include <list>
00011 // StdAir
00012 #include <stdair/factory/FacAbstract.hpp>
00013 #include <stdair/service/FacSupervisor.hpp>
00014 #include <stdair/service/Logger.hpp>
00015
00016 namespace stdair {
00017
00021     template <typename BOM>
00022     class FacBom : public FacAbstract {
00023
00025         typedef std::list<BOM*> BomPool_T;
00026         typedef typename BOM::Key_T Key_T;
00027
00028
00029     public:
00030         // ////////////////////////////////// Business methods //////////////////////////////////
00037         static FacBom& instance();
00038
00042         BOM& create ();
00043         BOM& create (const Key_T&);
00044
00045     protected:
00049         FacBom() {}
00050
00051     public:
00055         ~FacBom() {
00056             clean();
00057         }
00058     };
00059 }
```

```

00058
00062     void clean();
00063
00064
00065 private:
00066     // ////////////////////////////////// Attributes //////////////////////////////////
00070     static FacBom* _instance;
00071
00075     BomPool_T _pool;
00076 };
00077
00078
00079 // //////////////////////////////////////////////////////////////////////
00080 template <typename BOM> FacBom<BOM>* FacBom<BOM>::_instance = NULL;
00081
00082 // //////////////////////////////////////////////////////////////////////
00083 template <typename BOM> FacBom<BOM>& FacBom<BOM>::instance () {
00084     if (_instance == NULL) {
00085         _instance = new FacBom ();
00086         assert (_instance != NULL);
00087
00088         FacSupervisor::instance().registerBomFactory (_instance);
00089     }
00090     return *_instance;
00091 }
00092
00093 // //////////////////////////////////////////////////////////////////////
00094 template <typename BOM> void FacBom<BOM>::clean () {
00095     // Destroy all the objects
00096     for (typename BomPool_T::iterator itBom = _pool.begin();
00097          itBom != _pool.end(); ++itBom) {
00098         BOM* currentBom_ptr = *itBom;
00099         assert (currentBom_ptr != NULL);
00100         delete currentBom_ptr; currentBom_ptr = NULL;
00101     }
00102
00103     // Empty the pool.
00104     _pool.clear();
00105
00106     // Reset the static instance.
00107     _instance = NULL;
00108 }
00109
00110 // //////////////////////////////////////////////////////////////////////
00111 template <typename BOM> BOM& FacBom<BOM>::create () {
00112     Key_T lKey;
00113     return instance().create (lKey);
00114 }
00115
00116 // //////////////////////////////////////////////////////////////////////
00117 template <typename BOM> BOM& FacBom<BOM>::create (const Key_T& iKey) {
00118     BOM* oBom_ptr = new BOM (iKey);
00119     assert (oBom_ptr != NULL);
00120     _pool.push_back (oBom_ptr);
00121     return *oBom_ptr;
00122 }
00123
00124 }
00125 #endif // __STDAIR_FAC_FACBOM_HPP

```

35.495 stdair/factory/FacBomManager.hpp File Reference

```

#include <iosfwd>
#include <list>
#include <map>
#include <boost/static_assert.hpp>
#include <boost/type_traits/is_same.hpp>
#include <stdair/bom/BomHolder.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/factory/FacAbstract.hpp>
#include <stdair/factory/FacBom.hpp>
#include <stdair/bom/SegmentDate.hpp>

```

Classes

- class [stdair::FacBomManager](#)
Utility class for linking StdAir-based objects.

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.496 FacBomManager.hpp

```

00001 #ifndef __STDAIR_FAC_FACBOMMANAGER_HPP
00002 #define __STDAIR_FAC_FACBOMMANAGER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <list>
00010 #include <map>
00011 // Boost
00012 #include <boost/static_assert.hpp>
00013 #include <boost/type_traits/is_same.hpp>
00014 // StdAir
00015 #include <stdair/bom/BomHolder.hpp>
00016 #include <stdair/bom/BomManager.hpp>
00017 #include <stdair/factory/FacAbstract.hpp>
00018 #include <stdair/factory/FacBom.hpp>
00019 // Stdair BOM Objects
00020 #include <stdair/bom/SegmentDate.hpp>
00021
00022

```

```

00023 namespace stdair {
00024
00028 class FacBomManager : public FacAbstract {
00029 public:
00030     // //////////// Business methods. ////////////
00039     template <typename OBJECT2, typename OBJECT1>
00040     static BomHolder<OBJECT2>* getBomHolderPtr (OBJECT1&);
00041
00051     template <typename OBJECT2, typename OBJECT1>
00052     static BomHolder<OBJECT2>& addBomHolder (OBJECT1&);
00053
00065     template <typename OBJECT1, typename OBJECT2>
00066     static void addToList (OBJECT1&, OBJECT2&);
00067
00080     template <typename OBJECT1, typename OBJECT2>
00081     static void addToMap (OBJECT1&, OBJECT2&, const MapKey_T&);
00082
00094     template <typename OBJECT1, typename OBJECT2>
00095     static void addToMap (OBJECT1&, OBJECT2&);
00096
00108     template <typename OBJECT1, typename OBJECT2>
00109     static void addToListAndMap (OBJECT1&, OBJECT2&);
00110
00123     template <typename OBJECT1, typename OBJECT2>
00124     static void addToListAndMap (OBJECT1&, OBJECT2&, const MapKey_T&);
00125
00132     template <typename PARENT, typename CHILD>
00133     static void linkWithParent (PARENT&, CHILD&);
00134
00145     template <typename OBJECT2, typename OBJECT1>
00146     static void cloneHolder (OBJECT1&, const OBJECT1&);
00147
00148
00149 private:
00162     template <typename OBJECT1, typename OBJECT2>
00163     static void addToList (BomHolder<OBJECT2>&, OBJECT1&, OBJECT2&);
00164
00178     template <typename OBJECT1, typename OBJECT2>
00179     static void addToMap (BomHolder<OBJECT2>&, OBJECT1&, OBJECT2&,
00180                          const MapKey_T&);
00181
00190     template <typename OBJECT2, typename OBJECT1>
00191     static BomHolder<OBJECT2>& getBomHolder (OBJECT1&);
00192
00193
00194 protected:
00200     FacBomManager() { }
00201
00202 public:
00206     ~FacBomManager() { }
00207 };
00208
00209 // //////////////////////////////////////
00210 // Public business method.
00211 // Compile time assertion to check OBJECT1 and OBJECT2 types.
00212 template <typename OBJECT2, typename OBJECT1>
00213 BomHolder<OBJECT2>& FacBomManager::addBomHolder (OBJECT1& ioObject1) {
00214
00215     //
00216     // Compile time assertion: this function must never be called with the
00217     // following list of couple types:
00218     // <SegmentDate, SegmentDate>

```

```

00219     //
00220     BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, SegmentDate>::value == false
00221                          || boost::is_same<OBJECT2, SegmentDate>::value == false
00222     ));
00223     BomHolder<OBJECT2>* lBomHolder_ptr =
00224         &FacBom<BomHolder<OBJECT2> >::instance().create();
00225
00226     const bool hasInsertBeenSuccessful =
00227         ioObject1._holderMap.insert (typename HolderMap_T::
00228                                     value_type (&typeid (OBJECT2),
00229                                                  lBomHolder_ptr)).second;
00230     assert (hasInsertBeenSuccessful == true);
00231
00232     return *lBomHolder_ptr;
00233 }
00234
00235 // //////////////////////////////////////
00236 // Public business method.
00237 // Compile time assertion to check OBJECT1 and OBJECT2 types.
00238 template <typename OBJECT2, typename OBJECT1>
00239 BomHolder<OBJECT2>* FacBomManager::getBomHolderPtr (OBJECT1& ioObject1) {
00240
00241     //
00242     // Compile time assertion: this function must never be called with the
00243     // following list of couple types:
00244     // <SegmentDate, SegmentDate>
00245     //
00246     BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, SegmentDate>::value == false
00247                          || boost::is_same<OBJECT2, SegmentDate>::value == false
00248     ));
00249     BomHolder<OBJECT2>* lBomHolder_ptr = NULL;
00250
00251     // Find the corresponding BomHolder within the object1, if existing.
00252     HolderMap_T::const_iterator itHolder =
00253         ioObject1._holderMap.find (&typeid (OBJECT2));
00254
00255     if (itHolder != ioObject1._holderMap.end()) {
00256         lBomHolder_ptr = static_cast<BomHolder<OBJECT2>*> (itHolder->second);
00257     }
00258
00259     return lBomHolder_ptr;
00260 }
00261
00262 // //////////////////////////////////////
00263 // Private method.
00264 template <typename OBJECT2, typename OBJECT1>
00265 BomHolder<OBJECT2>& FacBomManager::getBomHolder (OBJECT1& ioObject1) {
00266
00267     BomHolder<OBJECT2>* lBomHolder_ptr = NULL;
00268
00269     // Find the corresponding BomHolder within the object1. If it does
00270     // not exist, then create one.
00271     HolderMap_T::const_iterator itHolder =
00272         ioObject1._holderMap.find (&typeid (OBJECT2));
00273
00274     if (itHolder == ioObject1._holderMap.end()) {
00275         lBomHolder_ptr = &addBomHolder<OBJECT2, OBJECT1> (ioObject1);
00276     } else {
00277         lBomHolder_ptr = static_cast<BomHolder<OBJECT2>*> (itHolder->second);

```

```

00279     }
00280
00281     assert (lBomHolder_ptr != NULL);
00282
00283     return *lBomHolder_ptr;
00284 }
00285
00286 // //////////////////////////////////////
00287 // Private method.
00288 template <typename OBJECT1, typename OBJECT2>
00289 void FacBomManager::addToList (BomHolder<OBJECT2>& ioBomHolder,
00290                               OBJECT1& ioObject1, OBJECT2& ioObject2) {
00291     ioBomHolder._bomList.push_back (&ioObject2);
00292 }
00293
00294 // //////////////////////////////////////
00295 // Public business method.
00296 // This method is specialized for the following couple types:
00297 // <SegmentDate, SegmentDate>
00298 template <typename OBJECT1, typename OBJECT2>
00299 void FacBomManager::addToList (OBJECT1& ioObject1, OBJECT2& ioObject2) {
00300     BomHolder<OBJECT2>& lBomHolder = getBomHolder<OBJECT2> (ioObject1);
00301
00302     addToList<OBJECT1, OBJECT2> (lBomHolder, ioObject1, ioObject2);
00303 }
00304
00305 // //////////////////////////////////////
00306 // Private method.
00307 template <typename OBJECT1, typename OBJECT2>
00308 void FacBomManager::addToMap (BomHolder<OBJECT2>& ioBomHolder,
00309                              OBJECT1& ioObject1, OBJECT2& ioObject2,
00310                              const MapKey_T& iKey) {
00311
00312     const bool insertionSucceeded =
00313         ioBomHolder._bomMap.insert (typename std::map<const MapKey_T, OBJECT2*>::
00314                                     value_type (iKey, &ioObject2)).second;
00315
00316     if (insertionSucceeded == false) {
00317         // Build a nice message, so that the error be fully explicit
00318         std::ostringstream oStr;
00319         oStr << "The given object ('" << iKey
00320             << "') can not be added to the map of '" << ioObject1.describeKey()
00321             << "' object. That map already contains: '"
00322
00323         unsigned int idx = 0;
00324         for (typename std::map<const MapKey_T, OBJECT2*>::const_iterator iter =
00325             ioBomHolder._bomMap.begin();
00326             iter != ioBomHolder._bomMap.end(); ++iter, ++idx) {
00327             const OBJECT2* lCurrentObject_ptr = iter->second;
00328             assert (lCurrentObject_ptr != NULL);
00329
00330             if (idx != 0) {
00331                 oStr << "; ";
00332             }
00333             oStr << lCurrentObject_ptr->describeKey();
00334         }
00335         oStr << "'";
00336
00337         STDAIR_LOG_ERROR (oStr.str());
00338         throw ObjectLinkingException (oStr.str());
00339     }
00340 }

```

```

00341     }
00342
00343     // //////////////////////////////////////
00344     // Public business method.
00345     // Compile time assertion to check OBJECT1 and OBJECT2 types.
00346     template <typename OBJECT1, typename OBJECT2> void FacBomManager::
00347     addToMap (OBJECT1& ioObject1, OBJECT2& ioObject2, const MapKey_T& iKey) {
00348         //
00349         // Compile time assertion: this function must never be called with the
00350         // following list of couple types:
00351         // <SegmentDate, SegmentDate>
00352         //
00353         BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, SegmentDate>::value == false
00354                                || boost::is_same<OBJECT2, SegmentDate>::value == false
00355                            ));
00356         BomHolder<OBJECT2>& lBomHolder = getBomHolder<OBJECT2> (ioObject1);
00357         addToMap<OBJECT1, OBJECT2> (lBomHolder, ioObject1, ioObject2, iKey);
00358     }
00359
00360     // Public business method.
00361     // Compile time assertion to check OBJECT1 and OBJECT2 types.
00362     // //////////////////////////////////////
00363     template <typename OBJECT1, typename OBJECT2>
00364     void FacBomManager::addToMap (OBJECT1& ioObject1, OBJECT2& ioObject2) {
00365         //
00366         // Compile time assertion: this function must never be called with the
00367         // following list of couple types:
00368         // <SegmentDate, SegmentDate>
00369         //
00370         BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, SegmentDate>::value == false
00371                                || boost::is_same<OBJECT2, SegmentDate>::value == false
00372                            ));
00373         const MapKey_T& lKey = ioObject2.describeKey();
00374         addToMap (ioObject1, ioObject2, lKey);
00375     }
00376
00377     // //////////////////////////////////////
00378     // Public business method.
00379     // Compile time assertion to check OBJECT1 and OBJECT2 types.
00380     template <typename OBJECT1, typename OBJECT2>
00381     void FacBomManager::addToListAndMap (OBJECT1& ioObject1, OBJECT2& ioObject2,
00382                                           const MapKey_T& iKey) {
00383         //
00384         // Compile time assertion: this function must never be called with the
00385         // following list of couple types:
00386         // <SegmentDate, SegmentDate>
00387         //
00388         BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, SegmentDate>::value == false
00389                                || boost::is_same<OBJECT2, SegmentDate>::value == false
00390                            ));
00391         BomHolder<OBJECT2>& lBomHolder = getBomHolder<OBJECT2> (ioObject1);
00392         addToList<OBJECT1, OBJECT2> (lBomHolder, ioObject1, ioObject2);
00393         addToMap<OBJECT1, OBJECT2> (lBomHolder, ioObject1, ioObject2, iKey);
00394     }
00395
00396
00397
00398
00399

```

```

00400 // //////////////////////////////////////
00401 // Public business method.
00402 // Compile time assertion to check OBJECT1 and OBJECT2 types.
00403 template <typename OBJECT1, typename OBJECT2> void FacBomManager::
00404 addToListAndMap (OBJECT1& ioObject1, OBJECT2& ioObject2) {
00405
00406     //
00407     // Compile time assertion: this function must never be called with the
00408     // following list of couple types:
00409     // <SegmentDate, SegmentDate>
00410     //
00411     BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, SegmentDate>::value == false
00412         || boost::is_same<OBJECT2, SegmentDate>::value == false
00413     ));
00414
00415     const MapKey_T& lKey = ioObject2.describeKey();
00416     addToListAndMap<OBJECT1, OBJECT2> (ioObject1, ioObject2, lKey);
00417 }
00418
00419 // Public business method valid for all PARENT and CHILD types.
00420 // (No compile time assertion to check PARENT and CHILD types.)
00421 // //////////////////////////////////////
00422 template <typename PARENT, typename CHILD> void FacBomManager::
00423 linkWithParent (PARENT& ioParent, CHILD& ioChild) {
00424     ioChild._parent = &ioParent;
00425 }
00426
00427 // //////////////////////////////////////
00428 // Public business method valid for all PARENT and CHILD types.
00429 // (No compile time assertion to check PARENT and CHILD types.)
00430 template <typename OBJECT2, typename OBJECT1> void FacBomManager::
00431 cloneHolder (OBJECT1& ioDest, const OBJECT1& iOri) {
00432     const BomHolder<OBJECT2>& lOriginHolder =
00433         BomManager::getBomHolder<OBJECT2> (iOri);
00434
00435     BomHolder<OBJECT2>& lDestHolder = getBomHolder<OBJECT2> (ioDest);
00436     lDestHolder._bomList = lOriginHolder._bomList;
00437     lDestHolder._bomMap = lOriginHolder._bomMap;
00438 }
00439
00440 // //////////////////////////////////////
00441 //
00442 // Specialization of the template method addToList above for the types
00443 // <SegmentDate, SegmentDate>.
00444 // Add an element to the marketing segment date list of a segment date.
00445 //
00446 // //////////////////////////////////////
00447
00448 template<>
00449 inline void FacBomManager::addToList <SegmentDate,SegmentDate>
00450 (SegmentDate& ioSegmentDate,
00451  SegmentDate& ioMarketingSegmentDate) {
00452
00453     ioSegmentDate._marketingSegmentDateList.push_back(&ioMarketingSegmentDate);
00454 }
00455
00456 // //////////////////////////////////////
00457 //
00458 // TODO:
00459 // This specialization is needed for all the objects in the current
00460 // BOM tree.

```

```

00461 // (An inventory is the parent of flight dates, a flight date is the
00462 // parent of segment dates and leg dates, ...)
00463 //
00464 // //////////////////////////////////////
00465
00466
00467 }
00468
00469 // //////////////////////////////////////
00470
00471 #endif // __STDAIR_FAC_FACBOMMANAGER_HPP

```

35.497 stdair/service/DBSessionManager.cpp File Reference

```

#include <cassert>
#include <string>
#include <sstream>
#include <soci.h>
#include <mysql/soci-mysql.h>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/BasDBParams.hpp>
#include <stdair/service/DBSessionManager.hpp>
#include <stdair/service/Logger.hpp>

```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.498 DBSessionManager.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <string>
00007 #include <sstream>
00008 // SOCI
00009 #if defined(SOCI_HEADERS_BURIED)
00010 #include <soci/core/soci.h>
00011 #include <soci/backends/mysql/soci-mysql.h>
00012 #else // SOCI_HEADERS_BURIED
00013 #include <soci.h>
00014 #include <mysql/soci-mysql.h>
00015 #endif // SOCI_HEADERS_BURIED
00016 // StdAir
00017 #include <stdair/stdair_exceptions.hpp>
00018 #include <stdair/basic/BasDBParams.hpp>

```

```

00019 #include <stdair/service/DBSessionManager.hpp>
00020 #include <stdair/service/Logger.hpp>
00021
00022 namespace stdair {
00023
00024 ///////////////////////////////////////////////////////////////////
00025 DBSessionManager::DBSessionManager () : _dbSession (NULL) {
00026 }
00027
00028 ///////////////////////////////////////////////////////////////////
00029 DBSessionManager::DBSessionManager (const DBSessionManager&)
00030 : _dbSession (NULL) {
00031     assert (false);
00032 }
00033
00034 ///////////////////////////////////////////////////////////////////
00035 DBSessionManager::~DBSessionManager () {
00036     // std::cout << "In DBSessionManager destructor" << std::endl;
00037     dbFinalise();
00038 }
00039
00040 ///////////////////////////////////////////////////////////////////
00041 void DBSessionManager::dbInit (const BasDBParams& iDBParams) {
00042
00043     // Database parameters
00044     std::ostringstream oStr;
00045     oStr << "db=" << iDBParams.getDBName() << " user=" << iDBParams.getUser()
00046         << " password=" << iDBParams.getPassword()
00047         << " port=" << iDBParams.getPort() << " host=" << iDBParams.getHost();
00048     const std::string lDBSessionConnectionString (oStr.str());
00049
00050     // Instanciate the database session: nothing else is performed at that stage
00051     _dbSession = new DBSession_T;
00052
00053     try {
00054         // Open the connection to the database
00055         _dbSession->open (soci::mysql, lDBSessionConnectionString);
00056
00057     } catch (std::exception const& lException) {
00058         std::ostringstream oMessage;
00059         oMessage <<"Error while opening a connection to database: "
00060             << lException.what() << std::endl
00061             << "Database parameters used:"
00062             << " db=" << iDBParams.getDBName()
00063             << " user=" << iDBParams.getUser()
00064             << " port=" << iDBParams.getPort()
00065             << " host=" << iDBParams.getHost();
00066         throw SQLDatabaseConnectionImpossibleException (oMessage.str());
00067     }
00068 }
00069
00070 ///////////////////////////////////////////////////////////////////
00071 void DBSessionManager::dbFinalise () {
00072     delete _dbSession; _dbSession = NULL;
00073 }
00074
00075 ///////////////////////////////////////////////////////////////////
00076 void DBSessionManager::init (const BasDBParams& iDBParams) {
00077     DBSessionManager& lInstance = instance();
00078     lInstance.dbInit (iDBParams);
00079 }
00080

```

```

00081 // //////////////////////////////////////
00082 DBSessionManager& DBSessionManager::instance() {
00083     static DBSessionManager _instance;
00084     return _instance;
00085 }
00086
00087 // //////////////////////////////////////
00088 void DBSessionManager::clean() {
00089 }
00090
00091 // //////////////////////////////////////
00092 DBSession_T& DBSessionManager::getDBSession() const {
00093     if (_dbSession == NULL) {
00094         throw NonInitialisedDBSessionManagerException ("");
00095     }
00096     assert (_dbSession != NULL);
00097     return *_dbSession;
00098 }
00099
00100 }

```

35.499 stdair/service/DBSessionManager.hpp File Reference

```
#include <stdair/stdair_db.hpp>
```

Classes

- class [stdair::DBSessionManager](#)

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.500 DBSessionManager.hpp

```

00001 #ifndef __STDAIR_SVC_DBSESSIONMANAGER_HPP
00002 #define __STDAIR_SVC_DBSESSIONMANAGER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_db.hpp>
00009
00010 namespace stdair {
00011
00012     // Forward declarations
00013     struct BasDBParams;
00014
00017     class DBSessionManager {
00018         // Friend classes
00019         friend class FacSupervisor;
00020         friend class STDAIR_Service;

```

```

00021
00022     public:
00024         static DBSessionManager& instance();
00025
00028         DBSession_T& getDBSession() const;
00029
00030
00031     private:
00034         DBSessionManager ();
00036         DBSessionManager (const DBSessionManager&);
00038         ~DBSessionManager ();
00039
00042         void dbInit (const BasDBParams&);
00043
00046         void dbFinalise ();
00047
00048
00049     private:
00052         static void init (const BasDBParams&);
00053
00055         static void clean();
00056
00057
00058     private:
00060         DBSession_T* _dbSession;
00061     };
00062
00063 }
00064 #endif // __STDAIR_SVC_DBSESSIONMANAGER_HPP

```

35.501 stdair/service/FacServiceAbstract.cpp File Reference

```

#include <cassert>
#include <stdair/service/ServiceAbstract.hpp>
#include <stdair/service/FacServiceAbstract.hpp>

```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

35.502 FacServiceAbstract.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // StdAir
00007 #include <stdair/service/ServiceAbstract.hpp>
00008 #include <stdair/service/FacServiceAbstract.hpp>
00009
00010 namespace stdair {
00011

```

```

00012 // //////////////////////////////////////
00013 FacServiceAbstract::~FacServiceAbstract() {
00014     clean();
00015 }
00016
00017 // //////////////////////////////////////
00018 void FacServiceAbstract::clean() {
00019     for (ServicePool_T::iterator itService = _pool.begin();
00020          itService != _pool.end(); itService++) {
00021         ServiceAbstract* currentService_ptr = *itService;
00022         assert (currentService_ptr != NULL);
00023
00024         delete (currentService_ptr); currentService_ptr = NULL;
00025     }
00026
00027     // Empty the pool of Service Factories
00028     _pool.clear();
00029 }
00030
00031 }

```

35.503 stdair/service/FacServiceAbstract.hpp File Reference

```
#include <vector>
```

Classes

- class [stdair::FacServiceAbstract](#)

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.504 FacServiceAbstract.hpp

```

00001 #ifndef __STDAIR_SVC_FACSERVICEABSTRACT_HPP
00002 #define __STDAIR_SVC_FACSERVICEABSTRACT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <vector>
00009
00010 namespace stdair {
00011
00012     // Forward declarations
00013     class ServiceAbstract;
00014
00015     class FacServiceAbstract {
00016     public:
00017
00018         typedef std::vector<ServiceAbstract*> ServicePool_T;

```

```

00021
00022     virtual ~FacServiceAbstract();
00023
00024     void clean();
00025
00026 protected:
00027     FacServiceAbstract() {}
00028
00029     ServicePool_T _pool;
00030 };
00031
00032 }
00033 #endif // __STDAIR_SVC_FACSERVICEABSTRACT_HPP

```

35.505 stdair/service/FacSTDAIRServiceContext.cpp File Reference

```

#include <cassert>
#include <stdair/service/FacSupervisor.hpp>
#include <stdair/service/FacSTDAIRServiceContext.hpp>
#include <stdair/service/STDAIR_ServiceContext.hpp>

```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

35.506 FacSTDAIRServiceContext.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // StdAir
00007 #include <stdair/service/FacSupervisor.hpp>
00008 #include <stdair/service/FacSTDAIRServiceContext.hpp>
00009 #include <stdair/service/STDAIR_ServiceContext.hpp>
00010
00011 namespace stdair {
00012
00013     FacSTDAIRServiceContext* FacSTDAIRServiceContext::_instance = NULL;
00014
00015     // //////////////////////////////////////
00016     FacSTDAIRServiceContext::~FacSTDAIRServiceContext() {
00017         _instance = NULL;
00018     }
00019
00020     // //////////////////////////////////////
00021     FacSTDAIRServiceContext& FacSTDAIRServiceContext::instance() {
00022
00023         if (_instance == NULL) {
00024             _instance = new FacSTDAIRServiceContext();
00025             assert (_instance != NULL);

```

```

00026
00027     FacSupervisor::instance().registerServiceFactory (_instance);
00028 }
00029 return *_instance;
00030 }
00031
00032 // //////////////////////////////////////
00033 STDAIR_ServiceContext& FacSTDAIRServiceContext::create() {
00034     STDAIR_ServiceContext* aServiceContext_ptr = NULL;
00035
00036     aServiceContext_ptr = new STDAIR_ServiceContext ();
00037     assert (aServiceContext_ptr != NULL);
00038
00039     // The new object is added to the Bom pool
00040     _pool.push_back (aServiceContext_ptr);
00041
00042     return *aServiceContext_ptr;
00043 }
00044
00045 }

```

35.507 stdair/service/FacSTDAIRServiceContext.hpp File Reference

```
#include <stdair/service/FacServiceAbstract.hpp>
```

Classes

- class [stdair::FacSTDAIRServiceContext](#)
Factory for [Bucket](#).

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.508 FacSTDAIRServiceContext.hpp

```

00001 #ifndef __STDAIR_SVC_FACSTDAIRSERVICECONTEXT_HPP
00002 #define __STDAIR_SVC_FACSTDAIRSERVICECONTEXT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/service/FacServiceAbstract.hpp>
00009
00010 namespace stdair {
00011
00012     class STDAIR_ServiceContext;
00013
00014     class FacSTDAIRServiceContext : public FacServiceAbstract {
00015     public:
00016
00017     };
00018
00019 }

```

```

00028     static FacSTDAIRServiceContext& instance();
00029
00036     ~FacSTDAIRServiceContext();
00037
00045     STDAIR_ServiceContext& create();
00046
00047
00048     protected:
00054     FacSTDAIRServiceContext() {}
00055
00056     private:
00060     static FacSTDAIRServiceContext* _instance;
00061 };
00062
00063 }
00064 #endif // __STDAIR_SVC_FACSTDAIRSERVICECONTEXT_HPP

```

35.509 stdair/service/FacSupervisor.cpp File Reference

```

#include <cassert>
#include <stdair/factory/FacAbstract.hpp>
#include <stdair/service/FacServiceAbstract.hpp>
#include <stdair/service/FacSupervisor.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/service/DBSessionManager.hpp>

```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.510 FacSupervisor.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // StdAir
00007 #include <stdair/factory/FacAbstract.hpp>
00008 #include <stdair/service/FacServiceAbstract.hpp>
00009 #include <stdair/service/FacSupervisor.hpp>
00010 #include <stdair/service/Logger.hpp>
00011 #include <stdair/service/DBSessionManager.hpp>
00012
00013 namespace stdair {
00014
00015     FacSupervisor* FacSupervisor::_instance = NULL;
00016
00017 // //////////////////////////////////////
00018 FacSupervisor& FacSupervisor::instance() {

```

```

00019     if (_instance == NULL) {
00020         _instance = new FacSupervisor();
00021     }
00022
00023     return *_instance;
00024 }
00025
00026 // //////////////////////////////////////
00027 FacSupervisor::~FacSupervisor() {
00028     cleanBomLayer();
00029     cleanServiceLayer();
00030 }
00031
00032 // //////////////////////////////////////
00033 void FacSupervisor::registerBomFactory (FacAbstract* ioFac_ptr) {
00034     _bomPool.push_back (ioFac_ptr);
00035 }
00036
00037 // //////////////////////////////////////
00038 void FacSupervisor::registerServiceFactory (FacServiceAbstract* ioFac_ptr) {
00039     _svcPool.push_back (ioFac_ptr);
00040 }
00041
00042 // //////////////////////////////////////
00043 void FacSupervisor::cleanBomLayer() {
00044     for (BomFactoryPool_T::const_iterator itFactory = _bomPool.begin();
00045          itFactory != _bomPool.end(); itFactory++) {
00046         const FacAbstract* currentFactory_ptr = *itFactory;
00047         assert (currentFactory_ptr != NULL);
00048
00049         delete (currentFactory_ptr); currentFactory_ptr = NULL;
00050     }
00051
00052     // Empty the pool of BOM factories
00053     _bomPool.clear();
00054 }
00055
00056 // //////////////////////////////////////
00057 void FacSupervisor::cleanServiceLayer() {
00058     for (ServiceFactoryPool_T::const_iterator itFactory = _svcPool.begin();
00059          itFactory != _svcPool.end(); itFactory++) {
00060         const FacServiceAbstract* currentFactory_ptr = *itFactory;
00061         assert (currentFactory_ptr != NULL);
00062
00063         delete (currentFactory_ptr); currentFactory_ptr = NULL;
00064     }
00065
00066     // Empty the pool of Service Factories
00067     _svcPool.clear();
00068 }
00069
00070 // //////////////////////////////////////
00071 void FacSupervisor::cleanLoggerService() {
00072     // Clean the static instance of the log service
00073     Logger::clean();
00074 }
00075
00076 // //////////////////////////////////////
00077 void FacSupervisor::cleanDBSessionManager() {
00078     // Clean the static instance of the database service
00079     DBSessionManager::clean();
00080 }

```

```

00081
00082 // //////////////////////////////////////
00083 void FacSupervisor::cleanAll() {
00084
00085     // Clean the static instance of the database session manager
00086     cleanDBSessionManager();
00087
00088     // Clean the static instance of the log service
00089     cleanLoggerService();
00090
00091     // Clean the static instance of the FacSupervisor.
00092     // This in turn will invoke the destructor (~FacSupervisor() method)
00093     // of the static instance, thus cleaning both the BOM and service layers.
00094     delete _instance; _instance = NULL;
00095 }
00096
00097 }

```

35.511 stdair/service/FacSupervisor.hpp File Reference

```

#include <iosfwd>
#include <list>

```

Classes

- class [stdair::FacSupervisor](#)

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.512 FacSupervisor.hpp

```

00001 #ifndef __STDAIR_SVC_FACSUPERVISOR_HPP
00002 #define __STDAIR_SVC_FACSUPERVISOR_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <list>
00010
00011 namespace stdair {
00012
00014     class FacAbstract;
00015     class FacServiceAbstract;
00016
00020     class FacSupervisor {
00021     public:
00025         typedef std::list<FacAbstract*> BomFactoryPool_T;
00026         typedef std::list<FacServiceAbstract*> ServiceFactoryPool_T;

```

```

00027
00034     static FacSupervisor& instance();
00035
00043     void registerBomFactory (FacAbstract*);
00044
00052     void registerServiceFactory (FacServiceAbstract*);
00053
00060     void cleanBomLayer();
00061
00068     void cleanServiceLayer();
00069
00073     static void cleanLoggerService();
00074
00078     static void cleanDBSessionManager();
00079
00085     static void cleanAll();
00086
00093     ~FacSupervisor();
00094
00095
00096 protected:
00102     FacSupervisor() {}
00103     FacSupervisor (const FacSupervisor&) {}
00104
00105 private:
00109     static FacSupervisor* _instance;
00110
00114     BomFactoryPool_T _bomPool;
00115
00119     ServiceFactoryPool_T _svcPool;
00120 };
00121 }
00122 #endif // __STDAIR_SVC_FACSUPERVISOR_HPP

```

35.513 stdair/service/Logger.cpp File Reference

```

#include <iostream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/service/Logger.hpp>

```

Namespaces

- namespace `stdair`
Handle on the *StdAir* library context.

35.514 Logger.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <iostream>
00006 // StdAir Logger
00007 #include <stdair/stdair_exceptions.hpp>

```

```

00008 #include <stdair/service/Logger.hpp>
00009
00010 namespace stdair {
00011
00012 // //////////////////////////////////////
00013 Logger::Logger()
00014 : _level (LOG::DEBUG), _logStream (&std::cout),
00015   _hasBeenInitialised (false) {
00016 }
00017
00018 // //////////////////////////////////////
00019 Logger::Logger (const Logger&)
00020 : _level (LOG::DEBUG), _logStream (&std::cout),
00021   _hasBeenInitialised (false) {
00022   assert (false);
00023 }
00024
00025 // //////////////////////////////////////
00026 Logger::~Logger() {
00027   // std::cout << "In Logger destructor" << std::endl;
00028 }
00029
00030 // //////////////////////////////////////
00031 void Logger::init (const BasLogParams& iLogParams) {
00032
00033   //
00034   Logger& lInstance = instance();
00035   const bool hasBeenInitialised = lInstance.getStatus();
00036   if (hasBeenInitialised == true
00037       && iLogParams.getForcedInitialisationFlag() == false) {
00038     STDAIR_LOG_ERROR ("Error: the log stream has already been initialised");
00039     assert (false);
00040   }
00041
00042   lInstance.setLevel (iLogParams._logLevel);
00043   lInstance.setStream (iLogParams._logStream);
00044   lInstance.setStatus (true);
00045 }
00046
00047 // //////////////////////////////////////
00048 Logger& Logger::instance() {
00049   static Logger _instance;
00050   return _instance;
00051 }
00052
00053 // //////////////////////////////////////
00054 BasLogParams Logger::getLogParams() {
00055   std::ostream* oStream_ptr = instance()._logStream;
00056   assert (oStream_ptr != NULL);
00057   return BasLogParams (instance()._level, *oStream_ptr);
00058 }
00059
00060 // //////////////////////////////////////
00061 void Logger::clean() {
00062   Logger& lInstance = instance();
00063   lInstance.setStatus (false);
00064 }
00065
00066 }

```

35.515 stdair/service/Logger.hpp File Reference

```
#include <cassert>
#include <sstream>
#include <string>
#include <stdair/stdair_log.hpp>
#include <stdair/basic/BasLogParams.hpp>
```

Classes

- class [stdair::Logger](#)

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

Defines

- #define [STDAIR_LOG_CORE](#)(iLevel, iToBeLogged)
- #define [STDAIR_LOG_CRITICAL](#)(iToBeLogged) STDAIR_LOG_CORE (stdair::LOG::CRITICAL, iToBeLogged)
- #define [STDAIR_LOG_ERROR](#)(iToBeLogged) STDAIR_LOG_CORE (stdair::LOG::ERROR, iToBeLogged)
- #define [STDAIR_LOG_NOTIFICATION](#)(iToBeLogged) STDAIR_LOG_CORE (stdair::LOG::NOTIFICATION, iToBeLogged)
- #define [STDAIR_LOG_WARNING](#)(iToBeLogged) STDAIR_LOG_CORE (stdair::LOG::WARNING, iToBeLogged)
- #define [STDAIR_LOG_DEBUG](#)(iToBeLogged) STDAIR_LOG_CORE (stdair::LOG::DEBUG, iToBeLogged)
- #define [STDAIR_LOG_VERBOSE](#)(iToBeLogged) STDAIR_LOG_CORE (stdair::LOG::VERBOSE, iToBeLogged)

35.515.1 Define Documentation

35.515.1.1 #define STDAIR_LOG_CORE(iLevel, iToBeLogged)

Value:

```
{ std::ostringstream ostr; ostr << iToBeLogged; \
  stdair::Logger::instance()<log (iLevel, __LINE__, __FILE__, ostr.str()); }
```

Definition at line 16 of file [Logger.hpp](#).

35.515.1.2 `#define STDAIR_LOG_CRITICAL(iToBeLogged) STDAIR_LOG_CORE`
`(stdair::LOG::CRITICAL, iToBeLogged)`

Definition at line 20 of file [Logger.hpp](#).

35.515.1.3 `#define STDAIR_LOG_ERROR(iToBeLogged) STDAIR_LOG_CORE`
`(stdair::LOG::ERROR, iToBeLogged)`

Definition at line 23 of file [Logger.hpp](#).

Referenced by [stdair::EventQueue::calculateProgress\(\)](#), [stdair::EventQueue::getActualTotalNbOfEvents\(\)](#), [stdair::ParsedKey::getBoardingTime\(\)](#), [stdair::EventQueue::getCurrentNbOfEvents\(\)](#), [stdair::EventQueue::getExpectedNbOfEvents\(\)](#), [stdair::ParsedKey::getFlightDateKey\(\)](#), [stdair::ParsedKey::getInventoryKey\(\)](#), [stdair::BomManager::getObject\(\)](#), [stdair::ParsedKey::getSegmentKey\(\)](#), and [stdair::EventQueue::updateStatus\(\)](#).

35.515.1.4 `#define STDAIR_LOG_NOTIFICATION(iToBeLogged) STDAIR_LOG_CORE`
`(stdair::LOG::NOTIFICATION, iToBeLogged)`

Definition at line 26 of file [Logger.hpp](#).

35.515.1.5 `#define STDAIR_LOG_WARNING(iToBeLogged) STDAIR_LOG_CORE`
`(stdair::LOG::WARNING, iToBeLogged)`

Definition at line 29 of file [Logger.hpp](#).

35.515.1.6 `#define STDAIR_LOG_DEBUG(iToBeLogged) STDAIR_LOG_CORE`
`(stdair::LOG::DEBUG, iToBeLogged)`

Definition at line 32 of file [Logger.hpp](#).

Referenced by [stdair::ParsedKey::getBoardingTime\(\)](#), [stdair::ParsedKey::getFlightDateKey\(\)](#), [stdair::ParsedKey::getInventoryKey\(\)](#), [stdair::ParsedKey::getSegmentKey\(\)](#), [stdair::TimePeriod::isDepartureTimeValid\(\)](#), and [stdair::BomRetriever::retrieveSegmentDateFromLongKey\(\)](#).

35.515.1.7 `#define STDAIR_LOG_VERBOSE(iToBeLogged) STDAIR_LOG_CORE`
`(stdair::LOG::VERBOSE, iToBeLogged)`

Definition at line 35 of file [Logger.hpp](#).

35.516 Logger.hpp

```
00001 #ifndef __STDAIR_SVC_LOGGER_HPP
00002 #define __STDAIR_SVC_LOGGER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <cassert>
00009 #include <sstream>
00010 #include <string>
00011 // StdAir
00012 #include <stdair/stdair_log.hpp>
00013 #include <stdair/basic/BasLogParams.hpp>
```

```

00014
00015 // ////////////////////////////////// LOG MACROS //////////////////////////////////
00016 #define STDAIR_LOG_CORE(iLevel, iToBeLogged) \
00017     { std::ostringstream ostr; ostr << iToBeLogged; \
00018       stdair::Logger::instance().log (iLevel, __LINE__, __FILE__, ostr.str()); }
00019
00020 #define STDAIR_LOG_CRITICAL(iToBeLogged) \
00021     STDAIR_LOG_CORE (stdair::LOG::CRITICAL, iToBeLogged)
00022
00023 #define STDAIR_LOG_ERROR(iToBeLogged) \
00024     STDAIR_LOG_CORE (stdair::LOG::ERROR, iToBeLogged)
00025
00026 #define STDAIR_LOG_NOTIFICATION(iToBeLogged) \
00027     STDAIR_LOG_CORE (stdair::LOG::NOTIFICATION, iToBeLogged)
00028
00029 #define STDAIR_LOG_WARNING(iToBeLogged) \
00030     STDAIR_LOG_CORE (stdair::LOG::WARNING, iToBeLogged)
00031
00032 #define STDAIR_LOG_DEBUG(iToBeLogged) \
00033     STDAIR_LOG_CORE (stdair::LOG::DEBUG, iToBeLogged)
00034
00035 #define STDAIR_LOG_VERBOSE(iToBeLogged) \
00036     STDAIR_LOG_CORE (stdair::LOG::VERBOSE, iToBeLogged)
00037 // ////////////////////////////////// (END OF) LOG MACROS //////////////////////////////////
00038
00039
00040 namespace stdair {
00041
00042     class Logger {
00043     friend class FacSupervisor;
00044     friend class STDAIR_Service;
00045
00046     public:
00047
00048         template <typename T>
00049         void log (const LOG::EN_LogLevel iLevel, const int iLineNumber,
00050                  const std::string& iFileName, const T& iToBeLogged) {
00051             assert (_logStream != NULL);
00052             if (iLevel <= _level) {
00053                 *_logStream << "[" << LOG::_logLevels[iLevel] << "]" << iFileName << ":"
00054                    << iLineNumber << ": " << iToBeLogged << std::endl;
00055             }
00056         }
00057
00058         static Logger& instance();
00059
00060     private:
00061         // ////////////////////////////////// Initialisation and finalisation //////////////////////////////////
00062         bool getStatus() const {
00063             return _hasBeenInitialised;
00064         }
00065
00066         void setLevel (const LOG::EN_LogLevel& iLevel) {
00067             _level = iLevel;
00068         }
00069
00070         void setStream (std::ostream& ioStream) {
00071             _logStream = &ioStream;
00072         }
00073
00074         void setStatus (const bool iStatus) {

```

```

00101     _hasBeenInitialised = iStatus;
00102 }
00103
00104     Logger();
00112     Logger (const Logger&);
00116     ~Logger();
00117
00123     static void init (const BasLogParams&);
00124
00128     static BasLogParams getLogParams();
00129
00133     static void clean();
00134
00135
00136 private:
00137     // ////////////////////////////////// Attributes //////////////////////////////////
00141     LOG::EN_LogLevel _level;
00142
00146     std::ostream* _logStream;
00147
00151     bool _hasBeenInitialised;
00152 };
00153
00154 }
00155 #endif // __STDAIR_SVC_LOGGER_HPP
00156

```

35.517 stdair/service/ServiceAbstract.cpp File Reference

```
#include <stdair/service/ServiceAbstract.hpp>
```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

35.518 ServiceAbstract.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // StdAir
00005 #include <stdair/service/ServiceAbstract.hpp>
00006
00007 namespace stdair {
00008
00009 }

```

35.519 stdair/service/ServiceAbstract.hpp File Reference

```
#include <iosfwd>
```

Classes

- class [stdair::ServiceAbstract](#)

Namespaces

- namespace [stdair](#)

Handle on the StdAir library context.

Functions

- template<class charT , class traits >
std::basic_ostream< charT, traits > & [operator<<](#) (std::basic_ostream< charT, traits > &ioOut, const [stdair::ServiceAbstract](#) &iService)
- template<class charT , class traits >
std::basic_istream< charT, traits > & [operator>>](#) (std::basic_istream< charT, traits > &ioln, [stdair::ServiceAbstract](#) &iService)

35.519.1 Function Documentation

35.519.1.1 template<class charT , class traits > std::basic_ostream<charT, traits>&
operator<< (std::basic_ostream< charT, traits > & *ioOut*, const
[stdair::ServiceAbstract](#) & *iService*) [inline]

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (p653) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line [58](#) of file [ServiceAbstract.hpp](#).

References [stdair::ServiceAbstract::toStream\(\)](#).

35.519.1.2 template<class charT , class traits > std::basic_istream<charT, traits>& operator>> (std::basic_istream< charT, traits > & *ioln*,
[stdair::ServiceAbstract](#) & *iService*) [inline]

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (pp655-657) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line [86](#) of file [ServiceAbstract.hpp](#).

References [stdair::ServiceAbstract::fromStream\(\)](#).

35.520 ServiceAbstract.hpp

```
00001 #ifndef __STDAIR_SVC_SERVICEABSTRACT_HPP
00002 #define __STDAIR_SVC_SERVICEABSTRACT_HPP
00003
00004 // //////////////////////////////////////
```

```

00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009
00010 namespace stdair {
00011
00015     class ServiceAbstract {
00016     public:
00017
00021         virtual ~ServiceAbstract() {}
00022
00028         virtual void toStream (std::ostream& ioOut) const {}
00029
00035         virtual void fromStream (std::istream& ioIn) {}
00036
00040         // virtual const std::string describe() const = 0;
00041
00042     protected:
00046         ServiceAbstract() {}
00047     };
00048 }
00049
00055 template <class charT, class traits>
00056 inline
00057 std::basic_ostream<charT, traits>&
00058 operator<< (std::basic_ostream<charT, traits>& ioOut,
00059           const stdair::ServiceAbstract& iService) {
00065     std::basic_ostringstream<charT,traits> ostr;
00066     ostr.copyfmt (ioOut);
00067     ostr.width (0);
00068
00069     // Fill string stream
00070     iService.toStream (ostr);
00071
00072     // Print string stream
00073     ioOut << ostr.str();
00074
00075     return ioOut;
00076 }
00077
00083 template <class charT, class traits>
00084 inline
00085 std::basic_istream<charT, traits>&
00086 operator>> (std::basic_istream<charT, traits>& ioIn,
00087            stdair::ServiceAbstract& ioService) {
00088     // Fill Service object with input stream
00089     ioService.fromStream (ioIn);
00090     return ioIn;
00091 }
00092
00093 #endif // __STDAIR_SVC_SERVICEABSTRACT_HPP

```

35.521 stdair/service/STDAIR_Service.cpp File Reference

```

#include <cassert>

#include <sstream>

#include <stdair/stdair_types.hpp>

```

```

#include <stdair/basic/BasChronometer.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomRetriever.hpp>
#include <stdair/bom/BomJSONExport.hpp>
#include <stdair/bom/BomDisplay.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/EventQueue.hpp>
#include <stdair/bom/EventStruct.hpp>
#include <stdair/bom/BookingRequestStruct.hpp>
#include <stdair/bom/DatePeriod.hpp>
#include <stdair/command/CmdBomManager.hpp>
#include <stdair/service/FacSupervisor.hpp>
#include <stdair/service/FacSTDAIRServiceContext.hpp>
#include <stdair/service/STDAIR_ServiceContext.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/service/DBSessionManager.hpp>
#include <stdair/STDAIR_Service.hpp>

```

Namespaces

- namespace [bpt](#)
- namespace [stdair](#)

Handle on the StdAir library context.

35.522 STDAIR_Service.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 #if BOOST_VERSION >= 104100
00008 // Boost Property Tree
00009 #include <boost/property_tree/ptree.hpp>
00010 #include <boost/property_tree/json_parser.hpp>
00011 #endif // BOOST_VERSION >= 104100
00012 // StdAir
00013 #include <stdair/stdair_types.hpp>
00014 #include <stdair/basic/BasChronometer.hpp>
00015 #include <stdair/bom/BomManager.hpp>
00016 #include <stdair/bom/BomRetriever.hpp>
00017 #include <stdair/bom/BomJSONExport.hpp>
00018 #include <stdair/bom/BomDisplay.hpp>

```

```
00019 #include <stdair/bom/BomRoot.hpp>
00020 #include <stdair/bom/EventQueue.hpp>
00021 #include <stdair/bom/EventStruct.hpp>
00022 #include <stdair/bom/BookingRequestStruct.hpp>
00023 #include <stdair/bom/DatePeriod.hpp>
00024 #include <stdair/command/CmdBomManager.hpp>
00025 #include <stdair/service/FacSupervisor.hpp>
00026 #include <stdair/service/FacSTDAIRServiceContext.hpp>
00027 #include <stdair/service/STDAIR_ServiceContext.hpp>
00028 #include <stdair/service/Logger.hpp>
00029 #include <stdair/service/DBSessionManager.hpp>
00030 #include <stdair/STDAIR_Service.hpp>
00031
00032 #if BOOST_VERSION >= 104100
00033 namespace bpt = boost::property_tree;
00034 #else // BOOST_VERSION >= 104100
00035 namespace bpt {
00036     typedef char ptree;
00037 }
00038 #endif // BOOST_VERSION >= 104100
00039
00040 namespace stdair {
00041
00042     //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00043     STDAIR_Service::STDAIR_Service() : _stdairServiceContext (NULL) {
00044
00045         // Initialise the service context
00046         initServiceContext();
00047
00048         // Initialise the (remaining of the) context
00049         init();
00050     }
00051
00052     //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00053     STDAIR_Service::STDAIR_Service (const STDAIR_Service& iService)
00054         : _stdairServiceContext (NULL) {
00055         assert (false);
00056     }
00057
00058     //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00059     STDAIR_Service::STDAIR_Service (const BasLogParams& iLogParams)
00060         : _stdairServiceContext (NULL) {
00061
00062         // Initialise the service context
00063         initServiceContext();
00064
00065         // Set the log file
00066         logInit (iLogParams);
00067
00068         // Initialise the (remaining of the) context
00069         init();
00070     }
00071
00072     //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00073     STDAIR_Service::STDAIR_Service (const BasLogParams& iLogParams,
00074                                     const BasDBParams& iDBParams)
00075         : _stdairServiceContext (NULL) {
00076
00077         // Initialise the service context
00078         initServiceContext();
00079
00080         // Set the log file
```

```

00081     logInit (iLogParams);
00082
00083     // Create a database session
00084     dbInit (iDBParams);
00085
00086     // Initialise the (remaining of the) context
00087     init();
00088 }
00089
00090 // //////////////////////////////////////
00091 STDAIR_Service::~STDAIR_Service() {
00092     // Delete/Clean all the objects from memory
00093     finalise();
00094 }
00095
00096 // //////////////////////////////////////
00097 void STDAIR_Service::initServiceContext() {
00098     // Initialise the service context
00099     STDAIR_ServiceContext& lSTDAIR_ServiceContext =
00100         FacSTDAIRServiceContext::instance().create();
00101
00102     // Store the stdair service context
00103     _stdairServiceContext = &lSTDAIR_ServiceContext;
00104 }
00105
00106 // //////////////////////////////////////
00107 void STDAIR_Service::logInit (const BasLogParams& iLogParams) {
00108     Logger::init (iLogParams);
00109 }
00110
00111 // //////////////////////////////////////
00112 void STDAIR_Service::dbInit (const BasDBParams& iDBParams) {
00113     DBSessionManager::init (iDBParams);
00114
00115     // Store the database parameters into the StdAir service context
00116     assert (_stdairServiceContext != NULL);
00117     STDAIR_ServiceContext& lSTDAIR_ServiceContext = *_stdairServiceContext;
00118     lSTDAIR_ServiceContext.setDBParams (iDBParams);
00119 }
00120
00121 // //////////////////////////////////////
00122 void STDAIR_Service::init() {
00123 }
00124
00125 // //////////////////////////////////////
00126 BomRoot& STDAIR_Service::getBomRoot() const {
00127     // Retrieve the StdAir service context
00128     assert (_stdairServiceContext != NULL);
00129     const STDAIR_ServiceContext& lSTDAIR_ServiceContext = *_stdairServiceContext;
00130
00131     return lSTDAIR_ServiceContext.getBomRoot();
00132 }
00133
00134 // //////////////////////////////////////
00135 EventQueue& STDAIR_Service::getEventQueue() const {
00136     // Retrieve the StdAir service context
00137     assert (_stdairServiceContext != NULL);
00138     const STDAIR_ServiceContext& lSTDAIR_ServiceContext = *_stdairServiceContext;
00139
00140     return lSTDAIR_ServiceContext.getEventQueue();
00141 }

```

```

00141 // //////////////////////////////////////
00142 BasLogParams STDAIR_Service::getLogParams() const {
00143     return Logger::getLogParams();
00144 }
00145
00146 // //////////////////////////////////////
00147 const BasDBParams& STDAIR_Service::getDBParams() const {
00148     // Retrieve the StdAir service context
00149     assert (_stdairServiceContext != NULL);
00150     const STDAIR_ServiceContext& lSTDAIR_ServiceContext = *_stdairServiceContext;
00151
00152     return lSTDAIR_ServiceContext.getDBParams();
00153 }
00154
00155 // //////////////////////////////////////
00156 const ServiceInitialisationType& STDAIR_Service::
00157 getServiceInitialisationType() const {
00158     // Retrieve the StdAir service context
00159     assert (_stdairServiceContext != NULL);
00160     const STDAIR_ServiceContext& lSTDAIR_ServiceContext = *_stdairServiceContext;
00161
00162     return lSTDAIR_ServiceContext.getServiceInitialisationType();
00163 }
00164
00165 // //////////////////////////////////////
00166 void STDAIR_Service::buildSampleBom() {
00167     // Retrieve the StdAir service context
00168     assert (_stdairServiceContext != NULL);
00169     const STDAIR_ServiceContext& lSTDAIR_ServiceContext = *_stdairServiceContext;
00170
00171     // Retrieve the BOM tree root
00172     BomRoot& lBomRoot = lSTDAIR_ServiceContext.getBomRoot();
00173
00174     // Delegate the building process to the dedicated command
00175     CmdBomManager::buildSampleBom (lBomRoot);
00176 }
00177
00178 // //////////////////////////////////////
00179 void STDAIR_Service::
00180 buildDummyInventory (const CabinCapacity_T& iCabinCapacity) {
00181     // Retrieve the StdAir service context
00182     assert (_stdairServiceContext != NULL);
00183     const STDAIR_ServiceContext& lSTDAIR_ServiceContext = *_stdairServiceContext;
00184
00185     // Retrieve the BOM tree root
00186     BomRoot& lBomRoot = lSTDAIR_ServiceContext.getBomRoot();
00187
00188     // Delegate the building process to the dedicated command
00189     CmdBomManager::buildDummyInventory (lBomRoot, iCabinCapacity);
00190 }
00191
00192 // //////////////////////////////////////
00193 void STDAIR_Service::
00194 buildSampleTravelSolutionForPricing (TravelSolutionList_T& ioTravelSolutionList
00195 ) {
00196     // Build a sample list of travel solution structures
00197     CmdBomManager::buildSampleTravelSolutionForPricing (ioTravelSolutionList);
00198 }
00199
00200 // //////////////////////////////////////

```

```

00198 void STDAIR_Service::
00199 buildSampleTravelSolutions (TravelSolutionList_T& ioTravelSolutionList) {
00200     // Build a sample list of travel solution structures
00201     CmdBomManager::buildSampleTravelSolutions (ioTravelSolutionList);
00202 }
00203
00204 // //////////////////////////////////////
00205 BookingRequestStruct STDAIR_Service::
00206 buildSampleBookingRequest (const bool isForCRS) {
00207     // Build a sample booking request structure
00208     if (isForCRS == true) {
00209         return CmdBomManager::buildSampleBookingRequestForCRS();
00210     }
00211     return CmdBomManager::buildSampleBookingRequest();
00212 }
00213
00214 // //////////////////////////////////////
00215 std::string STDAIR_Service::
00216 jsonExport (const stdair::AirlineCode_T& iAirlineCode,
00217             const stdair::FlightNumber_T& iFlightNumber,
00218             const stdair::Date_T& iDepartureDate) const {
00219     std::ostringstream oStr;
00220
00221     // Retrieve the StdAir service context
00222     assert (_stdairServiceContext != NULL);
00223     const STDAIR_ServiceContext& lSTDAIR_ServiceContext = *_stdairServiceContext;
00224
00225     // Retrieve the BOM tree root
00226     BomRoot& lBomRoot = lSTDAIR_ServiceContext.getBomRoot();
00227
00228     // Retrieve the flight-date object corresponding to the key
00229     FlightDate* lFlightDate_ptr =
00230         BomRetriever::retrieveFlightDateFromKeySet (lBomRoot, iAirlineCode,
00231                                                     iFlightNumber, iDepartureDate);
00232
00233     // Dump the content of the whole BOM tree into the string
00234     if (lFlightDate_ptr != NULL) {
00235         BomJSONExport::jsonExport (oStr, *lFlightDate_ptr);
00236     } else {
00237         #if BOOST_VERSION >= 104100
00238             //
00239             bpt::ptree lPropertyTree;
00240
00241             // Build the appropriate message, so that the client may know that
00242             // no flight-date can be found for that given key.
00243             std::ostringstream oNoFlightDateStream;
00244             oNoFlightDateStream << "No flight-date found for the given key: '"
00245                                 << iAirlineCode << iFlightNumber
00246                                 << " - " << iDepartureDate << "'";
00247             const std::string oNoFlightDateString (oNoFlightDateStream.str());
00248
00249             // Put in the property tree the fact that no flight-date has been found.
00250             // \note That is not (necessary) an error.
00251             lPropertyTree.put ("error", oNoFlightDateString.c_str());
00252
00253             // Write the property tree into the JSON stream.
00254             write_json (oStr, lPropertyTree);
00255         #endif
00256     }
00257 }

```

```

00258 #endif // BOOST_VERSION >= 104100
00259     }
00260
00261     return ostr.str();
00262 }
00263
00264 // //////////////////////////////////////
00265 std::string STDAIR_Service::list (const AirlineCode_T& iAirlineCode,
00266                                 const FlightNumber_T& iFlightNumber) const {
00267     std::ostringstream ostr;
00268
00269     // Retrieve the StdAir service context
00270     assert (_stdairServiceContext != NULL);
00271     const STDAIR_ServiceContext& lSTDAIR_ServiceContext = *_stdairServiceContext;
00272
00273     // Retrieve the BOM tree root
00274     BomRoot& lBomRoot = lSTDAIR_ServiceContext.getBomRoot();
00275
00276     // Dump the content of the whole BOM tree into the string
00277     BomDisplay::list (ostr, lBomRoot, iAirlineCode, iFlightNumber);
00278
00279     return ostr.str();
00280 }
00281
00282 // //////////////////////////////////////
00283 std::string STDAIR_Service::listAirportPairDateRange () const {
00284     std::ostringstream ostr;
00285
00286     // Retrieve the StdAir service context
00287     assert (_stdairServiceContext != NULL);
00288     const STDAIR_ServiceContext& lSTDAIR_ServiceContext = *_stdairServiceContext;
00289
00290     // Retrieve the BOM tree root
00291     BomRoot& lBomRoot = lSTDAIR_ServiceContext.getBomRoot();
00292
00293     // Dump the content of the whole BOM tree into the string
00294     BomDisplay::listAirportPairDateRange (ostr, lBomRoot);
00295
00296     return ostr.str();
00297 }
00298
00299 // //////////////////////////////////////
00300 bool STDAIR_Service::check (const AirlineCode_T& iAirlineCode,
00301                             const FlightNumber_T& iFlightNumber,
00302                             const stdair::Date_T& iDepartureDate) const {
00303     std::ostringstream ostr;
00304
00305     // Retrieve the StdAir service context
00306     assert (_stdairServiceContext != NULL);
00307     const STDAIR_ServiceContext& lSTDAIR_ServiceContext = *_stdairServiceContext;
00308
00309     // Retrieve the BOM tree root
00310     BomRoot& lBomRoot = lSTDAIR_ServiceContext.getBomRoot();
00311
00312     // Dump the content of the whole BOM tree into the string
00313     const FlightDate* lFlightDate_ptr =
00314         BomRetriever::retrieveFlightDateFromKeySet (lBomRoot, iAirlineCode,
00315                                                     iFlightNumber,
00316                                                     iDepartureDate);

```

```

00317
00318     return (lFlightDate_ptr != NULL);
00319 }
00320
00321 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00322 bool STDAIR_Service::check (const stdair::AirportCode_T& ioOrigin,
00323                             const stdair::AirportCode_T& ioDestination,
00324                             const stdair::Date_T& ioDepartureDate) const {
00325     std::ostringstream oStr;
00326
00327     // Retrieve the StdAir service context
00328     assert (_stdairServiceContext != NULL);
00329     const STDAIR_ServiceContext& lSTDAIR_ServiceContext = *_stdairServiceContext;
00330
00331     // Retrieve the BOM tree root
00332     BomRoot& lBomRoot = lSTDAIR_ServiceContext.getBomRoot();
00333
00334     // Dump the content of the whole BOM tree into the string
00335     stdair::DatePeriodList_T lDatePeriodList;
00336     BomRetriever::retrieveDatePeriodListFromKeySet (lBomRoot, ioOrigin,
00337                                                     ioDestination,
00338                                                     ioDepartureDate,
00339                                                     lDatePeriodList);
00340
00341     return (lDatePeriodList.size() != 0);
00342 }
00343
00344 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00345 std::string STDAIR_Service::csvDisplay() const {
00346     std::ostringstream oStr;
00347
00348     // Retrieve the StdAir service context
00349     assert (_stdairServiceContext != NULL);
00350     const STDAIR_ServiceContext& lSTDAIR_ServiceContext = *_stdairServiceContext;
00351
00352     // Retrieve the BOM tree root
00353     BomRoot& lBomRoot = lSTDAIR_ServiceContext.getBomRoot();
00354
00355     // Dump the content of the whole BOM tree into the string
00356     BomDisplay::csvDisplay (oStr, lBomRoot);
00357
00358     return oStr.str();
00359 }
00360
00361 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00362 std::string STDAIR_Service::
00363 csvDisplay (const stdair::AirlineCode_T& iAirlineCode,
00364             const stdair::FlightNumber_T& iFlightNumber,
00365             const stdair::Date_T& iDepartureDate) const {
00366     std::ostringstream oStr;
00367
00368     // Retrieve the StdAir service context
00369     assert (_stdairServiceContext != NULL);
00370     const STDAIR_ServiceContext& lSTDAIR_ServiceContext = *_stdairServiceContext;
00371
00372     // Retrieve the BOM tree root
00373     BomRoot& lBomRoot = lSTDAIR_ServiceContext.getBomRoot();
00374
00375     // Retrieve the flight-date object corresponding to the key

```

```

00376     FlightDate* lFlightDate_ptr =
00377         BomRetriever::retrieveFlightDateFromKeySet (lBomRoot, iAirlineCode,
00378                                                     iFlightNumber, iDepartureDate);

00379
00380     // Dump the content of the whole BOM tree into the string
00381     if (lFlightDate_ptr != NULL) {
00382         BomDisplay::csvDisplay (oStr, *lFlightDate_ptr);
00383     } else {
00384         oStr << "    No flight-date found for the given key: '"
00386             << iAirlineCode << iFlightNumber << " - " << iDepartureDate << "'";
00387     }
00388
00389     return oStr.str();
00390 }
00391
00392 // //////////////////////////////////////
00393 std::string STDAIR_Service::
00394 csvDisplay (const TravelSolutionList_T& iTravelSolutionList) const {
00395
00396     // Dump the content of the whole list of travel solutions into the string
00397     std::ostringstream oStr;
00398     BomDisplay::csvDisplay (oStr, iTravelSolutionList);
00399
00400     return oStr.str();
00401 }
00402
00403 // //////////////////////////////////////
00404 std::string STDAIR_Service::
00405 csvDisplay (const stdair::AirportCode_T& iOrigin,
00406             const stdair::AirportCode_T& iDestination,
00407             const stdair::Date_T& iDepartureDate) const {
00408     std::ostringstream oStr;
00409
00410     // Retrieve the StdAir service context
00411     assert (_stdairServiceContext != NULL);
00412     const STDAIR_ServiceContext& lSTDAIR_ServiceContext = *_stdairServiceContext;

00413
00414     // Retrieve the BOM tree root
00415     BomRoot& lBomRoot = lSTDAIR_ServiceContext.getBomRoot();
00416
00417     // Retrieve the flight-date object corresponding to the key
00418     DatePeriodList_T lDatePeriodList;
00419     BomRetriever::retrieveDatePeriodListFromKeySet (lBomRoot, iOrigin,
00420                                                     iDestination, iDepartureDate,
00421                                                     lDatePeriodList);
00422
00423     // Dump the content of the whole BOM tree into the string
00424     if (lDatePeriodList.empty()) {
00425         oStr << "    No fare-rule found for the given key: '"
00426             << iOrigin << "-" << iDestination << " - " << iDepartureDate << "'";
00427     } else {
00428         BomDisplay::csvDisplay (oStr, lDatePeriodList);
00429     }
00430
00431     return oStr.str();
00432 }
00433
00434 // //////////////////////////////////////

```

```

00435 void STDAIR_Service::finalise() {
00436     // Clean all the objects
00437     FacSupervisor::cleanAll();
00438 }
00439
00440 // //////////////////////////////////////
00441 const Count_T& STDAIR_Service::
00442 getExpectedTotalNumberOfEventsToBeGenerated() const {
00443
00444     // Retrieve the StdAir service context
00445     assert (_stdairServiceContext != NULL);
00446     STDAIR_ServiceContext& lSTDAIR_ServiceContext = *_stdairServiceContext;
00447
00448     // Retrieve the event queue object instance
00449     const EventQueue& lQueue = lSTDAIR_ServiceContext.getEventQueue();
00450
00451     // Delegate the call to the dedicated command
00452     const Count_T& oExpectedTotalNumberOfEventsToBeGenerated =
00453         lQueue.getExpectedTotalNbOfEvents();
00454
00455     //
00456     return oExpectedTotalNumberOfEventsToBeGenerated;
00457 }
00458
00459 // //////////////////////////////////////
00460 const Count_T& STDAIR_Service::
00461 getExpectedTotalNumberOfEventsToBeGenerated (const EventType::EN_EventType& iTy
00462 pe) const {
00463
00464     // Retrieve the StdAir service context
00465     assert (_stdairServiceContext != NULL);
00466     STDAIR_ServiceContext& lSTDAIR_ServiceContext = *_stdairServiceContext;
00467
00468     // Retrieve the event queue object instance
00469     const EventQueue& lQueue = lSTDAIR_ServiceContext.getEventQueue();
00470
00471     // Delegate the call to the dedicated command
00472     const Count_T& oExpectedTotalNumberOfEventsToBeGenerated =
00473         lQueue.getExpectedTotalNbOfEvents (iType);
00474
00475     //
00476     return oExpectedTotalNumberOfEventsToBeGenerated;
00477 }
00478
00479 // //////////////////////////////////////
00480 const Count_T& STDAIR_Service::
00481 getActualTotalNumberOfEventsToBeGenerated() const {
00482
00483     // Retrieve the StdAir service context
00484     assert (_stdairServiceContext != NULL);
00485     STDAIR_ServiceContext& lSTDAIR_ServiceContext = *_stdairServiceContext;
00486
00487     // Retrieve the event queue object instance
00488     const EventQueue& lQueue = lSTDAIR_ServiceContext.getEventQueue();
00489
00490     // Delegate the call to the dedicated command
00491     const Count_T& oActualTotalNumberOfEventsToBeGenerated =
00492         lQueue.getActualTotalNbOfEvents();
00493
00494     //
00495     return oActualTotalNumberOfEventsToBeGenerated;
00496 }

```

```

00496
00497 // //////////////////////////////////////
00498 const Count_T& STDAIR_Service::
00499 getActualTotalNumberOfEventsToBeGenerated (const EventType::EN_EventType& iType
) const {
00500
00501     // Retrieve the StdAir service context
00502     assert (_stdairServiceContext != NULL);
00503     STDAIR_ServiceContext& lSTDAIR_ServiceContext = *_stdairServiceContext;
00504
00505     // Retrieve the event queue object instance
00506     const EventQueue& lQueue = lSTDAIR_ServiceContext.getEventQueue();
00507
00508     // Delegate the call to the dedicated command
00509     const Count_T& oActualTotalNumberOfEventsToBeGenerated =
00510         lQueue.getActualTotalNbOfEvents (iType);
00511
00512     //
00513     return oActualTotalNumberOfEventsToBeGenerated;
00514 }
00515
00516 // //////////////////////////////////////
00517 ProgressStatusSet STDAIR_Service::popEvent (EventStruct& ioEventStruct) const {
00518
00519     // Retrieve the StdAir service context
00520     assert (_stdairServiceContext != NULL);
00521     STDAIR_ServiceContext& lSTDAIR_ServiceContext = *_stdairServiceContext;
00522
00523     // Retrieve the event queue object instance
00524     EventQueue& lQueue = lSTDAIR_ServiceContext.getEventQueue();
00525
00526     // Extract the next event from the queue
00527     return lQueue.popEvent (ioEventStruct);
00528 }
00529
00530 // //////////////////////////////////////
00531 bool STDAIR_Service::isQueueDone() const {
00532
00533     // Retrieve the StdAir service context
00534     assert (_stdairServiceContext != NULL);
00535     STDAIR_ServiceContext& lSTDAIR_ServiceContext = *_stdairServiceContext;
00536
00537     // Retrieve the event queue object instance
00538     const EventQueue& lQueue = lSTDAIR_ServiceContext.getEventQueue();
00539
00540     // Calculates whether the event queue has been fully emptied
00541     const bool isQueueDone = lQueue.isQueueDone();
00542
00543     //
00544     return isQueueDone;
00545 }
00546
00547 // //////////////////////////////////////
00548 void STDAIR_Service::reset() const {
00549
00550     // Retrieve the StdAir service context
00551     assert (_stdairServiceContext != NULL);
00552     STDAIR_ServiceContext& lSTDAIR_ServiceContext = *_stdairServiceContext;
00553
00554     // Retrieve the event queue object instance
00555     EventQueue& lQueue = lSTDAIR_ServiceContext.getEventQueue();

```

```

00556
00557     // Delegate the call to the event queue object
00558     lQueue.reset();
00559 }
00560
00561 }
```

35.523 stdair/service/STDAIR_ServiceContext.cpp File Reference

35.524 STDAIR_ServiceContext.cpp

```

00001
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <cassert>
00010 #include <sstream>
00011 // StdAir
00012 #include <stdair/basic/BasConst_General.hpp>
00013 #include <stdair/bom/BomRoot.hpp>
00014 #include <stdair/bom/EventQueue.hpp>
00015 #include <stdair/factory/FacBom.hpp>
00016 #include <stdair/service/STDAIR_ServiceContext.hpp>
00017
00018 namespace stdair {
00019
00020     // //////////////////////////////////////
00021     STDAIR_ServiceContext::STDAIR_ServiceContext()
00022         : _bomRoot (NULL), _eventQueue (NULL),
00023           _initType (ServiceInitialisationType::NOT_YET_INITIALISED) {
00024         // Build the BomRoot object
00025         init();
00026     }
00027
00028     // //////////////////////////////////////
00029     STDAIR_ServiceContext::
00030     STDAIR_ServiceContext (const STDAIR_ServiceContext& iServiceContext)
00031         : _bomRoot (iServiceContext._bomRoot),
00032           _eventQueue (iServiceContext._eventQueue),
00033           _initType (ServiceInitialisationType::NOT_YET_INITIALISED) {
00034         assert (false);
00035     }
00036
00037     // //////////////////////////////////////
00038     STDAIR_ServiceContext::~STDAIR_ServiceContext() {
00039     }
00040
00041     // //////////////////////////////////////
00042     void STDAIR_ServiceContext::init() {
00043         //
00044         initBomRoot();
00045
00046         //
00047         initEventQueue();
00048     }
00049
00050     // //////////////////////////////////////
00051     void STDAIR_ServiceContext::initBomRoot() {
00052         _bomRoot = &FacBom<BomRoot>::instance().create();
```

```

00053     }
00054
00055     // //////////////////////////////////////
00056     void STDAIR_ServiceContext::initEventQueue() {
00057
00058         // The event queue key is just a string. For now, it is not used.
00059         const EventQueueKey lKey ("EQ01");
00060
00061         // Create an EventQueue object instance
00062         EventQueue& lEventQueue = FacBom<EventQueue>::instance().create (lKey);
00063
00064         // Store the event queue object
00065         _eventQueue = &lEventQueue;
00066     }
00067
00068     // //////////////////////////////////////
00069     const std::string STDAIR_ServiceContext::shortDisplay() const {
00070         std::ostringstream oStr;
00071         oStr << "STDAIR_ServiceContext -- " << _initType
00072             << " -- DB: " << _dbParams;
00073         if (_eventQueue != NULL) {
00074             oStr << " -- Queue: " << _eventQueue->toString();
00075         }
00076         return oStr.str();
00077     }
00078
00079     // //////////////////////////////////////
00080     const std::string STDAIR_ServiceContext::display() const {
00081         std::ostringstream oStr;
00082         oStr << shortDisplay();
00083         return oStr.str();
00084     }
00085
00086     // //////////////////////////////////////
00087     const std::string STDAIR_ServiceContext::describe() const {
00088         return shortDisplay();
00089     }
00090
00091     // //////////////////////////////////////
00092     BomRoot& STDAIR_ServiceContext::getBomRoot() const {
00093         assert (_bomRoot != NULL);
00094         return *_bomRoot;
00095     }
00096
00097     // //////////////////////////////////////
00098     EventQueue& STDAIR_ServiceContext::getEventQueue() const {
00099         assert (_eventQueue != NULL);
00100         return *_eventQueue;
00101     }
00102
00103 }
00104

```

35.525 stdair/service/STDAIR_ServiceContext.hpp File Reference

```

#include <string>

#include <stdair/stdair_basic_types.hpp>

#include <stdair/basic/BasLogParams.hpp>

```

```
#include <stdair/basic/BasDBParams.hpp>
#include <stdair/basic/ServiceInitialisationType.hpp>
#include <stdair/service/ServiceAbstract.hpp>
```

Classes

- class `stdair::STDAIR_ServiceContext`
Class holding the context of the Stdair services.

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

35.526 STDAIR_ServiceContext.hpp

```
00001 #ifndef __STDAIR_SVC_STDAIRSERVICECONTEXT_HPP
00002 #define __STDAIR_SVC_STDAIRSERVICECONTEXT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/stdair_basic_types.hpp>
00011 #include <stdair/basic/BasLogParams.hpp>
00012 #include <stdair/basic/BasDBParams.hpp>
00013 #include <stdair/basic/ServiceInitialisationType.hpp>
00014 #include <stdair/service/ServiceAbstract.hpp>
00015
00016 namespace stdair {
00017
00018     class BomRoot;
00019     class EventQueue;
00020
00021     class STDAIR_ServiceContext : public ServiceAbstract {
00022     friend class STDAIR_Service;
00023     friend class FacSTDAIRServiceContext;
00024
00025     private:
00026         // ////////// Getters //////////
00027         BomRoot& getBomRoot() const;
00028
00029         EventQueue& getEventQueue() const;
00030
00031         const BasDBParams& getDBParams() const {
00032             return _dbParams;
00033         }
00034
00035         const ServiceInitialisationType& getServiceInitialisationType() const {
00036             return _initType;
00037         }
00038     }
00039 }
```

```

00058
00059     private:
00060         // ////////// Setters //////////
00064         void setDBParams (const BasDBParams& iDBParams) {
00065             _dbParams = iDBParams;
00066         }
00067
00071         void setServiceInitialisationType (const ServiceInitialisationType& iSIT) {
00072             _initType = iSIT;
00073         }
00074
00075
00076     private:
00077         // ////////// Display Methods //////////
00081         const std::string shortDisplay() const;
00082
00086         const std::string display() const;
00087
00091         const std::string describe() const;
00092
00093
00094     private:
00095         // ////////// Construction / initialisation //////////
00099         STDAIR_ServiceContext();
00100
00107         STDAIR_ServiceContext (const STDAIR_ServiceContext&);
00108
00112         ~STDAIR_ServiceContext();
00113
00121         void init();
00122
00129         void initBomRoot();
00130
00137         void initEventQueue();
00138
00139
00140     private:
00141         // ////////// Children //////////
00145         BomRoot* _bomRoot;
00146
00150         EventQueue* _eventQueue;
00151
00155         BasDBParams _dbParams;
00156
00170         ServiceInitialisationType _initType;
00171     };
00172
00173 }
00174 #endif // __STDAIR_SVC_STDAIRSERVICECONTEXT_HPP

```

35.527 stdair/stdair_basic_types.hpp File Reference

```
#include <string>
```

```
#include <list>
```

Namespaces

- namespace [stdair](#)

Handle on the StdAir library context.

Typedefs

- typedef std::string [stdair::LocationCode_T](#)
- typedef unsigned long int [stdair::Distance_T](#)
- typedef LocationCode_T [stdair::AirportCode_T](#)
- typedef LocationCode_T [stdair::CityCode_T](#)
- typedef std::string [stdair::KeyDescription_T](#)
- typedef std::string [stdair::AirlineCode_T](#)
- typedef unsigned short [stdair::FlightNumber_T](#)
- typedef unsigned short [stdair::GuillotineNumber_T](#)
- typedef std::string [stdair::CabinCode_T](#)
- typedef std::string [stdair::FamilyCode_T](#)
- typedef std::string [stdair::ClassCode_T](#)
- typedef unsigned long [stdair::Identity_T](#)
- typedef std::string [stdair::TripType_T](#)
- typedef double [stdair::MonetaryValue_T](#)
- typedef double [stdair::RealNumber_T](#)
- typedef double [stdair::Percentage_T](#)
- typedef double [stdair::PriceValue_T](#)
- typedef double [stdair::YieldValue_T](#)
- typedef std::string [stdair::PriceCurrency_T](#)
- typedef double [stdair::Revenue_T](#)
- typedef double [stdair::Multiplier_T](#)
- typedef double [stdair::NbOfSeats_T](#)
- typedef unsigned int [stdair::Count_T](#)
- typedef short [stdair::PartySize_T](#)
- typedef double [stdair::NbOfRequests_T](#)
- typedef NbOfRequests_T [stdair::NbOfBookings_T](#)
- typedef NbOfRequests_T [stdair::NbOfCancellations_T](#)
- typedef unsigned short [stdair::NbOfTravelSolutions_T](#)
- typedef std::string [stdair::ClassList_String_T](#)
- typedef unsigned short [stdair::NbOfSegments_T](#)
- typedef unsigned short [stdair::NbOfAirlines_T](#)
- typedef double [stdair::Availability_T](#)
- typedef double [stdair::Fare_T](#)
- typedef bool [stdair::Flag_T](#)
- typedef unsigned int [stdair::UnsignedIndex_T](#)
- typedef std::string [stdair::Filename_T](#)
- typedef std::string [stdair::FileAddress_T](#)
- typedef float [stdair::ProgressPercentage_T](#)

35.528 stdair_basic_types.hpp

```
00001 #ifndef __STDAIR_STDAIR_BASIC_TYPES_HPP
00002 #define __STDAIR_STDAIR_BASIC_TYPES_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <list>
00010
00011 namespace stdair {
00012
00013 // //////////////////////////////////////
00014 // Basic types
00015 typedef std::string LocationCode_T;
00016
00017
00019 typedef unsigned long int Distance_T;
00020
00022 typedef LocationCode_T AirportCode_T;
00023
00025 typedef LocationCode_T CityCode_T;
00026
00028 typedef std::string KeyDescription_T;
00029
00031 typedef std::string AirlineCode_T;
00032
00034 typedef unsigned short FlightNumber_T;
00035
00037 typedef unsigned short GuillotineNumber_T;
00038
00041 typedef std::string CabinCode_T;
00042
00044 typedef std::string FamilyCode_T;
00045
00048 typedef std::string ClassCode_T;
00049
00051 typedef unsigned long Identity_T;
00052
00055 typedef std::string TripType_T;
00056
00058 typedef double MonetaryValue_T;
00059
00061 typedef double RealNumber_T;
00062
00064 typedef double Percentage_T;
00065
00067 typedef double PriceValue_T;
00068
00070 typedef double YieldValue_T;
00071
00073 typedef std::string PriceCurrency_T;
00074
00076 typedef double Revenue_T;
00077
00079 typedef double Multiplier_T;
00080
00083 typedef double NbOfSeats_T;
00084
00086 typedef unsigned int Count_T;
00087
```

```

00089  typedef short PartySize_T;
00090
00092  typedef double NbOfRequests_T;
00093
00095  typedef NbOfRequests_T NbOfBookings_T;
00096
00098  typedef NbOfRequests_T NbOfCancellations_T;
00099
00102  typedef unsigned short NbOfTravelSolutions_T;
00103
00105  typedef std::string ClassList_String_T;
00106
00108  typedef unsigned short NbOfSegments_T;
00109
00111  typedef unsigned short NbOfAirlines_T;
00112
00114  typedef double Availability_T;
00115
00117  typedef double Fare_T;
00118
00120  typedef bool Flag_T;
00121
00123  typedef unsigned int UnsignedIndex_T;
00124
00125  // ////////////////////////////////// Technical //////////////////////////////////
00129  typedef std::string Filename_T;
00130
00133  typedef std::string FileAddress_T;
00134
00137  typedef float ProgressPercentage_T;
00138
00139 }
00140 #endif // __STDAIR_STDAIR_BASIC_TYPES_HPP

```

35.529 stdair/stdair_date_time_types.hpp File Reference

```

#include <string>
#include <boost/date_time/gregorian/gregorian.hpp>
#include <boost/date_time/posix_time/posix_time.hpp>

```

Namespaces

- namespace `stdair`
Handle on the *StdAir* library context.

Typedefs

- typedef boost::posix_time::time_duration `stdair::Duration_T`
- typedef boost::gregorian::date `stdair::Date_T`
- typedef boost::posix_time::time_duration `stdair::Time_T`
- typedef boost::posix_time::ptime `stdair::DateTime_T`
- typedef boost::gregorian::date_period `stdair::DatePeriod_T`
- typedef std::string `stdair::DOW_String_T`

- typedef boost::gregorian::date_duration [stdair::DateOffset_T](#)
- typedef unsigned int [stdair::DayDuration_T](#)
- typedef bool [stdair::SaturdayStay_T](#)
- typedef long int [stdair::IntDuration_T](#)
- typedef unsigned long long int [stdair::LongDuration_T](#)
- typedef float [stdair::FloatDuration_T](#)

35.530 stdair_date_time_types.hpp

```

00001 #ifndef __STDAIR_STDAIR_DATE_TIME_TYPES_HPP
00002 #define __STDAIR_STDAIR_DATE_TIME_TYPES_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // Boost (Extended STL)
00010 #include <boost/date_time/gregorian/gregorian.hpp>
00011 #include <boost/date_time/posix_time/posix_time.hpp>
00012
00013 namespace stdair {
00014
00015     // ////////// Type definitions //////////
00017     typedef boost::posix_time::time_duration Duration\_T;
00018
00020     typedef boost::gregorian::date Date\_T;
00021
00023     typedef boost::posix_time::time_duration Time\_T;
00024
00026     typedef boost::posix_time::ptime DateTime\_T;
00027
00029     typedef boost::gregorian::date_period DatePeriod\_T;
00030
00032     typedef std::string DOW\_String\_T;
00033
00035     typedef boost::gregorian::date_duration DateOffset\_T;
00036
00038     typedef unsigned int DayDuration\_T;
00039
00041     typedef bool SaturdayStay\_T;
00042
00044     typedef long int IntDuration\_T;
00045
00047     typedef unsigned long long int LongDuration\_T;
00048
00050     typedef float FloatDuration\_T;
00051
00052 }
00053 #endif // __STDAIR_STDAIR_DATE_TIME_TYPES_HPP

```

35.531 stdair/stdair_db.hpp File Reference

```
#include <string>
```

Namespaces

- namespace [soci](#)
- namespace [stdair](#)

Handle on the StdAir library context.

Typedefs

- typedef soci::session [stdair::DBSession_T](#)
- typedef soci::statement [stdair::DBRequestStatement_T](#)
- typedef std::string [stdair::DBConnectionName_T](#)

35.532 stdair_db.hpp

```

00001 #ifndef __STDAIR_STDAIR_DB_HPP
00002 #define __STDAIR_STDAIR_DB_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009
00010 // Forward declarations
00011 namespace soci {
00012     class session;
00013     class statement;
00014 }
00015
00016 namespace stdair {
00017
00018     // ////////// Type definitions //////////
00020     typedef soci::session DBSession_T;
00021
00023     typedef soci::statement DBRequestStatement_T;
00024
00026     typedef std::string DBConnectionName_T;
00027
00028 }
00029 #endif // __STDAIR_STDAIR_DB_HPP

```

35.533 stdair/stdair_demand_types.hpp File Reference

```

#include <string>
#include <vector>
#include <map>
#include <boost/random/linear_congruential.hpp>
#include <boost/random/uniform_real.hpp>
#include <boost/random/variante_generator.hpp>

```

```
#include <boost/date_time/gregorian/gregorian.hpp>
#include <boost/date_time/posix_time/posix_time.hpp>
#include <boost/tuple/tuple.hpp>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/stdair_inventory_types.hpp>
```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

Typedefs

- typedef bool [stdair::ChangeFees_T](#)
- typedef bool [stdair::NonRefundable_T](#)
- typedef unsigned int [stdair::SaturdayStayRatio_T](#)
- typedef unsigned int [stdair::ChangeFeesRatio_T](#)
- typedef unsigned int [stdair::NonRefundableRatio_T](#)
- typedef std::string [stdair::PassengerType_T](#)
- typedef std::string [stdair::DistributionPatternId_T](#)
- typedef std::string [stdair::CancellationRateCurveId_T](#)
- typedef std::string [stdair::AirlinePreferenceId_T](#)
- typedef std::pair< Percentage_T, Percentage_T > [stdair::CancellationNoShowRatePair_T](#)
- typedef std::string [stdair::CharacteristicsPatternId_T](#)
- typedef std::string [stdair::CharacteristicsIndex_T](#)
- typedef double [stdair::WTP_T](#)
- typedef boost::tuples::tuple< double, WTP_T > [stdair::CharacteristicsWTP_tuple_T](#)
- typedef std::pair< WTP_T, MeanStdDevPair_T > [stdair::WTPDemandPair_T](#)
- typedef NbOfRequests_T [stdair::NbOfNoShows_T](#)
- typedef double [stdair::MatchingIndicator_T](#)
- typedef std::string [stdair::DemandStreamKeyStr_T](#)
- typedef std::string [stdair::ChannelLabel_T](#)
- typedef std::string [stdair::FrequentFlyer_T](#)
- typedef std::string [stdair::RequestStatus_T](#)
- typedef std::map< Identity_T, Identity_T > [stdair::BookingTSIDMap_T](#)
- typedef std::pair< CabinCode_T, ClassCode_T > [stdair::CabinClassPair_T](#)
- typedef std::list< CabinClassPair_T > [stdair::CabinClassPairList_T](#)
- typedef double [stdair::ProportionFactor_T](#)
- typedef std::list< ProportionFactor_T > [stdair::ProportionFactorList_T](#)
- typedef std::string [stdair::OnDString_T](#)
- typedef std::list< OnDString_T > [stdair::OnDStringList_T](#)

35.534 stdair_demand_types.hpp

```

00001 #ifndef __STDAIR_STDAIR_DEMAND_TYPES_HPP
00002 #define __STDAIR_STDAIR_DEMAND_TYPES_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010 #include <map>
00011 // Boost Random
00012 #include <boost/random/linear_congruential.hpp>
00013 #include <boost/random/uniform_real.hpp>
00014 #include <boost/random/variante_generator.hpp>
00015 // Boost (Extended STL)
00016 #include <boost/date_time/gregorian/gregorian.hpp>
00017 #include <boost/date_time/posix_time/posix_time.hpp>
00018 #include <boost/tuple/tuple.hpp>
00019 // StdAir
00020 #include <stdair/stdair_basic_types.hpp>
00021 #include <stdair/stdair_maths_types.hpp>
00022 #include <stdair/stdair_inventory_types.hpp>
00023
00024
00025 namespace stdair {
00026
00027     // ////////// Type definitions //////////
00029     typedef bool ChangeFees_T;
00030
00032     typedef bool NonRefundable_T;
00033
00035     typedef bool SaturdayStay_T;
00036
00039     typedef unsigned int SaturdayStayRatio_T;
00040
00043     typedef unsigned int ChangeFeesRatio_T;
00044
00047     typedef unsigned int NonRefundableRatio_T;
00048
00051     typedef std::string PassengerType_T;
00052
00054     typedef std::string DistributionPatternId_T;
00055
00057     typedef std::string CancellationRateCurveId_T;
00058
00060     typedef std::string AirlinePreferenceId_T;
00061
00063     typedef std::pair<Percentage_T, Percentage_T> CancellationNoShowRatePair_T;
00064
00067     typedef std::string CharacteristicsPatternId_T;
00068
00070     typedef std::string CharacteristicsIndex_T;
00071
00073     typedef double WTP_T;
00074
00076     typedef boost::tuples::tuple<double, WTP_T> CharacteristicsWTP_tuple_T;
00077
00079     typedef std::pair<WTP_T, MeanStdDevPair_T> WTPDemandPair_T;
00080
00082     typedef NbOfRequests_T NbOfCancellations_T;

```

```

00083
00085     typedef NbOfRequests_T NbOfNoShows_T;
00086
00088     typedef double MatchingIndicator_T;
00089
00091     typedef std::string DemandStreamKeyStr_T;
00092
00094     typedef std::string ChannelLabel_T;
00095
00097     typedef std::string FrequentFlyer_T;
00098
00101     typedef std::string RequestStatus_T;
00102
00104     typedef std::map<Identity_T, Identity_T> BookingTSIDMap_T;
00105
00107     typedef std::pair<CabinCode_T, ClassCode_T> CabinClassPair_T;
00108
00110     typedef std::list<CabinClassPair_T> CabinClassPairList_T;
00111
00113     typedef double ProportionFactor_T;
00114
00116     typedef std::list<ProportionFactor_T> ProportionFactorList_T;
00117
00119     typedef std::string OnDString_T;
00120
00122     typedef std::list<OnDString_T> OnDStringList_T;
00123
00124 }
00125 #endif // __STDAIR_STDAIR_DEMAND_TYPES_HPP

```

35.535 stdair/stdair_event_types.hpp File Reference

```
#include <string>
```

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

Typedefs

- typedef std::string `stdair::EventName_T`
- typedef double `stdair::NbOfEvents_T`
- typedef std::string `stdair::EventQueueID_T`
- typedef std::string `stdair::EventGeneratorKey_T`

35.536 stdair_event_types.hpp

```

00001 #ifndef __STDAIR_STDAIR_EVENT_TYPES_HPP
00002 #define __STDAIR_STDAIR_EVENT_TYPES_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section

```

```

00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009
00010 namespace stdair {
00011
00012     // ////////// Type definitions //////////
00014     typedef std::string EventName_T;
00015
00017     typedef double NbOfEvents_T;
00018
00020     typedef std::string EventQueueID_T;
00021
00023     typedef std::string EventGeneratorKey_T;
00024
00025 }
00026 #endif // __STDAIR_STDAIR_EVENT_TYPES_HPP

```

35.537 stdair/stdair_exceptions.hpp File Reference

```
#include <string>
```

Classes

- class [stdair::RootException](#)
Root of the stdair exceptions.
- class [stdair::FileNotFoundException](#)
- class [stdair::NonInitialisedLogServiceException](#)
- class [stdair::NonInitialisedServiceException](#)
- class [stdair::NonInitialisedContainerException](#)
- class [stdair::NonInitialisedRelationShipException](#)
- class [stdair::MemoryAllocationException](#)
- class [stdair::ObjectLinkingException](#)
- class [stdair::DocumentNotFoundException](#)
- class [stdair::ParserException](#)
- class [stdair::SerialisationException](#)
- class [stdair::KeyNotFoundException](#)
- class [stdair::CodeConversionException](#)
- class [stdair::CodeDuplicationException](#)
- class [stdair::ObjectCreationDuplicationException](#)
- class [stdair::ObjectNotFoundException](#)
- class [stdair::ParsingFileFailedException](#)
- class [stdair::SQLDatabaseException](#)
- class [stdair::NonInitialisedDBSessionManagerException](#)
- class [stdair::SQLDatabaseConnectionImpossibleException](#)
- class [stdair::EventException](#)
- class [stdair::EventQueueException](#)

Namespaces

- namespace `stdair`

Handle on the StdAir library context.

35.538 stdair_exceptions.hpp

```

00001 #ifndef __STDAIR_STDAIR_EXCEPTIONS_HPP
00002 #define __STDAIR_STDAIR_EXCEPTIONS_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009
00010 namespace stdair {
00011
00012     class RootException : public std::exception {
00013     public:
00014         RootException (const std::string& iWhat) : _what (iWhat) {}
00015         RootException() : _what ("No further details") {}
00016
00017         virtual ~RootException() throw() {}
00018
00019         const char* what() const throw() {
00020             return _what.c_str();
00021         }
00022
00023     protected:
00024         std::string _what;
00025     };
00026
00027     class FileNotFoundException : public RootException {
00028     public:
00029         FileNotFoundException (const std::string& iWhat) : RootException (iWhat) {}
00030     };
00031
00032     class NonInitialisedLogServiceException : public RootException {
00033     public:
00034         NonInitialisedLogServiceException (const std::string& iWhat)
00035             : RootException (iWhat) {}
00036     };
00037
00038     class NonInitialisedServiceException : public RootException {
00039     public:
00040         NonInitialisedServiceException (const std::string& iWhat)
00041             : RootException (iWhat) {}
00042     };
00043
00044     class NonInitialisedContainerException : public RootException {
00045     public:
00046         NonInitialisedContainerException (const std::string& iWhat)
00047             : RootException (iWhat) {}
00048     };
00049
00050     class NonInitialisedRelationShipException : public RootException {
00051     public:
00052         NonInitialisedRelationShipException (const std::string& iWhat)
00053             : RootException (iWhat) {}
00054     };

```

```
00086     };
00087
00089     class MemoryAllocationException : public RootException {
00090     public:
00092         MemoryAllocationException (const std::string& iWhat)
00093             : RootException (iWhat) {}
00094     };
00095
00097     class ObjectLinkingException : public RootException {
00098     public:
00100         ObjectLinkingException (const std::string& iWhat) : RootException (iWhat) {}
00101     };
00102
00104     class DocumentNotFoundException : public RootException {
00105     public:
00107         DocumentNotFoundException (const std::string& iWhat)
00108             : RootException (iWhat) {}
00109     };
00110
00112     class ParserException : public RootException {
00113     public:
00115         ParserException (const std::string& iWhat) : RootException (iWhat) {}
00116     };
00117
00119     class SerialisationException : public RootException {
00120     public:
00122         SerialisationException (const std::string& iWhat) : RootException (iWhat) {}
00123     };
00124
00126     class KeyNotFoundException : public RootException {
00127     public:
00129         KeyNotFoundException (const std::string& iWhat) : RootException (iWhat) {}
00130     };
00131
00133     class CodeConversionException : public ParserException {
00134     public:
00136         CodeConversionException (const std::string& iWhat)
00137             : ParserException (iWhat) {}
00138     };
00139
00141     class CodeDuplicationException : public ParserException {
00142     public:
00144         CodeDuplicationException (const std::string& iWhat)
00145             : ParserException (iWhat) {}
00146     };
00147
00149     class ObjectCreationDuplicationException : public ParserException {
00150     public:
00152         ObjectCreationDuplicationException (const std::string& iWhat)
00153             : ParserException (iWhat) {}
00154     };
00155
00157     class ObjectNotFoundException : public RootException {
00158     public:
00160         ObjectNotFoundException (const std::string& iWhat)
00161             : RootException (iWhat) {}
00162     };
00163
00165     class ParsingFileFailedException : public ParserException {
00166     public:
00168         ParsingFileFailedException (const std::string& iWhat)
00169             : ParserException (iWhat) {}
```

```

00170     };
00171
00173     class SQLiteDatabaseException : public RootException {
00174     public:
00176         SQLiteDatabaseException (const std::string& iWhat) : RootException (iWhat) {}
00177     };
00178
00180     class NonInitialisedDBSessionManagerException : public RootException {
00181     public:
00183         NonInitialisedDBSessionManagerException (const std::string& iWhat)
00184             : RootException (iWhat) {}
00185     };
00186
00188     class SQLiteDatabaseConnectionImpossibleException : public SQLiteDatabaseException {
00189     public:
00191         SQLiteDatabaseConnectionImpossibleException (const std::string& iWhat)
00192             : SQLiteDatabaseException (iWhat) {}
00193     };
00194
00196     class EventException : public RootException {
00197     public:
00199         EventException (const std::string& iWhat) : RootException (iWhat) {}
00200     };
00201
00203     class EventQueueException : public RootException {
00204     public:
00206         EventQueueException (const std::string& iWhat) : RootException (iWhat) {}
00207     };
00208 }
00209 }
00210 #endif // __STDAIR_STDAIR_EXCEPTIONS_HPP

```

35.539 stdair/stdair_fare_types.hpp File Reference

Namespaces

- namespace `stdair`
Handle on the StdAir library context.

Typedefs

- typedef double `stdair::NbOfFareRules_T`

35.540 stdair_fare_types.hpp

```

00001 #ifndef __STDAIR_STDAIR_FARE_TYPES_HPP
00002 #define __STDAIR_STDAIR_FARE_TYPES_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007
00008 namespace stdair {
00009
00010     // ////////// Type definitions //////////

```

```

00012     typedef double NbOfFareRules_T;
00013
00014 }
00015 #endif // __STDAIR_STDAIR_FARE_TYPES_HPP

```

35.541 stdair/stdair_file.hpp File Reference

```

#include <string>
#include <boost/utility.hpp>
#include <stdair/stdair_basic_types.hpp>

```

Classes

- class [stdair::RootFilePath](#)
Root of the input and output files.
- class [stdair::InputFilePath](#)

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.542 stdair_file.hpp

```

00001 #ifndef __STDAIR_STDAIR_FILE_HPP
00002 #define __STDAIR_STDAIR_FILE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // Boost
00010 #include <boost/utility.hpp>
00011 // StdAir
00012 #include <stdair/stdair_basic_types.hpp>
00013
00014 namespace stdair {
00015
00022     class RootFilePath {
00023     public:
00027         RootFilePath (const Filename_T& iFilename) :
00028             _filename (iFilename) {}
00032         RootFilePath () : _filename ("MyFilename") {}
00033
00037         virtual ~RootFilePath() {}
00038
00042         const char * name() const {
00043             return _filename.c_str();
00044         }
00045

```

```

00046     protected:
00050         const Filename_T _filename;
00051     };
00052
00054     class InputFilePath : public RootFilePath {
00055     public:
00057         InputFilePath (const Filename_T& iFilename) :
00058             RootFilePath (iFilename) {}
00059     };
00060
00061 }
00062 #endif // __STDAIR_STDAIR_FILE_HPP

```

35.543 stdair/stdair_inventory_types.hpp File Reference

```

#include <string>
#include <vector>
#include <map>
#include <list>
#include <boost/multi_array.hpp>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/stdair_date_time_types.hpp>

```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

Typedefs

- typedef std::string [stdair::NetworkID_T](#)
- typedef std::vector< AirlineCode_T > [stdair::AirlineCodeList_T](#)
- typedef std::vector< ClassList_String_T > [stdair::ClassList_StringList_T](#)
- typedef std::vector< ClassCode_T > [stdair::ClassCodeList_T](#)
- typedef unsigned short [stdair::SubclassCode_T](#)
- typedef std::string [stdair::FlightPathCode_T](#)
- typedef std::map< CabinCode_T, ClassList_String_T > [stdair::CabinBookingClassMap_T](#)
- typedef double [stdair::CabinCapacity_T](#)
- typedef double [stdair::NbOfFlightDates_T](#)
- typedef double [stdair::CommittedSpace_T](#)
- typedef double [stdair::UPR_T](#)
- typedef double [stdair::BookingLimit_T](#)
- typedef double [stdair::AuthorizationLevel_T](#)

- typedef double [stdair::CapacityAdjustment_T](#)
- typedef double [stdair::BlockSpace_T](#)
- typedef bool [stdair::AvailabilityStatus_T](#)
- typedef std::vector< Availability_T > [stdair::BucketAvailabilities_T](#)
- typedef double [stdair::NbOfYields_T](#)
- typedef double [stdair::NbOfInventoryControlRules_T](#)
- typedef bool [stdair::CensorshipFlag_T](#)
- typedef short [stdair::DTD_T](#)
- typedef short [stdair::DCP_T](#)
- typedef std::list< DCP_T > [stdair::DCPList_T](#)
- typedef std::map< DTD_T, RealNumber_T > [stdair::DTDFratMap_T](#)
- typedef std::map< FloatDuration_T, float > [stdair::DTDProbMap_T](#)
- typedef std::vector< CensorshipFlag_T > [stdair::CensorshipFlagList_T](#)
- typedef double [stdair::BookingRatio_T](#)
- typedef double [stdair::Yield_T](#)
- typedef unsigned int [stdair::YieldLevel_T](#)
- typedef std::map< YieldLevel_T, MeanStdDevPair_T > [stdair::YieldLevelDemandMap_T](#)
- typedef std::pair< Yield_T, MeanStdDevPair_T > [stdair::YieldDemandPair_T](#)
- typedef double [stdair::BidPrice_T](#)
- typedef std::vector< BidPrice_T > [stdair::BidPriceVector_T](#)
- typedef unsigned int [stdair::SeatIndex_T](#)
- typedef std::string [stdair::ControlMode_T](#)
- typedef double [stdair::OverbookingRate_T](#)
- typedef double [stdair::ProtectionLevel_T](#)
- typedef std::vector< double > [stdair::EmsrValueList_T](#)
- typedef std::vector< double > [stdair::BookingLimitVector_T](#)
- typedef std::vector< double > [stdair::ProtectionLevelVector_T](#)
- typedef boost::multi_array< double, 2 > [stdair::SnapshotBlock_T](#)
- typedef SnapshotBlock_T::index_range [stdair::SnapshotBlockRange_T](#)
- typedef SnapshotBlock_T::array_view< 1 >::type [stdair::SegmentCabinDTDSnapshotView_T](#)
- typedef SnapshotBlock_T::array_view< 2 >::type [stdair::SegmentCabinDTDRangeSnapshotView_T](#)
- typedef SnapshotBlock_T::const_array_view< 1 >::type [stdair::ConstSegmentCabinDTDSnapshotView_T](#)
- typedef SnapshotBlock_T::const_array_view< 2 >::type [stdair::ConstSegmentCabinDTDRangeSnapshotView_T](#)
- typedef unsigned short [stdair::BlockNumber_T](#)
- typedef unsigned short [stdair::BlockIndex_T](#)

35.544 stdair_inventory_types.hpp

```
00001 #ifndef __STDAIR_STDAIR_INVENTORY_TYPES_HPP
00002 #define __STDAIR_STDAIR_INVENTORY_TYPES_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010 #include <map>
00011 #include <list>
00012 // BOOST
00013 #include <boost/multi_array.hpp>
00014 // StdAir
00015 #include <stdair/stdair_basic_types.hpp>
00016 #include <stdair/stdair_maths_types.hpp>
00017 #include <stdair/stdair_date_time_types.hpp>
00018
00019 namespace stdair {
00020
00021 // ////////// Type definitions //////////
00023 typedef std::string NetworkID_T;
00024
00026 typedef std::vector<AirlineCode_T> AirlineCodeList_T;
00027
00029 typedef std::vector<ClassList_String_T> ClassList_StringList_T;
00030
00032 typedef std::vector<ClassCode_T> ClassCodeList_T;
00033
00037 typedef unsigned short SubclassCode_T;
00038
00040 typedef std::string FlightPathCode_T;
00041
00044 typedef std::map<CabinCode_T, ClassList_String_T> CabinBookingClassMap_T;
00045
00048 typedef double CabinCapacity_T;
00049
00051 typedef double NbOfFlightDates_T;
00052
00054 typedef double CommittedSpace_T;
00055
00057 typedef double UPR_T;
00058
00060 typedef double BookingLimit_T;
00061
00063 typedef double AuthorizationLevel_T;
00064
00066 typedef double CapacityAdjustment_T;
00067
00069 typedef double BlockSpace_T;
00070
00072 typedef bool AvailabilityStatus_T;
00073
00075 typedef std::vector<Availability_T> BucketAvailabilities_T;
00076
00078 typedef double NbOfYields_T;
00079
00081 typedef double NbOfInventoryControlRules_T;
00082
00084 typedef bool CensorshipFlag_T;
```

```
00085
00087     typedef short DTD_T;
00088
00090     typedef short DCP_T;
00091
00093     typedef std::list<DCP_T> DCPList_T;
00094
00096     typedef std::map<DTD_T, RealNumber_T> DTDFratMap_T;
00097
00099     typedef std::map<FloatDuration_T, float> DTDPProbMap_T;
00100
00103     typedef std::vector<CensorshipFlag_T> CensorshipFlagList_T;
00104
00107     typedef double BookingRatio_T;
00108
00110     typedef double Yield_T;
00111
00113     typedef unsigned int YieldLevel_T;
00114
00116     typedef std::map<YieldLevel_T, MeanStdDevPair_T> YieldLevelDemandMap_T;
00117
00119     typedef std::pair<Yield_T, MeanStdDevPair_T> YieldDemandPair_T;
00120
00122     typedef double BidPrice_T;
00123
00125     typedef std::vector<BidPrice_T> BidPriceVector_T;
00126
00128     typedef unsigned int SeatIndex_T;
00129
00131     typedef std::string ControlMode_T;
00132
00134     typedef double OverbookingRate_T;
00135
00138     typedef double BookingLimit_T;
00139
00142     typedef double ProtectionLevel_T;
00143
00145     typedef std::vector<double> EmsrValueList_T;
00146
00149     typedef std::vector<double> BookingLimitVector_T;
00150
00153     typedef std::vector<double> ProtectionLevelVector_T;
00154
00156     typedef boost::multi_array<double, 2> SnapshotBlock_T;
00157
00159     typedef SnapshotBlock_T::index_range SnapshotBlockRange_T;
00160
00162     typedef SnapshotBlock_T::array_view<1>::type SegmentCabinDTDSnapshotView_T;
00163
00165     typedef SnapshotBlock_T::array_view<2>::type
SegmentCabinDTDRangeSnapshotView_T;
00166
00168     typedef SnapshotBlock_T::const_array_view<1>::type
ConstSegmentCabinDTDSnapshotView_T;
00169
00171     typedef SnapshotBlock_T::const_array_view<2>::type
ConstSegmentCabinDTDRangeSnapshotView_T;
00172
00174     typedef unsigned short BlockNumber_T;
00175
00177     typedef unsigned short BlockIndex_T;
00178
```

```
00179 }
00180 #endif // __STDAIR_STDAIR_INVENTORY_TYPES_HPP
```

35.545 stdair/stdair_log.hpp File Reference

```
#include <string>
```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.
- namespace [stdair::LOG](#)

Enumerations

- enum [stdair::LOG::EN_LogLevel](#) {
[stdair::LOG::CRITICAL](#) = 0, [stdair::LOG::ERROR](#), [stdair::LOG::NOTIFICATION](#),
[stdair::LOG::WARNING](#),
[stdair::LOG::DEBUG](#), [stdair::LOG::VERBOSE](#), [stdair::LOG::LAST_VALUE](#) }

Variables

- static const std::string [stdair::LOG::_logLevels](#) [LAST_VALUE]

35.546 stdair_log.hpp

```
00001 #ifndef __STDAIR_STDAIR_LOG_HPP
00002 #define __STDAIR_STDAIR_LOG_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009
00010 namespace stdair {
00011
00012     // Forward declarations
00013     class STDAIR_Service;
00014
00015     // ////////////////////////////////// Log //////////////////////////////////
00017     namespace LOG {
00018         typedef enum {
00019             CRITICAL = 0,
00020             ERROR,
00021             NOTIFICATION,
00022             WARNING,
00023             DEBUG,
00024             VERBOSE,
00025             LAST_VALUE
```

```

00026     } EN_LogLevel;
00027
00028     static const std::string _logLevels[LAST_VALUE] =
00029     {"C", "E", "N", "W", "D", "V"};
00030 }
00031
00032 }
00033 #endif // __STDAIR_STDAIR_LOG_HPP

```

35.547 stdair/stdair_maths_types.hpp File Reference

```

#include <string>
#include <vector>
#include <map>
#include <boost/random/linear_congruential.hpp>
#include <boost/random/uniform_real.hpp>
#include <boost/random/normal_distribution.hpp>
#include <boost/random/exponential_distribution.hpp>
#include <boost/random/variante_generator.hpp>

```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

Typedefs

- typedef unsigned int [stdair::ReplicationNumber_T](#)
- typedef unsigned long int [stdair::ExponentialSeed_T](#)
- typedef unsigned long int [stdair::UniformSeed_T](#)
- typedef unsigned long int [stdair::RandomSeed_T](#)
- typedef boost::minstd_rand [stdair::BaseGenerator_T](#)
- typedef boost::uniform_real [stdair::UniformDistribution_T](#)
- typedef boost::variante_generator< BaseGenerator_T &, UniformDistribution_T > [stdair::UniformGenerator_T](#)
- typedef boost::normal_distribution [stdair::NormalDistribution_T](#)
- typedef boost::variante_generator< BaseGenerator_T &, NormalDistribution_T > [stdair::NormalGenerator_T](#)
- typedef boost::exponential_distribution [stdair::ExponentialDistribution_T](#)
- typedef boost::variante_generator< BaseGenerator_T &, ExponentialDistribution_T > [stdair::ExponentialGenerator_T](#)
- typedef double [stdair::MeanValue_T](#)
- typedef double [stdair::StdDevValue_T](#)
- typedef std::pair< MeanValue_T, StdDevValue_T > [stdair::MeanStdDevPair_T](#)
- typedef float [stdair::Probability_T](#)

35.548 stdair_maths_types.hpp

```

00001 #ifndef __STDAIR_STDAIR_MATHS_TYPES_HPP
00002 #define __STDAIR_STDAIR_MATHS_TYPES_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010 #include <map>
00011 // Boost Random
00012 #include <boost/random/linear_congruential.hpp>
00013 #include <boost/random/uniform_real.hpp>
00014 #include <boost/random/normal_distribution.hpp>
00015 #include <boost/random/exponential_distribution.hpp>
00016 #include <boost/random/variante_generator.hpp>
00017
00018 namespace stdair {
00019
00020     // ////////// Type definitions //////////
00024     typedef unsigned int ReplicationNumber_T;
00025
00029     typedef unsigned long int ExponentialSeed_T;
00030
00034     typedef unsigned long int UniformSeed_T;
00035
00039     typedef unsigned long int RandomSeed_T;
00040
00044     typedef boost::minstd_rand BaseGenerator_T;
00045
00049     typedef boost::uniform_real<> UniformDistribution_T;
00050
00054     typedef boost::variante_generator<BaseGenerator_T&,
00055                                     UniformDistribution_T> UniformGenerator_T;
00056
00060     typedef boost::normal_distribution<> NormalDistribution_T;
00061
00065     typedef boost::variante_generator<BaseGenerator_T&,
00066                                     NormalDistribution_T> NormalGenerator_T;
00067
00069     typedef boost::exponential_distribution<> ExponentialDistribution_T;
00070
00071
00073     typedef boost::variante_generator<BaseGenerator_T&,
00074                                     ExponentialDistribution_T>
00075     ExponentialGenerator_T;
00076
00079     typedef double MeanValue_T;
00080
00084     typedef double StdDevValue_T;
00085
00089     typedef std::pair<MeanValue_T, StdDevValue_T> MeanStdDevPair_T;
00090
00094     typedef float Probability_T;
00095
00096 }
00097 #endif // __STDAIR_STDAIR_MATHS_TYPES_HPP

```

35.549 stdair/stdair_rm_types.hpp File Reference

```
#include <string>
#include <vector>
#include <stdair/stdair_basic_types.hpp>
```

Namespaces

- namespace `stdair`
Handle on the `StdAir` library context.

Typedefs

- typedef `std::string` `stdair::ForecasterMode_T`
- typedef `short` `stdair::HistoricalDataLimit_T`
- typedef `std::string` `stdair::OptimizerMode_T`
- typedef `NbOfBookings_T` `stdair::PolicyDemand_T`
- typedef `std::vector< double >` `stdair::GeneratedDemandVector_T`
- typedef `std::vector< GeneratedDemandVector_T >` `stdair::GeneratedDemandVectorHolder_T`
- typedef `double` `stdair::SellupProbability_T`
- typedef `std::vector< SellupProbability_T >` `stdair::SellupProbabilityVector_T`
- typedef `std::vector< double >` `stdair::SellupFactorHolder_T`

35.550 stdair_rm_types.hpp

```
00001 #ifndef __STDAIR_STDAIR_RM_TYPES_HPP
00002 #define __STDAIR_STDAIR_RM_TYPES_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010 // StdAir
00011 #include <stdair/stdair_basic_types.hpp>
00012
00013 namespace stdair {
00014
00015     // ////////// Type definitions //////////
00017     typedef std::string ForecasterMode_T;
00018
00020     typedef short HistoricalDataLimit_T;
00021
00023     typedef std::string OptimizerMode_T;
00024
00026     typedef NbOfBookings_T PolicyDemand_T;
00027
00030     typedef std::vector<double> GeneratedDemandVector_T;
00031
```

```

00033     typedef std::vector<GeneratedDemandVector_T> GeneratedDemandVectorHolder_T;
00034
00036     typedef double SellupProbability_T;
00037
00039     typedef std::vector<SellupProbability_T> SellupProbabilityVector_T;
00040
00042     typedef std::vector<double> SellupFactorHolder_T;
00043
00044 }
00045 #endif // __STDAIR_STDAIR_RM_TYPES_HPP

```

35.551 stdair/STDAIR_Service.hpp File Reference

```

#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_service_types.hpp>
#include <stdair/basic/BasLogParams.hpp>
#include <stdair/basic/BasDBParams.hpp>
#include <stdair/basic/ServiceInitialisationType.hpp>
#include <stdair/basic/EventType.hpp>
#include <stdair/bom/TravelSolutionTypes.hpp>

```

Classes

- class [stdair::STDAIR_Service](#)
Interface for the STDAIR Services.

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

35.552 STDAIR_Service.hpp

```

00001
00012 #ifndef __STDAIR_STDAIR_HPP
00013 #define __STDAIR_STDAIR_HPP
00014
00015 // //////////////////////////////////////
00016 // Import section
00017 // //////////////////////////////////////
00018 // STL
00019 #include <string>
00020 // StdAir
00021 #include <stdair/stdair_inventory_types.hpp>
00022 #include <stdair/stdair_service_types.hpp>
00023 #include <stdair/basic/BasLogParams.hpp>

```

```

00024 #include <stdair/basic/BasDBParams.hpp>
00025 #include <stdair/basic/ServiceInitialisationType.hpp>
00026 #include <stdair/basic/EventType.hpp>
00027 #include <stdair/bom/TravelSolutionTypes.hpp>
00028
00029 namespace stdair {
00030
00032     class BomRoot;
00033     class EventQueue;
00034     class EventStruct;
00035     struct ProgressStatusSet;
00036     struct BookingRequestStruct;
00037     class STDAIR_ServiceContext;
00038
00039
00043     class STDAIR_Service {
00044     public:
00045         // ////////// Constructors and destructors //////////
00049         STDAIR_Service();
00050
00062         STDAIR_Service (const BasLogParams&);
00063
00079         STDAIR_Service (const BasLogParams&, const BasDBParams&);
00080
00084         ~STDAIR_Service();
00085
00086
00087     public:
00088         // ////////////////// Business support methods //////////////////
00108         void buildSampleBom();
00109
00131         void buildDummyInventory (const CabinCapacity_T& iCabinCapacity);
00132
00147         void buildSampleTravelSolutionForPricing (TravelSolutionList_T&);
00148
00166         void buildSampleTravelSolutions (TravelSolutionList_T&);
00167
00196         BookingRequestStruct buildSampleBookingRequest (const bool isForCRS = false);
00197
00212         const Count_T& getExpectedTotalNumberOfEventsToBeGenerated() const;
00213
00230         const Count_T&
00231         getExpectedTotalNumberOfEventsToBeGenerated (const EventType::EN_EventType&)
00232         const;
00245         const Count_T& getActualTotalNumberOfEventsToBeGenerated() const;
00246
00260         const Count_T&
00261         getActualTotalNumberOfEventsToBeGenerated (const EventType::EN_EventType&) co
00262         nst;
00279         ProgressStatusSet popEvent (EventStruct&) const;
00280
00286         bool isQueueDone() const;
00287
00292         void reset() const;
00293
00294
00295     public:
00296         // ////////////////// Export support methods //////////////////
00306         std::string jsonExport (const AirlineCode_T&, const FlightNumber_T&,

```

```
00307         const Date_T& iDepartureDate) const;
00308
00309
00310     public:
00311         // //////////// Display support methods ////////////
00312         std::string list (const AirlineCode_T& iAirlineCode = "all",
00326             const FlightNumber_T& iFlightNumber = 0) const;
00327
00334         std::string listAirportPairDateRange () const;
00335
00345         bool check (const AirlineCode_T&, const FlightNumber_T&,
00346             const Date_T& iDepartureDate) const;
00347
00360         bool check (const AirportCode_T&, const AirportCode_T&,
00361             const Date_T& iDepartureDate) const;
00362
00370         std::string csvDisplay () const;
00371
00381         std::string csvDisplay (const AirlineCode_T&, const FlightNumber_T&,
00382             const Date_T& iDepartureDate) const;
00383
00391         std::string csvDisplay (const TravelSolutionList_T&) const;
00392
00403         std::string csvDisplay (const AirportCode_T&, const AirportCode_T&,
00404             const Date_T& iDepartureDate) const;
00405
00406
00407     public:
00408         // //////////// Getters ////////////
00417         BomRoot& getBomRoot () const;
00418
00427         EventQueue& getEventQueue () const;
00428
00434         BasLogParams getLogParams () const;
00435
00442         const BasDBParams& getDBParams () const;
00443
00452         const ServiceInitialisationType& getServiceInitialisationType () const;
00453
00454
00455     private:
00456         // //////////// Construction and Destruction helper methods ////////////
00463         STDAIR_Service (const STDAIR_Service&);
00464
00469         void initServiceContext ();
00470
00488         void logInit (const BasLogParams&);
00489
00495         void dbInit (const BasDBParams&);
00496
00512         void init ();
00513
00517         void finalise ();
00518
00519
00520     private:
00521         // //////////// Service Context ////////////
00525         STDAIR_ServiceContext* _stdairServiceContext;
00526     };
00527 }
00528 #endif // __STDAIR_STDAIR_HPP
```

35.553 stdair/stdair_service_types.hpp File Reference

```
#include <boost/shared_ptr.hpp>
```

Namespaces

- namespace [stdair](#)
Handle on the StdAir library context.

Typedefs

- typedef boost::shared_ptr< STDAIR_Service > [stdair::STDAIR_ServicePtr_T](#)

35.554 stdair_service_types.hpp

```
00001 #ifndef __STDAIR_STDAIR_SERVICE_HPP
00002 #define __STDAIR_STDAIR_SERVICE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // Boost (Extended STL)
00008 #include <boost/shared_ptr.hpp>
00009
00010 namespace stdair {
00011
00012     // Forward declarations
00013     class STDAIR_Service;
00014
00016     typedef boost::shared_ptr<STDAIR_Service> STDAIR_ServicePtr_T;
00017
00018 }
00019 #endif // __STDAIR_STDAIR_SERVICE_HPP
```

35.555 stdair/stdair_types.hpp File Reference

```
#include <stdair/stdair_exceptions.hpp>
#include <stdair/stdair_log.hpp>
#include <stdair/stdair_db.hpp>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/stdair_fare_types.hpp>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_rm_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
```

```
#include <stdair/stdair_service_types.hpp>
```

35.556 stdair_types.hpp

```
00001 #ifndef __STDAIR_STDAIR_TYPES_HPP
00002 #define __STDAIR_STDAIR_TYPES_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_exceptions.hpp>
00009 #include <stdair/stdair_log.hpp>
00010 #include <stdair/stdair_db.hpp>
00011 #include <stdair/stdair_basic_types.hpp>
00012 #include <stdair/stdair_demand_types.hpp>
00013 #include <stdair/stdair_maths_types.hpp>
00014 #include <stdair/stdair_fare_types.hpp>
00015 #include <stdair/stdair_inventory_types.hpp>
00016 #include <stdair/stdair_rm_types.hpp>
00017 #include <stdair/stdair_date_time_types.hpp>
00018 #include <stdair/stdair_service_types.hpp>
00019
00020 #endif // __STDAIR_STDAIR_TYPES_HPP
```

35.557 stdair/ui/cmdline/readline_autocomp.hpp File Reference

```
#include <string>
#include <iosfwd>
#include <cstdio>
#include <sys/types.h>
#include <sys/file.h>
#include <sys/stat.h>
#include <sys/errno.h>
#include <readline/readline.h>
#include <readline/history.h>
```

Classes

- struct [COMMAND](#)

Typedefs

- typedef int(* [pt2Func](#))(char *)

Functions

- char * [getwd](#) ()
- char * [xmalloc](#) (size_t)
- int [com_list](#) (char *)
- int [com_view](#) (char *)
- int [com_rename](#) (char *)
- int [com_stat](#) (char *)
- int [com_pwd](#) (char *)
- int [com_delete](#) (char *)
- int [com_help](#) (char *)
- int [com_cd](#) (char *)
- int [com_quit](#) (char *)
- char * [stripwhite](#) (char *iString)
- COMMAND * [find_command](#) (char *iString)
- char * [dupstr](#) (char *iString)
- int [execute_line](#) (char *line)
- char * [command_generator](#) (char *text, int state)
- char ** [fileman_completion](#) (char *text, int start, int end)
- void [initialize_readline](#) ()
- void [too_dangerous](#) (char *caller)
- int [valid_argument](#) (char *caller, char *arg)

Variables

- COMMAND [commands](#) []
- int [done](#)
- static char [syscom](#) [1024]

35.557.1 Typedef Documentation

35.557.1.1 typedef int(* pt2Func)(char *)

Definition at line 35 of file [readline_autocomp.hpp](#).

35.557.2 Function Documentation

35.557.2.1 char* getwd ()

[readline_autocomp.hpp](#) -- A tiny application which demonstrates how to use the GNU Readline library. This application interactively allows users to manipulate files and their modes.

Referenced by [com_pwd\(\)](#).

35.557.2.2 char* xmalloc (size_t)

Referenced by [dupstr\(\)](#).

35.557.2.3 void com_list (char * *arg*)

List the file(s) named in *arg*.

Definition at line 264 of file [readline_autocomp.hpp](#).

35.557.2.4 int com_view (char * *arg*)

Definition at line 274 of file [readline_autocomp.hpp](#).

References [valid_argument\(\)](#).

35.557.2.5 int com_rename (char * *arg*)

Definition at line 284 of file [readline_autocomp.hpp](#).

References [too_dangerous\(\)](#).

35.557.2.6 int com_stat (char * *arg*)

Definition at line 289 of file [readline_autocomp.hpp](#).

References [valid_argument\(\)](#).

35.557.2.7 int com_pwd (char * *ignore*)

Definition at line 367 of file [readline_autocomp.hpp](#).

References [getwd\(\)](#).

Referenced by [com_cd\(\)](#).

35.557.2.8 int com_delete (char * *arg*)

Definition at line 315 of file [readline_autocomp.hpp](#).

References [too_dangerous\(\)](#).

35.557.2.9 int com_help (char * *arg*)

Print out help for ARG, or for all of the commands if ARG is not present.

Definition at line 324 of file [readline_autocomp.hpp](#).

References [COMMAND::name](#).

35.557.2.10 int com_cd (char * *arg*)

Definition at line 356 of file [readline_autocomp.hpp](#).

References [com_pwd\(\)](#).

35.557.2.11 int com_quit (char * *arg*)

Definition at line 381 of file [readline_autocomp.hpp](#).

35.557.2.12 `char * stripwhite (char * string)`

Strip whitespace from the start and end of STRING. Return a pointer into STRING.

Definition at line 152 of file [readline_autocomp.hpp](#).

35.557.2.13 `COMMAND * find_command (char * name)`

Look up NAME as the name of a command, and return a pointer to that command. Return a NULL pointer if NAME isn't a command name.

Definition at line 136 of file [readline_autocomp.hpp](#).

References [COMMAND::name](#).

Referenced by [execute_line\(\)](#).

35.557.2.14 `char* dupstr (char * iString)`

Duplicate a string

Definition at line 85 of file [readline_autocomp.hpp](#).

References [xmalloc\(\)](#).

Referenced by [command_generator\(\)](#).

35.557.2.15 `int execute_line (char * line)`

Execute a command line.

Definition at line 94 of file [readline_autocomp.hpp](#).

References [find_command\(\)](#), and [COMMAND::func](#).

35.557.2.16 `char * command_generator (char * text, int state)`

Generator function for command completion. STATE lets us know whether to start from scratch; without any state (i.e. STATE == 0), then we start at the top of the list.

Definition at line 222 of file [readline_autocomp.hpp](#).

References [dupstr\(\)](#).

Referenced by [fileman_completion\(\)](#).

35.557.2.17 `char ** fileman_completion (char * text, int start, int end)`

Attempt to complete on the contents of TEXT. START and END bound the region of rl_line_buffer that contains the word to complete. TEXT is the word to complete. We can use the entire contents of rl_line_buffer in case we want to do some simple parsing. Return the array of matches, or NULL if there aren't any.

Definition at line 200 of file [readline_autocomp.hpp](#).

References [command_generator\(\)](#).

Referenced by [initialize_readline\(\)](#).

35.557.2.18 void initialize_readline ()

Tell the GNU Readline library how to complete. We want to try to complete on command names if this is the first word in the line, or on filenames if not.

Definition at line 185 of file [readline_autocomp.hpp](#).

References [fileman_completion\(\)](#).

35.557.2.19 void too_dangerous (char * caller)

Definition at line 387 of file [readline_autocomp.hpp](#).

Referenced by [com_delete\(\)](#), and [com_rename\(\)](#).

35.557.2.20 int valid_argument (char * caller, char * arg)

Definition at line 395 of file [readline_autocomp.hpp](#).

Referenced by [com_stat\(\)](#), and [com_view\(\)](#).

35.557.3 Variable Documentation**35.557.3.1 COMMAND commands[]**

Initial value:

```
{
  { "cd", (*com_cd)(), "Change to directory DIR" },
  { "delete", com_delete, "Delete FILE" },
  { "help", com_help, "Display this text" },
  { "?", com_help, "Synonym for 'help'" },
  { "list", com_list, "List files in DIR" },
  { "ls", com_list, "Synonym for 'list'" },
  { "pwd", com_pwd, "Print the current working directory" },
  { "quit", com_quit, "Quit using airinv" },
  { "rename", com_rename, "Rename FILE to NEWNAME" },
  { "stat", com_stat, "Print out statistics on FILE" },
  { "view", com_view, "View the contents of FILE" },
  { (char*) NULL, (pt2Func) NULL, (char*) NULL }
}
```

Definition at line 58 of file [readline_autocomp.hpp](#).

35.557.3.2 int done

When non-zero, this global means the user is done using this program.

Definition at line 80 of file [readline_autocomp.hpp](#).

35.557.3.3 char syscom[1024] [static]

String to pass to system(). This is for the LIST, VIEW and RENAME commands.

Definition at line 259 of file [readline_autocomp.hpp](#).

35.558 readline_autocomp.hpp

```
00001
00006 #ifndef __AIRINV_READLINE_AUTOCOMP_HPP
00007 #define __AIRINV_READLINE_AUTOCOMP_HPP
00008
00009 // STL
00010 #include <string>
00011 #include <iosfwd>
00012 #include <cstdio>
00013 #include <sys/types.h>
00014 #include <sys/file.h>
00015 #include <sys/stat.h>
00016 #include <sys/errno.h>
00017
00018 #include <readline/readline.h>
00019 #include <readline/history.h>
00020
00021 extern char* getwd();
00022 extern char* xmalloc (size_t);
00023
00024 /* The names of functions that actually do the manipulation. */
00025 int com_list (char*);
00026 int com_view (char*);
00027 int com_rename (char*);
00028 int com_stat (char*);
00029 int com_pwd (char*);
00030 int com_delete (char*);
00031 int com_help (char*);
00032 int com_cd (char*);
00033 int com_quit (char*);
00034
00035 typedef int (*pt2Func) (char*);
00036
00041 typedef struct {
00045     char const* name;
00046
00050     pt2Func *func;
00051
00055     char *doc;
00056 } COMMAND;
00057
00058 COMMAND commands[] = {
00059     { "cd", (*com_cd)(), "Change to directory DIR" },
00060     { "delete", com_delete, "Delete FILE" },
00061     { "help", com_help, "Display this text" },
00062     { "?", com_help, "Synonym for 'help'" },
00063     { "list", com_list, "List files in DIR" },
00064     { "ls", com_list, "Synonym for 'list'" },
00065     { "pwd", com_pwd, "Print the current working directory" },
00066     { "quit", com_quit, "Quit using airinv" },
00067     { "rename", com_rename, "Rename FILE to NEWNAME" },
00068     { "stat", com_stat, "Print out statistics on FILE" },
00069     { "view", com_view, "View the contents of FILE" },
00070     { (char*) NULL, (pt2Func) NULL, (char*) NULL }
00071 };
00072
00073 // Forward declarations
00074 char* stripwhite (char* iString);
00075 COMMAND* find_command (char* iString);
00076
00080 int done;
```

```
00081
00085 char* dupstr (char* iString) {
00086     char* r = xmalloc (std::strlen (iString) + 1);
00087     strcpy (r, iString);
00088     return r;
00089 }
00090
00094 int execute_line (char* line) {
00095     register int i;
00096     COMMAND* command;
00097     char* word;
00098
00099     /* Isolate the command word. */
00100     i = 0;
00101     while (line[i] && whitespace (line[i])) {
00102         i++;
00103     }
00104     word = line + i;
00105
00106     while (line[i] && !whitespace (line[i])) {
00107         i++;
00108     }
00109
00110     if (line[i]) {
00111         line[i++] = '\0';
00112     }
00113
00114     command = find_command (word);
00115
00116     if (!command) {
00117         std::cerr << word << ": No such command for airinv." << std::endl;
00118         return -1;
00119     }
00120
00121     /* Get argument to command, if any. */
00122     while (whitespace (line[i])) {
00123         i++;
00124     }
00125
00126     word = line + i;
00127
00128     /* Call the function. */
00129     return (*(command->func)) (word);
00130 }
00131
00136 COMMAND* find_command (char* name) {
00137     register int i;
00138
00139     for (i = 0; commands[i].name; i++) {
00140         if (strcmp (name, commands[i].name) == 0) {
00141             return (&commands[i]);
00142         }
00143     }
00144
00145     return (COMMAND*) NULL;
00146 }
00147
00152 char* stripwhite (char* string) {
00153     register char *s, *t;
00154
00155     for (s = string; whitespace (*s); s++) {
00156     }
```

```
00157
00158     if (*s == 0) {
00159         return s;
00160     }
00161
00162     t = s + strlen (s) - 1;
00163     while (t > s && whitespace (*t)) {
00164         t--;
00165     }
00166     *++t = '\0';
00167
00168     return s;
00169 }
00170
00171 /* ***** */
00172 /*
00173                      Interface to Readline Completion
00174 /*
00175 /* ***** */
00176
00177 char* command_generator (char* text, int state);
00178 char** fileman_completion (char* text, int start, int end);
00179
00180 void initialize_readline() {
00181     /* Allow conditional parsing of the ~/.inputrc file. */
00182     rl_readline_name = "airinv";
00183
00184     /* Tell the completer that we want a crack first. */
00185     rl_attempted_completion_function = (rl_completion_func_t*) fileman_completion;
00186 }
00187
00188 char** fileman_completion (char* text, int start, int end) {
00189     char **matches;
00190
00191     matches = (char**) NULL;
00192
00193     if (start == 0) {
00194         matches = completion_matches (text, command_generator);
00195     }
00196
00197     return matches;
00198 }
00199
00200 char* command_generator (char* text, int state) {
00201     static int list_index, len;
00202     char* name;
00203
00204     if (!state) {
00205         list_index = 0;
00206         len = strlen (text);
00207     }
00208
00209     /* Return the next name which partially matches from the command list. */
00210     while (name = commands[list_index].name) {
00211         ++list_index;
00212
00213         if (strncmp (name, text, len) == 0) {
00214             return dupstr (name);
00215         }
00216     }
00217
00218     /* If no names matched, then return NULL. */
00219 }
```

```
00246     return (char*) NULL;
00247 }
00248
00249 /* ***** */
00250 /* ***** */
00251 /* ***** airinv Commands ***** */
00252 /* ***** */
00253 /* ***** */
00254
00259 static char syscom[1024];
00260
00264 void com_list (char* arg) {
00265     if (!arg) {
00266         arg = "";
00267     }
00268
00269     std::ostringstream oStr;
00270     oStr << "ls -FClg " << arg;
00271     return system (oStr.c_str());
00272 }
00273
00274 int com_view (char* arg) {
00275     if (!valid_argument ("view", arg)) {
00276         return 1;
00277     }
00278
00279     std::ostringstream oStr;
00280     oStr << "more " << arg;
00281     return system (syscom);
00282 }
00283
00284 int com_rename (char* arg) {
00285     too_dangerous ("rename");
00286     return 1;
00287 }
00288
00289 int com_stat (char* arg) {
00290     struct stat finfo;
00291
00292     if (!valid_argument ("stat", arg)) {
00293         return 1;
00294     }
00295
00296     if (stat (arg, &finfo) == -1) {
00297         perror (arg);
00298         return 1;
00299     }
00300
00301     std::cout << "Statistics for \"" << arg << "':" << std::endl;
00302
00303     const std::string lPluralEnd1 = (finfo.st_nlink == 1) ? "" : "s";
00304     const std::string lPluralEnd2 = (finfo.st_size == 1) ? "" : "s";
00305     std::cout << arg << " has "
00306               << finfo.st_nlink << " link" << lPluralEnd1 << ", and is "
00307               << finfo.st_size << " byte" << lPluralEnd2 << " in length."
00308               << std::endl;
00309     std::cout << " Inode Last Change at: " << ctime (&finfo.st_ctime) << std::endl;
00310
00311     std::cout << " Last access at: " << ctime (&finfo.st_atime) << std::endl;
00312     std::cout << " Last modified at: " << ctime (&finfo.st_mtime) << std::endl;
00313     return 0;
00314 }
```

```
00314
00315 int com_delete (char* arg) {
00316     too_dangerous ("delete");
00317     return 1;
00318 }
00319
00324 int com_help (char* arg) {
00325     register int i;
00326     int printed = 0;
00327
00328     for (i = 0; commands[i].name; i++) {
00329         if (!*arg || (strcmp (arg, commands[i].name) == 0)) {
00330             printf ("%s\t\t%s.\n", commands[i].name, commands[i].doc);
00331             printed++;
00332         }
00333     }
00334
00335     if (!printed) {
00336         printf ("No commands match '%s'. Possibilities are:\n", arg);
00337
00338         for (i = 0; commands[i].name; i++) {
00339             /* Print in six columns. */
00340             if (printed == 6) {
00341                 printed = 0;
00342                 printf ("\n");
00343             }
00344
00345             printf ("%s\t", commands[i].name);
00346             printed++;
00347         }
00348
00349         if (printed)
00350             printf ("\n");
00351     }
00352     return 0;
00353 }
00354
00355 /* Change to the directory ARG. */
00356 int com_cd (char* arg) {
00357     if (chdir (arg) == -1) {
00358         perror (arg);
00359         return 1;
00360     }
00361
00362     com_pwd ("");
00363     return 0;
00364 }
00365
00366 /* Print out the current working directory. */
00367 int com_pwd (char* ignore) {
00368     char dir[1024], *s;
00369
00370     s = getwd (dir);
00371     if (s == 0) {
00372         printf ("Error getting pwd: %s\n", dir);
00373         return 1;
00374     }
00375
00376     printf ("Current directory is %s\n", dir);
00377     return 0;
00378 }
00379
```

```

00380 /* The user wishes to quit using this program. Just set DONE non-zero. */
00381 int com_quit (char* arg) {
00382     done = 1;
00383     return 0;
00384 }
00385
00386 /* Function which tells you that you can't do this. */
00387 void too_dangerous (char* caller) {
00388     fprintf (stderr,
00389             "%s: Too dangerous for me to distribute. Write it yourself.\n",
00390             caller);
00391 }
00392
00393 /* Return non-zero if ARG is a valid argument for CALLER, else print
00394  * an error message and return zero. */
00395 int valid_argument (char* caller, char* arg) {
00396     if (!arg || !*arg) {
00397         fprintf (stderr, "%s: Argument required.\n", caller);
00398         return 0;
00399     }
00400
00401     return 1;
00402 }
00403
00404 #endif // _AIRINV_READLINE_AUTOCOMP_HPP

```

35.559 stdair/ui/cmdline/SReadline.hpp File Reference

C++ wrapper around libreadline.

```

#include <stdio.h>
#include <readline/readline.h>
#include <readline/history.h>
#include <readline/keymaps.h>
#include <string>
#include <fstream>
#include <vector>
#include <stdexcept>
#include <map>
#include <boost/algorithm/string/trim.hpp>
#include <boost/tokenizer.hpp>
#include <boost/function.hpp>

```

Classes

- class [swift::SKeymap](#)
The readline keymap wrapper.
- class [swift::SReadline](#)

The readline library wrapper.

Namespaces

- namespace [swift](#)

The wrapper namespace.

35.559.1 Detailed Description

C++ wrapper around libreadline. Supported: editing, history, custom completers, keymaps. Attention: implementation is not thread safe! It is mainly because the readline library provides pure C interface and has many calls for an "atomic" completion operation

Definition in file [SReadline.hpp](#).

35.560 SReadline.hpp

```

00001
00011 //
00012 // Date:      17 December 2005
00013 //           03 April    2006
00014 //           20 April    2006
00015 //           07 May      2006
00016 //
00017 // Copyright (c) Sergey Satskiy 2005 - 2006
00018 //           <sergesatsky@yahoo.com>
00019 //
00020 // Permission to copy, use, modify, sell and distribute this software
00021 // is granted provided this copyright notice appears in all copies.
00022 // This software is provided "as is" without express or implied
00023 // warranty, and with no claim as to its suitability for any purpose.
00024 //
00025
00026 #ifndef SREADLINE_H
00027 #define SREADLINE_H
00028
00029 #include <stdio.h>
00030
00031 #include <readline/readline.h>
00032 #include <readline/history.h>
00033 #include <readline/keymaps.h>
00034
00035 #include <string>
00036 #include <fstream>
00037 #include <vector>
00038 #include <stdexcept>
00039 #include <map>
00040
00041 #include <boost/algorithm/string/trim.hpp>
00042 #include <boost/tokenizer.hpp>
00043 #include <boost/function.hpp>
00044
00045
00050 namespace {
00054     typedef std::vector<std::string> TokensStorage;
00055

```

```
00059     typedef std::vector<TokensStorage> CompletionsStorage;
00060
00064     typedef boost::function<int (int, int)> KeyCallback;
00065
00069     typedef std::map<int, KeyCallback> KeysBind;
00070
00074     const size_t DefaultHistoryLimit (64);
00075
00079     CompletionsStorage Completions;
00080
00084     TokensStorage Tokens;
00085
00089     std::map<Keymap, KeysBind> Keymaps;
00090
00094     bool KeymapWasSetup (false);
00095
00099     Keymap Earlykeymap (0);
00100
00101
00108     char* Generator (const char* text, int State);
00109
00110
00118     char** UserCompletion (const char* text, int start, int end);
00119
00120
00128     int KeyDispatcher (int Count, int Key);
00129
00130
00135     int StartupHook (void);
00136
00137
00145     template <typename Container>
00146     bool AreTokensEqual (const Container& Pattern, const Container& Input) {
00147         if (Input.size() > Pattern.size()) {
00148             return false;
00149         }
00150
00151         typename Container::const_iterator k (Pattern.begin());
00152         typename Container::const_iterator j (Input.begin());
00153         for ( ; j != Input.end(); ++k, ++j) {
00154             const std::string lPattern = *k;
00155             if (lPattern == "%file") {
00156                 continue;
00157             }
00158
00159             const std::string lInput = *j;
00160             if (lPattern != lInput) {
00161                 return false;
00162             }
00163         }
00164         return true;
00165     }
00166
00167     // See description near the prototype
00168     template <typename ContainerType>
00169     void SplitTokens (const std::string& Source, ContainerType& Container) {
00170         typedef boost::tokenizer<boost::char_separator<char> > TokenizerType;
00171
00172         // Set of token separators
00173         boost::char_separator<char> Separators (" \\t\\n");
00174         // Tokens provider
00175         TokenizerType Tokenizer (Source, Separators);
```

```

00176     Container.clear();
00177     for (TokenizerType::const_iterator k (Tokenizer.begin());
00178         k != Tokenizer.end(); ++k) {
00179         // Temporary storage for the token, in order to trim that latter
00180         std::string SingleToken (*k);
00181
00182         boost::algorithm::trim (SingleToken);
00183         Container.push_back (SingleToken);
00184     }
00185 }
00186
00187 // See description near the prototype
00188 char** UserCompletion (const char* text, int start, int end) {
00189     // No default completion at all
00190     rl_attempted_completion_over = 1;
00191
00192     if (Completions.empty() == true) {
00193         return NULL;
00194     }
00195
00196     // Memorise all the previous tokens
00197     std::string PreInput (rl_line_buffer, start);
00198     SplitTokens (PreInput, Tokens);
00199
00200     // Detect whether we should call the standard file name completer
00201     // or a custom one
00202     bool FoundPretender (false);
00203
00204     for (CompletionsStorage::const_iterator k (Completions.begin());
00205         k != Completions.end(); ++k) {
00206         const TokensStorage& lTokenStorage = *k;
00207         if (AreTokensEqual (lTokenStorage, Tokens) == false) {
00208             continue;
00209         }
00210
00211         if (lTokenStorage.size() > Tokens.size()) {
00212             FoundPretender = true;
00213             if (lTokenStorage [Tokens.size()] == "%file") {
00214                 // Standard file name completer - called for the "%file" keyword
00215                 return rl_completion_matches (text, rl_filename_completion_function);
00216             }
00217         }
00218     }
00219
00220     if (FoundPretender) {
00221         return rl_completion_matches (text, Generator);
00222     }
00223     return NULL;
00224 }
00225
00226 // See description near the prototype
00227 char* Generator (const char* text, int State) {
00228     static int Length;
00229     static CompletionsStorage::const_iterator Iterator;
00230
00231     if ( State == 0 ) {
00232         Iterator = Completions.begin();
00233         Length = strlen (text);
00234     }
00235
00236     for ( ; Iterator != Completions.end(); ++Iterator) {

```

```
00238     const TokensStorage& lCompletion = *Iterator;
00239     if (AreTokensEqual (lCompletion, Tokens) == false) {
00240         continue;
00241     }
00242
00243     if (lCompletion.size() > Tokens.size()) {
00244         if (lCompletion [Tokens.size()] == "%file") {
00245             continue;
00246         }
00247
00248         const char* lCompletionCharStr (lCompletion [Tokens.size()].c_str());
00249         if (strncmp (text, lCompletionCharStr, Length) == 0) {
00250             // Readline will free the allocated memory
00251             const size_t lCompletionSize = strlen (lCompletionCharStr) + 1;
00252             char* NewString (static_cast<char*> (malloc (lCompletionSize)));
00253             strcpy (NewString, lCompletionCharStr);
00254
00255             ++Iterator;
00256
00257             return NewString;
00258         }
00259     }
00260 }
00261
00262 return NULL;
00263 }
00264
00265
00266 // See the description near the prototype
00267 int KeyDispatcher (int Count, int Key ) {
00268     std::map< Keymap, KeysBind >::iterator Set (Keymaps.find (rl_get_keymap()));
00269     if (Set == Keymaps.end()) {
00270         // Most probably it happens because the header was
00271         // included into many compilation units and the
00272         // keymap setting calls were made in different files.
00273         // This is the problem of "global" data.
00274         // The storage of all the registered keymaps is in anonymous
00275         // namespace.
00276         throw std::runtime_error ("Error selecting a keymap.");
00277     }
00278
00279     (Set->second)[Key] (Count, Key);
00280     return 0;
00281 }
00282
00283 // See the description near the prototype
00284 int StartupHook (void) {
00285     if (KeymapWasSetup) {
00286         rl_set_keymap (Earlykeymap);
00287     }
00288     return 0;
00289 }
00290
00291 } // Anonymous namespace
00292
00293
00299 namespace swift {
00300
00307     class SKeymap {
00308     private:
00309         // Readline keymap
00310         Keymap keymap;
```

```

00311
00312     public:
00319     explicit SKeymap (bool PrintableBound = false) : keymap (NULL) {
00320         if (PrintableBound == true) {
00321             // Printable characters are bound
00322             keymap = rl_make_keymap();
00323
00324         } else {
00325             // Empty keymap
00326             keymap = rl_make_bare_keymap();
00327         }
00328
00329         if (keymap == NULL) {
00330             throw std::runtime_error ("Cannot allocate keymap.");
00331         }
00332
00333         // Register a new keymap in the global list
00334         Keymaps [keymap] = KeysBind();
00335     }
00336
00342     explicit SKeymap (Keymap Pattern) : keymap (rl_copy_keymap (Pattern)) {
00343         if ( keymap == NULL ) {
00344             throw std::runtime_error( "Cannot allocate keymap." );
00345         }
00346
00347         // Register a new keymap in the global list
00348         Keymaps [keymap] = KeysBind();
00349     }
00350
00354     ~SKeymap() {
00355         // Deregister the keymap
00356         Keymaps.erase (keymap);
00357         rl_discard_keymap (keymap);
00358     }
00359
00366     void Bind (int Key, KeyCallback Callback) {
00367         Keymaps [keymap][Key] = Callback;
00368
00369         if (rl_bind_key_in_map (Key, KeyDispatcher, keymap) != 0) {
00370             // Remove from the map just bound key
00371             Keymaps [keymap].erase (Key);
00372             throw std::runtime_error ("Invalid key.");
00373         }
00374     }
00375
00381     void Unbind (int Key) {
00382         rl_unbind_key_in_map (Key, keymap);
00383         Keymaps [keymap].erase (Key);
00384     }
00385
00386     // void Bind (const std::string& Sequence, boost::function<int (int, int)>);
00387     // void Unbind (std::string& Sequence);
00388
00389     public:
00395     SKeymap (const SKeymap& rhs) {
00396         if (this == &rhs) {
00397             return;
00398         }
00399         keymap = rl_copy_keymap (rhs.keymap);
00400     }
00401
00407     SKeymap& operator= (const SKeymap& rhs) {

```

```

00408         if (this == &rhs) {
00409             return *this;
00410         }
00411         keymap = rl_copy_keymap (rhs.keymap);
00412         return *this;
00413     }
00414
00415     friend class SReadline;
00416 };
00417
00424 class SReadline {
00425 public:
00431     SReadline (const size_t Limit = DefaultHistoryLimit) :
00432         HistoryLimit (Limit), HistoryFileName (""),
00433         OriginalCompletion (rl_attempted_completion_function) {
00434         rl_startup_hook = StartupHook;
00435         rl_attempted_completion_function = UserCompletion;
00436         using_history();
00437     }
00438
00446     SReadline( const std::string & historyFileName,
00447               const size_t Limit = DefaultHistoryLimit ) :
00448         HistoryLimit( Limit ),
00449         HistoryFileName( historyFileName ),
00450         OriginalCompletion( rl_attempted_completion_function )
00451     {
00452         rl_startup_hook = StartupHook;
00453         rl_attempted_completion_function = UserCompletion;
00454         using_history();
00455         LoadHistory( HistoryFileName );
00456     }
00457
00462     ~SReadline() {
00463         rl_attempted_completion_function = OriginalCompletion;
00464         SaveHistory (HistoryFileName);
00465     }
00466
00473     std::string GetLine (const std::string& Prompt) {
00474         bool Unused;
00475         return GetLine (Prompt, Unused);
00476     }
00477
00486     template <typename Container>
00487     std::string GetLine (const std::string& Prompt, Container& ReadTokens) {
00488         bool Unused;
00489         return GetLine (Prompt, ReadTokens, Unused);
00490     }
00491
00501     template <typename Container>
00502     std::string GetLine (const std::string& Prompt, Container& ReadTokens,
00503                         bool& BreakOut) {
00504         std::string Input (GetLine (Prompt, BreakOut));
00505         SplitTokens (Input, ReadTokens);
00506         return Input;
00507     }
00508
00509
00517     std::string GetLine (const std::string& Prompt, bool& BreakOut) {
00518         BreakOut = true;
00519
00520         char* ReadLine (readline (Prompt.c_str()));
00521         if (ReadLine == NULL) {

```

```
00522         return std::string();
00523     }
00524
00525     // It's OK
00526     BreakOut = false;
00527     std::string Input (ReadLine);
00528     free (ReadLine); ReadLine = NULL;
00529
00530     boost::algorithm::trim (Input);
00531     if (Input.empty() == false) {
00532         if (history_length == 0
00533             || Input != history_list()[ history_length - 1 ]->line) {
00534             add_history (Input.c_str());
00535
00536             if (history_length >= static_cast<int> (HistoryLimit)) {
00537                 stifle_history (HistoryLimit);
00538             }
00539         }
00540     }
00541
00542     return Input;
00543 }
00544
00545
00551 template <typename ContainerType>
00552 void GetHistory (ContainerType& Container) {
00553     for (int k (0); k < history_length; ++k ) {
00554         Container.push_back (history_list()[k]->line);
00555     }
00556 }
00557
00564 bool SaveHistory (std::ostream& OS) {
00565     if (!OS) {
00566         return false;
00567     }
00568
00569     for (int k (0); k < history_length; ++k) {
00570         OS << history_list()[ k ]->line << std::endl;
00571     }
00572     return true;
00573 }
00574
00581 bool SaveHistory (const std::string& FileName) {
00582     if (FileName.empty() == true) {
00583         return false;
00584     }
00585
00586     std::ofstream OS (FileName.c_str());
00587     return SaveHistory (OS);
00588 }
00589
00594 void ClearHistory() {
00595     clear_history();
00596 }
00597
00604 bool LoadHistory (std::istream& IS) {
00605     if (!IS) {
00606         return false;
00607     }
00608
00609     ClearHistory();
00610     std::string OneLine;
```

```

00611
00612     while (!getline (IS, OneLine).eof()) {
00613         boost::algorithm::trim( OneLine );
00614         if ((history_length == 0)
00615             || OneLine != history_list()[history_length - 1]->line) {
00616             add_history (OneLine.c_str());
00617         }
00618     }
00619     stifile_history (HistoryLimit);
00620     return true;
00621 }
00622
00629 bool LoadHistory (const std::string& FileName) {
00630     if (FileName.empty() == true) {
00631         return false;
00632     }
00633
00634     std::ifstream IS (FileName.c_str());
00635     return LoadHistory (IS);
00636 }
00637
00657 template <typename ContainerType>
00658 void RegisterCompletions (const ContainerType& Container) {
00659     Completions.clear();
00660     for (typename ContainerType::const_iterator k (Container.begin());
00661          k != Container.end(); ++k) {
00662         std::vector<std::string> OneLine;
00663         const std::string& kStr = static_cast<std::string> (*k);
00664
00665         SplitTokens (kStr, OneLine);
00666         Completions.push_back (OneLine);
00667     }
00668 }
00669
00675 void SetKeymap (SKeymap& NewKeymap) {
00676     rl_set_keymap (NewKeymap.keymap);
00677     KeymapWasSetup = true;
00678     Earlykeymap = NewKeymap.keymap;
00679 }
00680
00681
00682 private:
00683     // ////////////////////////////////// Attributes //////////////////////////////////
00684     const size_t HistoryLimit;
00685
00686     const std::string HistoryFileName;
00687
00692     rl_completion_func_t* OriginalCompletion;
00693 };
00694
00700 }; // namespace swift
00701
00702 #endif
00703

```

35.561 test/stdair/MPBomRoot.cpp File Reference

35.562 MPBomRoot.cpp

```
00001
```

```

00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <cassert>
00010 // StdAir Test
00011 #include <test/stdair/MPBomRoot.hpp>
00012
00013 namespace myprovider {
00014
00015 // //////////////////////////////////////
00016 BomRoot::BomRoot (const Key_T& iKey) : stdair::BomRoot (iKey) {
00017 }
00018
00019 // //////////////////////////////////////
00020 BomRoot::~BomRoot () {
00021 }
00022
00023 }

```

35.563 test/stdair/MPBomRoot.hpp File Reference

35.564 MPBomRoot.hpp

```

00001 #ifndef __MYPROVIDER_BOMROOT_HPP
00002 #define __MYPROVIDER_BOMROOT_HPP
00003
00008 // //////////////////////////////////////
00009 // Import section
00010 // //////////////////////////////////////
00011 // STL
00012 #include <string>
00013 // StdAir
00014 #include <stdair/bom/BomRoot.hpp>
00015
00016 namespace myprovider {
00017
00020 class BomRoot : public stdair::BomRoot {
00021 public:
00022 // /////////// Display support methods ///////////
00024 std::string toString() const { return describeKey(); }
00025
00028 const std::string describeKey() const { return std::string (""); }
00029
00030 public:
00034 BomRoot (const Key_T&);
00036 ~BomRoot ();
00038 BomRoot ();
00039 BomRoot (const BomRoot&);
00040 };
00041
00042 }
00046 #endif // __MYPROVIDER_BOMROOT_HPP

```

35.565 test/stdair/MPInventory.cpp File Reference

35.566 MPIInventory.cpp

```

00001
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <cassert>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 // StdAir Test
00013 #include <test/stdair/MPIInventory.hpp>
00014
00015 namespace myprovider {
00016
00017 // //////////////////////////////////////
00018 Inventory::Inventory (const Key_T& iKey) : stdair::Inventory (iKey) {
00019 }
00020
00021 // //////////////////////////////////////
00022 Inventory::~Inventory () {
00023 }
00024
00025 // //////////////////////////////////////
00026 std::string Inventory::toString() const {
00027     std::ostringstream oStr;
00028     oStr << _key.toString();
00029     return oStr.str();
00030 }
00031
00032 // //////////////////////////////////////
00033 const std::string Inventory::describeKey() const {
00034     return _key.toString();
00035 }
00036
00037 }

```

35.567 test/stdair/MPIInventory.hpp File Reference

35.568 MPIInventory.hpp

```

00001 #ifndef __MYPROVIDER_INVENTORY_HPP
00002 #define __MYPROVIDER_INVENTORY_HPP
00003
00008 // //////////////////////////////////////
00009 // Import section
00010 // //////////////////////////////////////
00011 // STL
00012 #include <list>
00013 // StdAir
00014 #include <stdair/bom/Inventory.hpp>
00015
00016 namespace myprovider {
00017
00018     class Inventory : public stdair::Inventory {
00019     public:
00020         // //////////// Display support methods ////////////
00022         std::string toString() const;
00023

```

```

00026     const std::string describeKey() const;
00027
00028 public:
00032     Inventory (const Key_T&);
00034     ~Inventory();
00036     Inventory ();
00037     Inventory (const Inventory&);
00038 };
00039
00040 // /////////// Type definitions ///////////
00042 typedef std::list<Inventory*> InventoryList_T;
00043
00044 }
00048 #endif // __MYPROVIDER_INVENTORY_HPP

```

35.569 test/stdair/StandardAirlineITTestSuite.cpp File Reference

35.570 StandardAirlineITTestSuite.cpp

```

00001
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <sstream>
00010 #include <fstream>
00011 #include <string>
00012 // Boost MPL
00013 #include <boost/mpl/push_back.hpp>
00014 #include <boost/mpl/vector.hpp>
00015 #include <boost/mpl/at.hpp>
00016 #include <boost/mpl/assert.hpp>
00017 #include <boost/type_traits/is_same.hpp>
00018 // Boost Unit Test Framework (UTF)
00019 #define BOOST_TEST_DYN_LINK
00020 #define BOOST_TEST_MAIN
00021 #define BOOST_TEST_MODULE StdAirTest
00022 #if BOOST_VERSION >= 103900
00023 #include <boost/test/unit_test.hpp>
00024 #else // BOOST_VERSION >= 103900
00025 #include <boost/test/test_tools.hpp>
00026 #include <boost/test/results_reporter.hpp>
00027 #include <boost/test/unit_test_suite.hpp>
00028 #include <boost/test/output_test_stream.hpp>
00029 #include <boost/test/unit_test_log.hpp>
00030 #include <boost/test/framework.hpp>
00031 #include <boost/test/detail/unit_test_parameters.hpp>
00032 #endif // BOOST_VERSION >= 103900
00033 // Boost Serialisation
00034 #include <boost/archive/text_oarchive.hpp>
00035 #include <boost/archive/text_iarchive.hpp>
00036 // StdAir
00037 #include <stdair/stdair_inventory_types.hpp>
00038 #include <stdair/service/Logger.hpp>
00039 #include <stdair/STDAIR_Service.hpp>
00040 #include <stdair/basic/float_utils.hpp>
00041 #include <stdair/bom/BomDisplay.hpp>
00042 #include <stdair/bom/BomRoot.hpp>
00043 #include <stdair/bom/BomManager.hpp>
00044 #include <stdair/factory/FacBom.hpp>

```

```
00045 #include <stdair/factory/FacBomManager.hpp>
00046 // StdAir Test Suite
00047 #include <test/stdair/StdairTestLib.hpp>
00048 #include <test/stdair/MPInventory.hpp>
00049
00050 namespace boost_utf = boost::unit_test;
00051
00052 #if BOOST_VERSION >= 103900
00053
00054 // (Boost) Unit Test XML Report
00055 std::ofstream utfReportStream ("StandardAirlineITTestSuite_utfresults.xml");
00056
00060 struct UnitTestConfig {
00062     UnitTestConfig() {
00063         boost_utf::unit_test_log.set_stream (utfReportStream);
00064         boost_utf::unit_test_log.set_format (boost_utf::XML);
00065         boost_utf::unit_test_log.set_threshold_level (boost_utf::log_test_units);
00066         // boost_utf::unit_test_log.set_threshold_level (boost_utf::log_successful_te
sts);
00067     }
00068
00070     ~UnitTestConfig() {
00071     }
00072 };
00073
00074
00075 // ////////////////////////////////// Main: Unit Test Suite //////////////////////////////////
00076
00077 // Set the UTF configuration (re-direct the output to a specific file)
00078 BOOST_GLOBAL_FIXTURE (UnitTestConfig);
00079
00080 // Start the test suite
00081 BOOST_AUTO_TEST_SUITE (master_test_suite)
00082
00083
00087 BOOST_AUTO_TEST_CASE (float_comparison_test) {
00088     float a = 0.2f;
00089     a = 5*a;
00090     const float b = 1.0f;
00091
00092     // Test the Boost way
00093     BOOST_CHECK_MESSAGE (a == b, "The two floats (" << a << " and " << b
<< " should be equal, but are not");
00094
00095     BOOST_CHECK_CLOSE (a, b, 0.0001);
00096
00097     // Test the Google way
00098     const FloatingPoint<float> lhs (a), rhs (b);
00099     BOOST_CHECK_MESSAGE (lhs.AlmostEquals (rhs),
10100         "The two floats (" << a << " and " << b
10101         << " should be equal, but are not");
10102 }
10103
10108 BOOST_AUTO_TEST_CASE (mpl_structure_test) {
10109     const stdair::ClassCode_T lBookingClassCodeA ("A");
10110     const stdair_test::BookingClass lA (lBookingClassCodeA);
10111     const stdair_test::Cabin lCabin (lA);
10112
10113     BOOST_CHECK_EQUAL (lCabin.toString(), lBookingClassCodeA);
10114     BOOST_CHECK_MESSAGE (lCabin.toString() == lBookingClassCodeA,
10115         "The cabin key, '" << lCabin.toString()
10116         << "' is not equal to '" << lBookingClassCodeA << "'");
10117 }
```

```

00118 // MPL
00119 typedef boost::mpl::vector<stdair_test::BookingClass> MPL_BookingClass;
00120 typedef boost::mpl::push_back<MPL_BookingClass,
00121                               stdair_test::Cabin>::type types;
00122
00123 if (boost::is_same<stdair_test::BookingClass,
00124                   stdair_test::Cabin::child>::value == false) {
00125     BOOST_REQUIRE ("The two types must be equal, but are not");
00126 }
00127
00128 if (boost::is_same<boost::mpl::at_c<types, 1>::type,
00129                   stdair_test::Cabin>::value == false) {
00130     BOOST_REQUIRE ("The type must be stdair_test::Cabin, but is not");
00131 }
00132 }
00133
00137 BOOST_AUTO_TEST_CASE (stdair_service_initialisation_test) {
00138     // Output log File
00139     const std::string lLogFilename ("StandardAirlineITTestSuite_init.log");
00140
00141     // Set the log parameters
00142     std::ofstream logOutputFile;
00143
00144     // Open and clean the log outputfile
00145     logOutputFile.open (lLogFilename.c_str());
00146     logOutputFile.clear();
00147
00148     // Initialise the stdair BOM
00149     const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
00150     stdair::STDAIR_Service stdairService (lLogParams);
00151
00152     // Retrieve (a reference on) the top of the BOM tree
00153     stdair::BomRoot& lBomRoot = stdairService.getBomRoot();
00154
00155     // Retrieve the BomRoot key, and compare it to the expected one
00156     const std::string& lBomRootKeyStr = lBomRoot.describeKey();
00157     const std::string lBomRootString (" -- ROOT -- ");
00158
00159     // DEBUG
00160     STDAIR_LOG_DEBUG ("The BOM root key is '" << lBomRootKeyStr
00161                      << "'. It should be equal to '" << lBomRootString << "'");
00162
00163     BOOST_CHECK_EQUAL (lBomRootKeyStr, lBomRootString);
00164     BOOST_CHECK_MESSAGE (lBomRootKeyStr == lBomRootString,
00165                          "The BOM root key, '" << lBomRootKeyStr
00166                          << "', should be equal to '" << lBomRootString
00167                          << "', but is not.");
00168
00169     // Build a sample BOM tree
00170     stdairService.buildSampleBom();
00171
00172     // DEBUG: Display the whole BOM tree
00173     const std::string& lCSVDump = stdairService.csvDisplay();
00174     STDAIR_LOG_DEBUG (lCSVDump);
00175
00176     // Close the Log outputFile
00177     logOutputFile.close();
00178 }
00179
00183 BOOST_AUTO_TEST_CASE (bom_structure_instantiation_test) {
00184     // Step 0.0: initialisation
00185     // Create the root of the Bom tree (i.e., a BomRoot object)

```

```

00186     stdair::BomRoot& lBomRoot =
00187         stdair::FacBom<stdair::BomRoot>::instance().create();
00188
00189     // Step 0.1: Inventory level
00190     // Create an Inventory (BA)
00191     const stdair::AirlineCode_T lBAAirlineCode ("BA");
00192     const stdair::InventoryKey lBAKey (lBAAirlineCode);
00193     myprovider::Inventory& lBAInv =
00194         stdair::FacBom<myprovider::Inventory>::instance().create (lBAKey);
00195     stdair::FacBomManager::addToList (lBomRoot, lBAInv);
00196
00197     BOOST_CHECK_EQUAL (lBAInv.describeKey(), lBAAirlineCode);
00198     BOOST_CHECK_MESSAGE (lBAInv.describeKey() == lBAAirlineCode,
00199         "The inventory key, '" << lBAInv.describeKey()
00200         << "'", should be equal to '" << lBAAirlineCode
00201         << "'", but is not");
00202
00203     // Create an Inventory for AF
00204     const stdair::AirlineCode_T lFAAirlineCode ("AF");
00205     const stdair::InventoryKey lAFKey (lFAAirlineCode);
00206     myprovider::Inventory& lAFInv =
00207         stdair::FacBom<myprovider::Inventory>::instance().create (lAFKey);
00208     stdair::FacBomManager::addToList (lBomRoot, lAFInv);
00209
00210     BOOST_CHECK_EQUAL (lAFInv.describeKey(), lFAAirlineCode);
00211     BOOST_CHECK_MESSAGE (lAFInv.describeKey() == lFAAirlineCode,
00212         "The inventory key, '" << lAFInv.describeKey()
00213         << "'", should be equal to '" << lFAAirlineCode
00214         << "'", but is not");
00215
00216     // Browse the inventories
00217     const myprovider::InventoryList_T& lInventoryList =
00218         stdair::BomManager::getList<myprovider::Inventory> (lBomRoot);
00219     const std::string lInventoryKeyArray[2] = {lBAAirlineCode, lFAAirlineCode};
00220     short idx = 0;
00221     for (myprovider::InventoryList_T::const_iterator itInv =
00222         lInventoryList.begin(); itInv != lInventoryList.end();
00223         ++itInv, ++idx) {
00224         const myprovider::Inventory* lInv_ptr = *itInv;
00225         BOOST_REQUIRE (lInv_ptr != NULL);
00226
00227         BOOST_CHECK_EQUAL (lInventoryKeyArray[idx], lInv_ptr->describeKey());
00228         BOOST_CHECK_MESSAGE (lInventoryKeyArray[idx] == lInv_ptr->describeKey(),
00229             "They inventory key, '" << lInventoryKeyArray[idx]
00230             << "'", does not match that of the Inventory object: '"
00231             << lInv_ptr->describeKey() << "'");
00232     }
00233 }
00234
00238 BOOST_AUTO_TEST_CASE (bom_structure_serialisation_test) {
00239
00240     // Backup (thanks to Boost.Serialisation) file
00241     const std::string lBackupFilename = "StandardAirlineITTestSuite_serial.txt";
00242
00243     // Output log File
00244     const std::string lLogFilename ("StandardAirlineITTestSuite_serial.log");
00245
00246     // Set the log parameters
00247     std::ofstream logOutputFile;
00248
00249     // Open and clean the log outputfile
00250     logOutputFile.open (lLogFilename.c_str());

```

```

00251     logOutputFile.clear();
00252
00253     // Initialise the stdair BOM
00254     const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
00255     stdair::STDAIR_Service stdairService (lLogParams);
00256
00257     // Build a sample BOM tree
00258     stdairService.buildSampleBom();
00259
00260     // DEBUG: Display the whole BOM tree
00261     const std::string& lCSVDump = stdairService.csvDisplay();
00262     STDAIR_LOG_DEBUG (lCSVDump);
00263
00264     // Retrieve (a reference on) the top of the BOM tree
00265     stdair::BomRoot& lBomRoot = stdairService.getBomRoot();
00266
00267     // Retrieve the BomRoot key, and compare it to the expected one
00268     const std::string lBAInvKeyStr ("BA");
00269     stdair::Inventory* lBAInv_ptr = lBomRoot.getInventory (lBAInvKeyStr);
00270
00271     // DEBUG
00272     STDAIR_LOG_DEBUG ("There should be an Inventory object corresponding to the '"
00273         << lBAInvKeyStr << "' key.");
00274
00275     BOOST_REQUIRE_MESSAGE (lBAInv_ptr != NULL,
00276         "An Inventory object should exist with the key, '"
00277         << lBAInvKeyStr << "'");
00278
00279     // create and open a character archive for output
00280     std::ofstream ofs (lBackupFilename.c_str());
00281
00282     // save data to archive
00283     {
00284         boost::archive::text_oarchive oa (ofs);
00285         // write class instance to archive
00286         oa << lBomRoot;
00287         // archive and stream closed when destructors are called
00288     }
00289
00290     // ... some time later restore the class instance to its original state
00291     stdair::BomRoot& lRestoredBomRoot =
00292         stdair::FacBom<stdair::BomRoot>::instance().create();
00293     {
00294         // create and open an archive for input
00295         std::ifstream ifs (lBackupFilename.c_str());
00296         boost::archive::text_iarchive ia(ifs);
00297         // read class state from archive
00298         ia >> lRestoredBomRoot;
00299         // archive and stream closed when destructors are called
00300     }
00301
00302     // DEBUG: Display the whole BOM tree
00303     std::ostream oRestoredCSVDumpStr;
00304     stdair::BomDisplay::csvDisplay (oRestoredCSVDumpStr, lRestoredBomRoot);
00305     STDAIR_LOG_DEBUG (oRestoredCSVDumpStr.str());
00306
00307     // Retrieve the BomRoot key, and compare it to the expected one
00308     const std::string& lBomRootKeyStr = lRestoredBomRoot.describeKey();
00309     const std::string lBomRootString (" -- ROOT -- ");
00310
00311     // DEBUG
00312     STDAIR_LOG_DEBUG ("The BOM root key is '" << lBomRootKeyStr

```

```

00313             << "'. It should be equal to '" << lBomRootString << "'");
00314
00315     BOOST_CHECK_EQUAL (lBomRootKeyStr, lBomRootString);
00316     BOOST_CHECK_MESSAGE (lBomRootKeyStr == lBomRootString,
00317         "The BOM root key, '" << lBomRootKeyStr
00318         << "'", should be equal to '" << lBomRootString
00319         << "'", but is not.");
00320
00321     // Retrieve the Inventory
00322     stdair::Inventory* lRestoredBAInv_ptr =
00323         lRestoredBomRoot.getInventory (lBAInvKeyStr);
00324
00325     // DEBUG
00326     STDAIR_LOG_DEBUG ("There should be an Inventory object corresponding to the '"
00327         << lBAInvKeyStr << "' key.");
00328
00329     BOOST_CHECK_MESSAGE (lRestoredBAInv_ptr != NULL,
00330         "An Inventory object should exist with the key, '"
00331         << lBAInvKeyStr << "'");
00332
00333     // Close the Log outputFile
00334     logOutputFile.close();
00335 }
00336
00337 // End the test suite
00338 BOOST_AUTO_TEST_SUITE_END()
00339
00340 #else // BOOST_VERSION >= 103900
00341 boost_utf::test_suite* init_unit_test_suite (int, char* []) {
00342     boost_utf::test_suite* test = BOOST_TEST_SUITE ("Unit test example 1");
00343     return test;
00344 }
00345 #endif // BOOST_VERSION >= 103900
00346

```

35.571 test/stdair/StdairTestLib.hpp File Reference

```

#include <string>
#include <sstream>

```

Classes

- struct [stdair_test::BookingClass](#)
- struct [stdair_test::Cabin](#)

Namespaces

- namespace [stdair_test](#)

35.572 StdairTestLib.hpp

```

00001 #ifndef __STDAIR_TST_STDAIR_TEST_LIB_HPP
00002 #define __STDAIR_TST_STDAIR_TEST_LIB_HPP

```

```
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 #include <string>
00008 #include <sstream>
00009
00013 namespace stdair_test {
00014
00016     struct BookingClass {
00017         std::string _classCode;
00019         BookingClass (const std::string& iClassCode)
00020             : _classCode (iClassCode) {
00021         }
00022
00024         std::string toString() const {
00025             std::ostringstream ostr;
00026             ostr << _classCode;
00027             return ostr.str();
00028         }
00029     };
00030
00032     struct Cabin {
00033         BookingClass _bookingClass;
00034         Cabin (const BookingClass& iBkgClass)
00035             : _bookingClass (iBkgClass) {
00036         }
00037
00039         std::string toString() const {
00040             std::ostringstream ostr;
00041             ostr << _bookingClass._classCode;
00042             return ostr.str();
00043         }
00044
00046         typedef BookingClass child;
00047     };
00048 }
00049
00050 #endif // __STDAIR_TST_STDAIR_TEST_LIB_HPP
```

Index

- ~AirlineClassList
 - stdair::AirlineClassList, [239](#)
- ~AirlineClassListKey
 - stdair::AirlineClassListKey, [243](#)
- ~AirlineFeature
 - stdair::AirlineFeature, [246](#)
- ~AirlineFeatureKey
 - stdair::AirlineFeatureKey, [249](#)
- ~AirlineStruct
 - stdair::AirlineStruct, [252](#)
- ~AirportPair
 - stdair::AirportPair, [255](#)
- ~AirportPairKey
 - stdair::AirportPairKey, [258](#)
- ~BasDBParams
 - stdair::BasDBParams, [262](#)
- ~BasLogParams
 - stdair::BasLogParams, [266](#)
- ~BomAbstract
 - stdair::BomAbstract, [270](#)
- ~BomHolder
 - stdair::BomHolder, [282](#)
- ~BomHolderKey
 - stdair::BomHolderKey, [285](#)
- ~BomRoot
 - stdair::BomRoot, [303](#)
- ~BomRootKey
 - stdair::BomRootKey, [308](#)
- ~BookingClass
 - stdair::BookingClass, [313](#)
- ~BookingClassKey
 - stdair::BookingClassKey, [324](#)
- ~BookingRequestStruct
 - stdair::BookingRequestStruct, [328](#)
- ~Bucket
 - stdair::Bucket, [333](#)
- ~BucketKey
 - stdair::BucketKey, [338](#)
- ~CancellationStruct
 - stdair::CancellationStruct, [342](#)
- ~ContinuousAttributeLite
 - stdair::ContinuousAttributeLite, [350](#)
- ~DatePeriod
 - stdair::DatePeriod, [354](#)
- ~DatePeriodKey
 - stdair::DatePeriodKey, [357](#)
- ~DbAbstract
 - stdair::DbAbstract, [359](#)
- ~DoWStruct
 - stdair::DoWStruct, [371](#)
- ~EventQueue
 - stdair::EventQueue, [378](#)
- ~EventQueueKey
 - stdair::EventQueueKey, [389](#)
- ~EventStruct
 - stdair::EventStruct, [392](#)
- ~FacAbstract
 - stdair::FacAbstract, [399](#)
- ~FacBom
 - stdair::FacBom, [400](#)
- ~FacBomManager
 - stdair::FacBomManager, [402](#)
- ~FacSTDAIRServiceContext
 - stdair::FacSTDAIRServiceContext, [409](#)
- ~FacServiceAbstract
 - stdair::FacServiceAbstract, [407](#)
- ~FacSupervisor
 - stdair::FacSupervisor, [411](#)
- ~FareFamily
 - stdair::FareFamily, [415](#)
- ~FareFamilyKey
 - stdair::FareFamilyKey, [419](#)
- ~FareFeatures
 - stdair::FareFeatures, [422](#)
- ~FareFeaturesKey
 - stdair::FareFeaturesKey, [427](#)
- ~FareOptionStruct
 - stdair::FareOptionStruct, [430](#)
- ~FlagSaver
 - stdair::FlagSaver, [434](#)
- ~FlightDate
 - stdair::FlightDate, [436](#)
- ~FlightDateKey
 - stdair::FlightDateKey, [442](#)
- ~FlightPeriod
 - stdair::FlightPeriod, [445](#)
- ~FlightPeriodKey
 - stdair::FlightPeriodKey, [448](#)
- ~GuillotineBlock
 - stdair::GuillotineBlock, [459](#)
- ~GuillotineBlockKey
 - stdair::GuillotineBlockKey, [468](#)

- ~Inventory
 - stdair::Inventory, [472](#)
- ~InventoryKey
 - stdair::InventoryKey, [477](#)
- ~KeyAbstract
 - stdair::KeyAbstract, [481](#)
- ~LegCabin
 - stdair::LegCabin, [486](#)
- ~LegCabinKey
 - stdair::LegCabinKey, [500](#)
- ~LegDate
 - stdair::LegDate, [504](#)
- ~LegDateKey
 - stdair::LegDateKey, [512](#)
- ~OnDDate
 - stdair::OnDDate, [527](#)
- ~OnDDateKey
 - stdair::OnDDateKey, [533](#)
- ~OptimisationNotificationStruct
 - stdair::OptimisationNotificationStruct, [536](#)
- ~ParsedKey
 - stdair::ParsedKey, [540](#)
- ~PeriodStruct
 - stdair::PeriodStruct, [554](#)
- ~PosChannel
 - stdair::PosChannel, [558](#)
- ~PosChannelKey
 - stdair::PosChannelKey, [561](#)
- ~ProgressStatusSet
 - stdair::ProgressStatusSet, [568](#)
- ~RMEventStruct
 - stdair::RMEventStruct, [575](#)
- ~RandomGeneration
 - stdair::RandomGeneration, [572](#)
- ~RootException
 - stdair::RootException, [578](#)
- ~RootFilePath
 - stdair::RootFilePath, [580](#)
- ~SKeymap
 - swift::SKeymap, [626](#)
- ~SReadline
 - swift::SReadline, [633](#)
- ~STDAIR_Service
 - stdair::STDAIR_Service, [639](#)
- ~SegmentCabin
 - stdair::SegmentCabin, [587](#)
- ~SegmentCabinKey
 - stdair::SegmentCabinKey, [596](#)
- ~SegmentDate
 - stdair::SegmentDate, [599](#)
- ~SegmentDateKey
 - stdair::SegmentDateKey, [607](#)
- ~SegmentPeriod
 - stdair::SegmentPeriod, [611](#)
- ~SegmentPeriodKey
 - stdair::SegmentPeriodKey, [616](#)
- ~ServiceAbstract
 - stdair::ServiceAbstract, [620](#)
- ~SnapshotStruct
 - stdair::SnapshotStruct, [628](#)
- ~StructAbstract
 - stdair::StructAbstract, [652](#)
- ~TimePeriod
 - stdair::TimePeriod, [654](#)
- ~TimePeriodKey
 - stdair::TimePeriodKey, [658](#)
- ~TravelSolutionStruct
 - stdair::TravelSolutionStruct, [660](#)
- ~VirtualClassStruct
 - stdair::VirtualClassStruct, [668](#)
- ~YieldFeatures
 - stdair::YieldFeatures, [671](#)
- ~YieldFeaturesKey
 - stdair::YieldFeaturesKey, [675](#)
- ~YieldRange
 - stdair::YieldRange, [678](#)
- ~YieldStore
 - stdair::YieldStore, [681](#)
- ~YieldStoreKey
 - stdair::YieldStoreKey, [684](#)
- _acp
 - stdair::LegCabin, [499](#)
- _airlineCode
 - stdair::ParsedKey, [542](#)
- _airlineFeature
 - stdair::Inventory, [476](#)
- _au
 - stdair::BookingClass, [321](#)
 - stdair::LegCabin, [498](#)
- _availability
 - stdair::Bucket, [337](#)
 - stdair::LegCabin, [497](#)
- _availabilityPool
 - stdair::LegCabin, [497](#)
 - stdair::SegmentCabin, [594](#)
- _availabilitySnapshotBlock
 - stdair::GuillotineBlock, [467](#)
- _bidPriceVector
 - stdair::LegCabin, [497](#)

- stdair::SegmentCabin, [594](#)
- _blockSpace
 - stdair::SegmentCabin, [593](#)
- _boardingDate
 - stdair::LegDate, [510](#)
 - stdair::SegmentDate, [605](#)
- _boardingDateOffset
 - stdair::SegmentPeriod, [615](#)
- _boardingPoint
 - stdair::ParsedKey, [542](#)
- _boardingTime
 - stdair::LegDate, [510](#)
 - stdair::ParsedKey, [543](#)
 - stdair::SegmentDate, [605](#)
 - stdair::SegmentPeriod, [615](#)
- _bomList
 - stdair::BomHolder, [283](#)
- _bomMap
 - stdair::BomHolder, [284](#)
- _bookingClass
 - stdair_test::Cabin, [341](#)
- _bookingCounter
 - stdair::SegmentCabin, [593](#)
- _bookingSnapshotBlock
 - stdair::GuillotineBlock, [466](#)
- _cabinBookingClassMap
 - stdair::SegmentPeriod, [615](#)
- _cabinForecastMap
 - stdair::OnDDate, [532](#)
- _cancellationPercentage
 - stdair::BookingClass, [321](#)
- _cancellationSnapshotBlock
 - stdair::GuillotineBlock, [466](#)
- _capacity
 - stdair::LegDate, [511](#)
 - stdair::SegmentCabin, [593](#)
- _classCode
 - stdair_test::BookingClass, [310](#)
- _classPathDemandMap
 - stdair::OnDDate, [531](#)
- _committedSpace
 - stdair::LegCabin, [497](#)
 - stdair::SegmentCabin, [594](#)
- _controlMode
 - stdair::AirlineFeature, [248](#)
- _cumulatedBookingLimit
 - stdair::BookingClass, [321](#)
- _cumulatedProtection
 - stdair::BookingClass, [320](#)
- _currentBidPrice
 - stdair::LegCabin, [497](#)
 - stdair::SegmentCabin, [594](#)
- _dcsRegrade
 - stdair::LegCabin, [498](#)
- _departureDate
 - stdair::ParsedKey, [542](#)
- _distance
 - stdair::LegDate, [511](#)
 - stdair::SegmentDate, [606](#)
- _elapsedTime
 - stdair::LegDate, [510](#)
 - stdair::SegmentDate, [605](#)
 - stdair::SegmentPeriod, [615](#)
- _etb
 - stdair::BookingClass, [322](#)
 - stdair::LegCabin, [499](#)
- _eventList
 - stdair::EventQueue, [386](#)
- _fare
 - stdair::AirlineClassList, [242](#)
- _fareFamilyActivation
 - stdair::SegmentCabin, [594](#)
- _filename
 - stdair::InputFilePath, [471](#)
 - stdair::RootFilePath, [580](#)
- _flightNumber
 - stdair::ParsedKey, [542](#)
- _forecasterMode
 - stdair::AirlineFeature, [248](#)
- _fullKey
 - stdair::ParsedKey, [542](#)
- _gav
 - stdair::LegCabin, [498](#)
- _generatedDemandVector
 - stdair::BookingClass, [323](#)
- _generator
 - stdair::RandomGeneration, [574](#)
- _groupNbOfBookings
 - stdair::BookingClass, [322](#)
 - stdair::LegCabin, [499](#)
- _groupPendingNbOfBookings
 - stdair::BookingClass, [322](#)
- _guillotineBlock
 - stdair::SegmentCabin, [593](#)
- _historicalDataLimit
 - stdair::AirlineFeature, [248](#)
- _holderMap
 - stdair::AirlineClassList, [242](#)
 - stdair::AirportPair, [257](#)
 - stdair::BomRoot, [306](#)

- stdair::BookingClass, [320](#)
- stdair::Bucket, [337](#)
- stdair::DatePeriod, [356](#)
- stdair::EventQueue, [386](#)
- stdair::FareFamily, [417](#)
- stdair::FareFeatures, [425](#)
- stdair::FlightDate, [440](#)
- stdair::FlightPeriod, [447](#)
- stdair::GuillotineBlock, [466](#)
- stdair::Inventory, [476](#)
- stdair::LegCabin, [496](#)
- stdair::LegDate, [510](#)
- stdair::OnDDate, [531](#)
- stdair::PosChannel, [560](#)
- stdair::SegmentCabin, [593](#)
- stdair::SegmentDate, [604](#)
- stdair::SegmentPeriod, [615](#)
- stdair::TimePeriod, [657](#)
- stdair::YieldFeatures, [674](#)
- _key
 - stdair::AirlineClassList, [241](#)
 - stdair::AirlineFeature, [248](#)
 - stdair::AirportPair, [257](#)
 - stdair::BomHolder, [283](#)
 - stdair::BomRoot, [306](#)
 - stdair::BookingClass, [320](#)
 - stdair::Bucket, [336](#)
 - stdair::DatePeriod, [356](#)
 - stdair::EventQueue, [386](#)
 - stdair::FareFamily, [417](#)
 - stdair::FareFeatures, [425](#)
 - stdair::FlightDate, [440](#)
 - stdair::FlightPeriod, [447](#)
 - stdair::GuillotineBlock, [465](#)
 - stdair::Inventory, [476](#)
 - stdair::LegCabin, [496](#)
 - stdair::LegDate, [509](#)
 - stdair::OnDDate, [531](#)
 - stdair::PosChannel, [560](#)
 - stdair::SegmentCabin, [592](#)
 - stdair::SegmentDate, [604](#)
 - stdair::SegmentPeriod, [614](#)
 - stdair::TimePeriod, [657](#)
 - stdair::YieldFeatures, [674](#)
 - stdair::YieldStore, [682](#)
- _logLevels
 - stdair::LOG, [236](#)
- _marketingSegmentDateList
 - stdair::SegmentDate, [605](#)
- _mean
 - stdair::BookingClass, [323](#)
- _min
 - stdair::SegmentCabin, [593](#)
- _nav
 - stdair::LegCabin, [498](#)
- _nbOfBookings
 - stdair::BookingClass, [321](#)
- _nbOfCancellations
 - stdair::BookingClass, [322](#)
- _nego
 - stdair::BookingClass, [321](#)
- _netClassAvailability
 - stdair::BookingClass, [323](#)
- _netRevenueAvailability
 - stdair::BookingClass, [323](#)
- _noShowPercentage
 - stdair::BookingClass, [321](#)
- _offDate
 - stdair::LegDate, [510](#)
 - stdair::SegmentDate, [605](#)
- _offDateOffset
 - stdair::SegmentPeriod, [615](#)
- _offPoint
 - stdair::LegDate, [510](#)
 - stdair::ParsedKey, [542](#)
- _offTime
 - stdair::LegDate, [510](#)
 - stdair::SegmentDate, [605](#)
 - stdair::SegmentPeriod, [615](#)
- _offeredCapacity
 - stdair::LegCabin, [496](#)
- _operatingSegmentDate
 - stdair::SegmentDate, [604](#)
- _parent
 - stdair::AirlineClassList, [242](#)
 - stdair::AirportPair, [257](#)
 - stdair::BookingClass, [320](#)
 - stdair::Bucket, [336](#)
 - stdair::DatePeriod, [356](#)
 - stdair::EventQueue, [386](#)
 - stdair::FareFamily, [417](#)
 - stdair::FareFeatures, [425](#)
 - stdair::FlightDate, [440](#)
 - stdair::FlightPeriod, [447](#)
 - stdair::GuillotineBlock, [465](#)
 - stdair::Inventory, [476](#)
 - stdair::LegCabin, [496](#)
 - stdair::LegDate, [509](#)
 - stdair::OnDDate, [531](#)
 - stdair::PosChannel, [560](#)

- stdair::SegmentCabin, [592](#)
- stdair::SegmentDate, [604](#)
- stdair::SegmentPeriod, [615](#)
- stdair::TimePeriod, [657](#)
- stdair::YieldFeatures, [674](#)
- stdair::YieldStore, [683](#)
- _physicalCapacity
 - stdair::LegCabin, [496](#)
- _pool
 - stdair::FacServiceAbstract, [407](#)
 - stdair::FacSTDAIRServiceContext, [410](#)
- _previousBidPrice
 - stdair::LegCabin, [497](#)
- _productAndPriceOrientedBookingSnapshotBlock
 - stdair::GuillotineBlock, [467](#)
- _progressStatus
 - stdair::EventQueue, [386](#)
- _progressStatusMap
 - stdair::EventQueue, [386](#)
- _protection
 - stdair::BookingClass, [321](#)
- _segmentAvailability
 - stdair::BookingClass, [323](#)
- _segmentCabinIndexMap
 - stdair::GuillotineBlock, [466](#)
- _soldSeat
 - stdair::LegCabin, [497](#)
- _soldSeats
 - stdair::Bucket, [337](#)
- _staffNbOfBookings
 - stdair::BookingClass, [322](#)
 - stdair::LegCabin, [499](#)
- _stdDev
 - stdair::BookingClass, [323](#)
- _stringCabinClassPairListMap
 - stdair::OnDDate, [532](#)
- _subclassCode
 - stdair::BookingClass, [320](#)
- _upr
 - stdair::LegCabin, [498](#)
 - stdair::SegmentCabin, [593](#)
- _value
 - stdair::date_time_element, [352](#)
- _valueTypesIndexMap
 - stdair::GuillotineBlock, [466](#)
- _virtualClassList
 - stdair::LegCabin, [498](#)
- _what
 - stdair::CodeConversionException, [346](#)
 - stdair::CodeDuplicationException, [347](#)
- stdair::DocumentNotFoundException, [370](#)
- stdair::EventException, [375](#)
- stdair::EventQueueException, [388](#)
- stdair::FileNotFoundException, [433](#)
- stdair::KeyNotFoundException, [483](#)
- stdair::MemoryAllocationException, [515](#)
- stdair::NonInitialisedContainerException, [516](#)
- stdair::NonInitialisedDBSessionManagerException, [518](#)
- stdair::NonInitialisedLogServiceException, [519](#)
- stdair::NonInitialisedRelationshipException, [520](#)
- stdair::NonInitialisedServiceException, [521](#)
- stdair::ObjectCreationDuplicationException, [523](#)
- stdair::ObjectLinkingException, [524](#)
- stdair::ObjectNotFoundException, [525](#)
- stdair::ParserException, [544](#)
- stdair::ParsingFileFailedException, [545](#)
- stdair::RootException, [578](#)
- stdair::SerialisationException, [619](#)
- stdair::SQLDatabaseConnectionImpossibleException, [630](#)
- stdair::SQLDatabaseException, [632](#)
- _wNbOfBookings
 - stdair::BookingClass, [322](#)
 - stdair::LegCabin, [499](#)
- _yield
 - stdair::AirlineClassList, [242](#)
 - stdair::BookingClass, [323](#)
- _yieldLevelDemandMap
 - stdair::LegCabin, [498](#)
- _yieldRangeUpperValue
 - stdair::Bucket, [337](#)
- A4P
 - stdair::SampleType, [582](#)
- A_RMC
 - stdair::PartnershipTechnique, [547](#)
- Abstract part of the Business Object Model (BOM), [149](#)
- activateFareFamily
 - stdair::SegmentCabin, [591](#)
- ADD_PK
 - stdair::ForecastingMethod, [454](#)
- addBidPriceVector

- stdair::TravelSolutionStruct, 662
- addBomHolder
 - stdair::FacBomManager, 403
- addCabinBookingClassList
 - stdair::SegmentPeriod, 613
- addClassAvailabilityMap
 - stdair::TravelSolutionStruct, 662
- addClassBpvMap
 - stdair::TravelSolutionStruct, 662
- addClassList
 - stdair::FareOptionStruct, 431
- addClassYieldMap
 - stdair::TravelSolutionStruct, 662
- addDateOffset
 - stdair::PeriodStruct, 555
- addDemandInformation
 - stdair::LegCabin, 495
- addEvent
 - stdair::EventQueue, 383
- addFareOption
 - stdair::TravelSolutionStruct, 662
- addSegment
 - stdair::TravelSolutionStruct, 662
- addStatus
 - stdair::EventQueue, 384
- addToList
 - stdair::FacBomManager, 403, 406
- addToListAndMap
 - stdair::FacBomManager, 404
- addToMap
 - stdair::FacBomManager, 403, 404
- addVirtualClass
 - stdair::LegCabin, 495
- AIRLINE_CODE_BA
 - stdair, 210, 228
- AirlineClassList
 - stdair::AirlineClassList, 239
- AirlineClassListDetailedList_T
 - stdair, 180
- AirlineClassListKey
 - stdair::AirlineClassListKey, 243
- AirlineClassListKey::serialize< ba::text_iarchive>
 - >
 - stdair, 219
- AirlineClassListKey::serialize< ba::text_oarchive>
 - >
 - stdair, 219
- AirlineClassListList_T
 - stdair, 179
- AirlineClassListMap_T
 - stdair, 180
- AirlineClassListWithKey_T
 - stdair, 180
- AirlineCode_T
 - stdair, 191
- AirlineCodeList_T
 - stdair, 200
- AirlineFeature
 - stdair::AirlineFeature, 246
- AirlineFeatureKey
 - stdair::AirlineFeatureKey, 249
- AirlineFeatureList_T
 - stdair, 180
- AirlineFeatureMap_T
 - stdair, 180
- AirlinePreferenceId_T
 - stdair, 197
- AirlineStruct
 - stdair::AirlineStruct, 251
- AIRPORT_BKK
 - stdair, 211, 229
- AIRPORT_LHR
 - stdair, 209, 227
- AIRPORT_SIN
 - stdair, 211, 229
- AIRPORT_SYD
 - stdair, 209, 228
- AirportCode_T
 - stdair, 190
- AirportPair
 - stdair::AirportPair, 255
- AirportPairDetailedList_T
 - stdair, 180
- AirportPairKey
 - stdair::AirportPairKey, 258
- AirportPairList_T
 - stdair, 180
- AirportPairMap_T
 - stdair, 180
- AirportPairWithKey_T
 - stdair, 180
- Archive
 - stdair::SampleType, 582
- AlmostEquals
 - FloatingPoint, 452
- archive
 - stdair::BomArchive, 271, 272
- AuthorizationLevel_T
 - stdair, 201
- Availability_T

- stdair, [193](#)
- AvailabilityStatus_T
 - stdair, [201](#)
- BasChronometer
 - stdair::BasChronometer, [260](#)
- BasDBParams
 - stdair::BasDBParams, [261](#), [262](#)
- base_iterator_t
 - stdair, [178](#)
- base_type
 - soci::type_conversion< stdair::AirlineStruct >, [664](#)
- BaseGenerator_T
 - stdair, [205](#)
- BasLogParams
 - stdair::BasLogParams, [266](#)
- batches/ Directory Reference, [151](#)
- batches/stdair.cpp, [685](#)
- BidPrice_T
 - stdair, [203](#)
- BidPriceVector_T
 - stdair, [203](#)
- BidPriceVectorHolder_T
 - stdair, [189](#)
- Bind
 - swift::SKeymap, [626](#)
- BINDIR
 - stdair-paths.hpp, [1169](#)
- Bits
 - FloatingPoint, [450](#)
- bits
 - FloatingPoint, [451](#)
- BKG_REQ
 - stdair::EventType, [396](#)
- BlockIndex_T
 - stdair, [205](#)
- BlockNumber_T
 - stdair, [205](#)
- BlockSpace_T
 - stdair, [201](#)
- BomAbstract
 - stdair::BomAbstract, [270](#)
- BomAbstract.hpp
 - operator<<, [808](#)
 - operator>>, [808](#)
- BomFactoryPool_T
 - stdair::FacSupervisor, [411](#)
- BomHolder
 - stdair::BomHolder, [282](#)
- BomHolderKey
 - stdair::BomHolderKey, [285](#)
- BomList_T
 - stdair::BomHolder, [281](#)
- BomMap_T
 - stdair::BomHolder, [281](#)
- BomRoot
 - stdair::BomRoot, [303](#)
- BomRoot::serialize< ba::text_iarchive >
 - stdair, [222](#)
- BomRoot::serialize< ba::text_oarchive >
 - stdair, [222](#)
- BomRootKey
 - stdair::BomRootKey, [307](#), [308](#)
- BomRootKey::serialize< ba::text_iarchive >
 - stdair, [220](#)
- BomRootKey::serialize< ba::text_oarchive >
 - stdair, [220](#)
- BookingClass
 - stdair::BookingClass, [313](#)
 - stdair_test::BookingClass, [310](#)
- BookingClassKey
 - stdair::BookingClassKey, [324](#)
- BookingClassList_T
 - stdair, [181](#)
- BookingClassMap_T
 - stdair, [181](#)
- BookingLimit_T
 - stdair, [201](#)
- BookingLimitVector_T
 - stdair, [204](#)
- BookingRatio_T
 - stdair, [202](#)
- BookingRequestPtr_T
 - stdair, [181](#)
- BookingRequestStruct
 - stdair::BookingRequestStruct, [327](#)
- BookingTSIDMap_T
 - stdair, [198](#)
- BooleanList_T
 - stdair::DoWStruct, [371](#)
- boost, [158](#)
- boost::serialization, [158](#)
- boost::serialization::access
 - stdair::AirlineClassList, [241](#)
 - stdair::AirlineClassListKey, [245](#)
 - stdair::BomRoot, [306](#)
 - stdair::BomRootKey, [309](#)

- stdair::Bucket, [336](#)
- stdair::BucketKey, [340](#)
- stdair::FareFamily, [417](#)
- stdair::FareFamilyKey, [420](#)
- stdair::FlightDate, [440](#)
- stdair::FlightDateKey, [443](#)
- stdair::GuillotineBlock, [465](#)
- stdair::GuillotineBlockKey, [469](#)
- stdair::Inventory, [475](#)
- stdair::InventoryKey, [479](#)
- stdair::LegCabinKey, [502](#)
- stdair::OnDDate, [531](#)
- stdair::OnDDateKey, [535](#)
- stdair::SegmentCabin, [592](#)
- stdair::SegmentCabinKey, [597](#)
- stdair::SegmentDate, [604](#)
- stdair::SegmentDateKey, [608](#)
- BOOST_DEFAULT_DATE_PERIOD
 - stdair, [208](#), [233](#)
- bpt, [158](#)
 - ptree, [158](#)
- BRK_PT
 - stdair::EventType, [396](#)
- Bucket
 - stdair::Bucket, [333](#)
- BucketAvailabilities_T
 - stdair, [201](#)
- BucketKey
 - stdair::BucketKey, [338](#)
- BucketKey::serialize< ba::text_iarchive >
 - stdair, [221](#)
- BucketKey::serialize< ba::text_oarchive >
 - stdair, [221](#)
- BucketList_T
 - stdair, [181](#)
- BucketMap_T
 - stdair, [181](#)
- buildDummyInventory
 - stdair::STDAIR_Service, [640](#)
- buildSampleBom
 - stdair::STDAIR_Service, [639](#)
- buildSampleBookingRequest
 - stdair::STDAIR_Service, [641](#)
- buildSampleTravelSolutionForPricing
 - stdair::STDAIR_Service, [640](#)
- buildSampleTravelSolutions
 - stdair::STDAIR_Service, [641](#)
- BUILTIN_SAMPLE
 - stdair::ServiceInitialisationType, [622](#)
- BUSINESS
 - stdair::PassengerType, [551](#)
- Cabin
 - stdair_test::Cabin, [341](#)
- CABIN_ECO
 - stdair, [211](#), [229](#)
- CABIN_Y
 - stdair, [210](#), [228](#)
- CabinBookingClassMap_T
 - stdair, [200](#)
- CabinCapacity_T
 - stdair, [200](#)
- CabinClassPair_T
 - stdair, [198](#)
- CabinClassPairList_T
 - stdair, [198](#)
- CabinCode_T
 - stdair, [191](#)
- CabinForecastMap_T
 - stdair, [186](#)
- CabinForecastPair_T
 - stdair, [186](#)
- calculateProgress
 - stdair::EventQueue, [385](#)
- cancel
 - stdair::BookingClass, [319](#)
- CancellationNoShowRatePair_T
 - stdair, [197](#)
- CancellationPtr_T
 - stdair, [181](#)
- CancellationRateCurveld_T
 - stdair, [197](#)
- CancellationStruct
 - stdair::CancellationStruct, [342](#)
- CapacityAdjustment_T
 - stdair, [201](#)
- CCM
 - stdair::SampleType, [582](#)
- CensorshipFlag_T
 - stdair, [202](#)
- CensorshipFlagList_T
 - stdair, [202](#)
- CHANGE_FEES
 - stdair, [210](#), [228](#)
- ChangeFees_T
 - stdair, [196](#)
- ChangeFeesRatio_T
 - stdair, [196](#)
- CHANNEL_DN
 - stdair, [216](#), [234](#)

- CHANNEL_IN
 - stdair, [216](#), [234](#)
- ChannelLabel_T
 - stdair, [198](#)
- CharacteristicsIndex_T
 - stdair, [197](#)
- CharacteristicsPatternId_T
 - stdair, [197](#)
- CharacteristicsWTP_tuple_T
 - stdair, [197](#)
- check
 - stdair::BasDBParams, [263](#)
 - stdair::BasLogParams, [267](#)
 - stdair::date_time_element, [352](#)
 - stdair::STDAIR_Service, [646](#)
- child
 - stdair_test::Cabin, [340](#)
- CityCode_T
 - stdair, [190](#)
- CLASS_CODE_Q
 - stdair, [211](#), [229](#)
- CLASS_CODE_Y
 - stdair, [210](#), [228](#)
- ClassAvailabilityMap_T
 - stdair, [188](#)
- ClassAvailabilityMapHolder_T
 - stdair, [189](#)
- ClassBpvMap_T
 - stdair, [189](#)
- ClassBpvMapHolder_T
 - stdair, [189](#)
- ClassCode_T
 - stdair, [191](#)
- ClassCodeList_T
 - stdair, [200](#)
- ClassList_String_T
 - stdair, [193](#)
- ClassList_StringList_T
 - stdair, [200](#)
- ClassYieldMap_T
 - stdair, [189](#)
- ClassYieldMapHolder_T
 - stdair, [189](#)
- clean
 - stdair::FacBom, [401](#)
 - stdair::FacServiceAbstract, [407](#)
 - stdair::FacSTDAIRServiceContext, [410](#)
- cleanAll
 - stdair::FacSupervisor, [413](#)
- cleanBomLayer
 - stdair::FacSupervisor, [413](#)
- cleanDBSessionManager
 - stdair::FacSupervisor, [413](#)
- cleanLoggerService
 - stdair::FacSupervisor, [413](#)
- cleanServiceLayer
 - stdair::FacSupervisor, [413](#)
- ClearHistory
 - swift::SReadline, [636](#)
- cloneHolder
 - stdair::FacBomManager, [405](#)
- CodeConversionException
 - stdair::CodeConversionException, [346](#)
- CodeDuplicationException
 - stdair::CodeDuplicationException, [347](#)
- com_cd
 - readline_autocomp.hpp, [1244](#)
- com_delete
 - readline_autocomp.hpp, [1244](#)
- com_help
 - readline_autocomp.hpp, [1244](#)
- com_list
 - readline_autocomp.hpp, [1243](#)
- com_pwd
 - readline_autocomp.hpp, [1244](#)
- com_quit
 - readline_autocomp.hpp, [1244](#)
- com_rename
 - readline_autocomp.hpp, [1244](#)
- com_stat
 - readline_autocomp.hpp, [1244](#)
- com_view
 - readline_autocomp.hpp, [1244](#)
- COMMAND, [348](#)
 - doc, [348](#)
 - func, [348](#)
 - name, [348](#)
- command_generator
 - readline_autocomp.hpp, [1245](#)
- commands
 - readline_autocomp.hpp, [1246](#)
- CommittedSpace_T
 - stdair, [201](#)
- ConstSegmentCabinDTDRangeSnapshotView_-T
 - stdair, [205](#)
- ConstSegmentCabinDTDSnapshotView_T
 - stdair, [204](#)
- ContinuousAttributeLite
 - stdair::ContinuousAttributeLite, [349](#), [350](#)

- ContinuousDistribution_T
 - stdair::ContinuousAttributeLite, [349](#)
- ControlMode_T
 - stdair, [203](#)
- count
 - stdair::ProgressStatus, [565](#)
- Count_T
 - stdair, [192](#)
- create
 - stdair::FacBom, [400](#), [401](#)
 - stdair::FacSTDAIRServiceContext, [409](#)
- CRITICAL
 - stdair::LOG, [236](#)
- CRS
 - stdair::SampleType, [582](#)
- csvAirlineClassDisplay
 - stdair::BomDisplay, [280](#)
- csvAirportPairDisplay
 - stdair::BomDisplay, [278](#)
- csvBookingClassDisplay
 - stdair::BomDisplay, [277](#)
- csvBucketDisplay
 - stdair::BomDisplay, [277](#)
- csvDateDisplay
 - stdair::BomDisplay, [279](#)
- csvDisplay
 - stdair::BomDisplay, [273–275](#), [277](#), [278](#)
 - stdair::STDAIR_Service, [646](#), [647](#)
- csvFareFamilyDisplay
 - stdair::BomDisplay, [276](#)
- csvFeatureListDisplay
 - stdair::BomDisplay, [279](#)
- csvFeaturesDisplay
 - stdair::BomDisplay, [279](#)
- csvLegCabinDisplay
 - stdair::BomDisplay, [276](#)
- csvLegDateDisplay
 - stdair::BomDisplay, [275](#)
- csvPosChannelDisplay
 - stdair::BomDisplay, [279](#)
- csvSegmentCabinDisplay
 - stdair::BomDisplay, [276](#)
- csvSegmentDateDisplay
 - stdair::BomDisplay, [276](#)
- csvSimFQTAirRACDisplay
 - stdair::BomDisplay, [278](#)
- csvTimeDisplay
 - stdair::BomDisplay, [279](#)
- CX
 - stdair::EventType, [396](#)
- DATADIR
 - stdair-paths.hpp, [1170](#)
- DATAROOTDIR
 - stdair-paths.hpp, [1170](#)
- DATE_20110115
 - stdair, [209](#)
- DATE_20111231
 - stdair, [209](#)
- Date_T
 - stdair, [194](#)
- date_time_element
 - stdair::date_time_element, [352](#)
- DateOffset_T
 - stdair, [195](#)
- DatePeriod
 - stdair::DatePeriod, [354](#)
- DatePeriod_T
 - stdair, [195](#)
- DatePeriodDetailedList_T
 - stdair, [182](#)
- DatePeriodKey
 - stdair::DatePeriodKey, [357](#)
- DatePeriodList_T
 - stdair, [182](#)
- DatePeriodMap_T
 - stdair, [182](#)
- DatePeriodWithKey_T
 - stdair, [182](#)
- DateTime_T
 - stdair, [195](#)
- day_p_t
 - stdair, [179](#)
- day_t
 - stdair, [178](#)
- DayDuration_T
 - stdair, [195](#)
- DbAbstract
 - stdair::DbAbstract, [359](#)
- DbAbstract.hpp
 - operator<<, [1172](#)
 - operator>>, [1172](#)
- DBConnectionName_T
 - stdair, [196](#)
- DBRequestStatement_T
 - stdair, [196](#)
- DBSession_T
 - stdair, [196](#)
- DCP_T
 - stdair, [202](#)
- DCPList_T

- stdair, [202](#)
- DEBUG
 - stdair::LOG, [236](#)
- DEFAULT_ADVANCE_PURCHASE
 - stdair, [215](#), [233](#)
- DEFAULT_AIRLINE_CODE
 - stdair, [217](#), [231](#)
- DEFAULT_AIRLINE_CODE_LIST
 - stdair, [223](#)
- DEFAULT_AIRPORT_CODE
 - stdair, [217](#), [231](#)
- DEFAULT_AVAILABILITY
 - stdair, [212](#), [226](#)
- DEFAULT_AVAILABILITY_STATUS
 - stdair, [217](#), [235](#)
- DEFAULT_BID_PRICE
 - stdair, [218](#), [232](#)
- DEFAULT_BID_PRICE_VECTOR
 - stdair, [223](#)
- DEFAULT_BLOCK_SPACE
 - stdair, [212](#), [226](#)
- DEFAULT_BOM_ROOT_KEY
 - stdair, [208](#), [229](#)
- DEFAULT_BOM_TREE_CHANGE_FEES
 - stdair, [210](#)
- DEFAULT_BOM_TREE_NON_REFUNDABLE
 - stdair, [210](#)
- DEFAULT_BOM_TREE_SATURDAY_STAY
 - stdair, [210](#)
- DEFAULT_BOOKING_NUMBER
 - stdair, [214](#)
- DEFAULT_CABIN_CAPACITY
 - stdair, [212](#), [226](#)
- DEFAULT_CABIN_CODE
 - stdair, [218](#), [232](#)
- DEFAULT_CHANNEL
 - stdair, [216](#), [231](#)
- DEFAULT_CLASS_AUTHORIZATION_LEVEL
 - stdair, [213](#), [226](#)
- DEFAULT_CLASS_AVAILABILITY
 - stdair, [235](#)
- DEFAULT_CLASS_BOOKING_LIMIT
 - stdair, [213](#), [226](#)
- DEFAULT_CLASS_CENSORSHIPFLAG
 - stdair, [213](#), [226](#)
- DEFAULT_CLASS_CENSORSHIPFLAG_LIST
 - stdair, [223](#)
- DEFAULT_CLASS_CODE
 - stdair, [218](#), [232](#)
- DEFAULT_CLASS_CODE_LIST
 - stdair, [223](#)
- DEFAULT_CLASS_FARE_VALUE
 - stdair, [230](#)
- DEFAULT_CLASS_MAX_AUTHORIZATION_LEVEL
 - stdair, [213](#), [226](#)
- DEFAULT_CLASS_MIN_AUTHORIZATION_LEVEL
 - stdair, [213](#), [227](#)
- DEFAULT_CLASS_NB_OF_BOOKINGS
 - stdair, [211](#), [225](#)
- DEFAULT_CLASS_NB_OF_CANCELLATIONS
 - stdair, [212](#), [226](#)
- DEFAULT_CLASS_NB_OF_NOSHOWS
 - stdair, [212](#), [226](#)
- DEFAULT_CLASS_OVERBOOKING_RATE
 - stdair, [213](#), [227](#)
- DEFAULT_CLASS_REMAINING_DEMAND_MEAN
 - stdair, [212](#), [225](#)
- DEFAULT_CLASS_REMAINING_DEMAND_STANDARD_DEVIATION
 - stdair, [212](#), [225](#)
- DEFAULT_CLASS_TOTAL_NB_OF_BOOKINGS
 - stdair, [211](#), [225](#)
- DEFAULT_CLASS_UNCONSTRAINED_DEMAND
 - stdair, [212](#), [225](#)
- DEFAULT_CLASS_YIELD_VALUE
 - stdair, [213](#)
- DEFAULT_CLOSED_CLASS_CODE
 - stdair, [211](#), [225](#)
- DEFAULT_COMMITTED_SPACE
 - stdair, [212](#), [226](#)
- DEFAULT_CURRENCY
 - stdair, [217](#), [227](#)
- DEFAULT_DATE
 - stdair, [209](#), [230](#)
- DEFAULT_DATE_OFFSET
 - stdair, [209](#), [233](#)
- DEFAULT_DATETIME
 - stdair, [209](#), [230](#)
- DEFAULT_DAY_DURATION
 - stdair, [208](#), [227](#)
- DEFAULT_DCP_LIST
 - stdair, [224](#)
- DEFAULT_DEPARTURE_DATE
 - stdair, [217](#), [231](#)
- DEFAULT_DESTINATION
 - stdair, [217](#), [232](#)

- DEFAULT_DICO_STUDIED_AIRLINE
stdair, [235](#)
- DEFAULT_DICO_STUDIED_DATE
stdair, [223](#)
- DEFAULT_DISTANCE_VALUE
stdair, [211](#), [225](#)
- DEFAULT_DOW_STRING
stdair, [208](#), [233](#)
- DEFAULT_DTD_FRAT5COEF_MAP
stdair, [224](#)
- DEFAULT_DTD_PROB_MAP
stdair, [224](#)
- DEFAULT_EPSILON_DURATION
stdair, [209](#), [230](#)
- DEFAULT_EPSILON_VALUE
stdair, [208](#), [227](#)
- DEFAULT_EVENT_OLDEST_DATE
stdair, [215](#), [229](#)
- DEFAULT_EVENT_OLDEST_DATETIME
stdair, [215](#), [229](#)
- DEFAULT_EVENT_QUEUE_ID
stdair, [214](#), [229](#)
- DEFAULT_FAMILY_CODE
stdair, [211](#)
- DEFAULT_FARE_FAMILY_CODE
stdair, [218](#), [232](#)
- DEFAULT_FARE_FAMILY_VALUE_TYPE
stdair, [218](#), [232](#)
- DEFAULT_FARE_VALUE
stdair, [213](#), [227](#)
- DEFAULT_FF_TIER
stdair, [216](#), [234](#)
- DEFAULT_FLIGHT_NUMBER
stdair, [217](#), [231](#)
- DEFAULT_FLIGHT_SPEED
stdair, [208](#), [230](#)
- DEFAULT_FLIGHTPATH_CODE
stdair, [211](#), [235](#)
- DEFAULT_GUILLOTINE_NUMBER
stdair, [217](#), [231](#)
- DEFAULT_KEY_FLD_DELIMITER
stdair, [219](#), [224](#)
- DEFAULT_KEY_SUB_FLD_DELIMITER
stdair, [219](#), [224](#)
- DEFAULT_KEY_TOKEN_DELIMITER
stdair, [219](#), [225](#)
- DEFAULT_LOAD_FACTOR_VALUE
stdair, [213](#), [227](#)
- DEFAULT_MATCHING_INDICATOR
stdair, [217](#), [235](#)
- DEFAULT_MAX_DTD
stdair, [223](#)
- DEFAULT_MAXIMAL_CONNECTION_TIME
stdair, [216](#), [234](#)
- DEFAULT_MINIMAL_CONNECTION_TIME
stdair, [216](#), [234](#)
- DEFAULT_NB_OF_DAYS_IN_A_YEAR
stdair, [208](#), [230](#)
- DEFAULT_NB_OF_FLIGHTDATES
stdair, [208](#), [229](#)
- DEFAULT_NBOFAIRLINES
stdair, [211](#), [230](#)
- DEFAULT_NULL_AIRLINE_CODE
stdair, [217](#), [231](#)
- DEFAULT_NULL_AIRPORT_CODE
stdair, [217](#), [231](#)
- DEFAULT_NULL_AVAILABILITY
stdair, [212](#), [226](#)
- DEFAULT_NULL_CLASS_CODE
stdair, [218](#), [232](#)
- DEFAULT_NULL_FARE_FAMILY_CODE
stdair, [218](#), [232](#)
- DEFAULT_NUMBER_OF_REQUIRED_SEATS
stdair, [235](#)
- DEFAULT_NUMBER_OF_SUBDIVISIONS
stdair, [208](#), [231](#)
- DEFAULT_OND_BOOKING_RATE
stdair, [213](#), [230](#)
- DEFAULT_OND_FARE_VALUE
stdair, [214](#)
- DEFAULT_OND_STRING_LIST
stdair, [224](#)
- DEFAULT_ORIGIN
stdair, [217](#), [231](#)
- DEFAULT_PARTY_SIZE
stdair, [215](#), [233](#)
- DEFAULT_POS
stdair, [216](#), [233](#)
- DEFAULT_PREFERRED_CABIN
stdair, [216](#), [234](#)
- DEFAULT_PREFERRED_DEPARTURE_DATE
stdair, [215](#), [233](#)
- DEFAULT_PREFERRED_DEPARTURE_TIME
stdair, [215](#), [233](#)
- DEFAULT_PROGRESS_STATUS
stdair, [214](#), [229](#)
- DEFAULT_RANDOM_SEED
stdair, [209](#), [230](#)
- DEFAULT_REQUEST_DATE
stdair, [215](#), [233](#)

- DEFAULT_REQUEST_DATE_TIME
 - stdair, [215](#), [234](#)
- DEFAULT_REQUEST_TIME
 - stdair, [215](#), [233](#)
- DEFAULT_REVENUE_VALUE
 - stdair, [213](#), [227](#)
- DEFAULT_SEAT_INDEX
 - stdair, [218](#), [232](#)
- DEFAULT_SEGMENT_CABIN_VALUE_TYPE
 - stdair, [218](#), [233](#)
- DEFAULT_STAY_DURATION
 - stdair, [215](#), [233](#)
- DEFAULT_VALUE_OF_TIME
 - stdair, [216](#), [234](#)
- DEFAULT_WTP
 - stdair, [215](#), [233](#)
- DEFAULT_YIELD_AVAILABILITY
 - stdair, [214](#)
- DEFAULT_YIELD_BOOKING_LIMIT
 - stdair, [214](#)
- DEFAULT_YIELD_CENSORSHIPFLAG
 - stdair, [214](#)
- DEFAULT_YIELD_MAX_VALUE
 - stdair, [214](#), [235](#)
- DEFAULT_YIELD_NB_OF_BOOKINGS
 - stdair, [214](#)
- DEFAULT_YIELD_NB_OF_CANCELLATIONS
 - stdair, [214](#)
- DEFAULT_YIELD_NB_OF_NOSHOWS
 - stdair, [214](#)
- DEFAULT_YIELD_OVERBOOKING_RATE
 - stdair, [214](#)
- DEFAULT_YIELD_VALUE
 - stdair, [214](#), [235](#)
- DEM
 - stdair::SampleType, [582](#)
- DemandGenerationMethod
 - stdair::DemandGenerationMethod, [365](#), [366](#)
- DemandGeneratorKey_T
 - stdair, [181](#)
- DemandStreamKeyStr_T
 - stdair, [198](#)
- describe
 - stdair::AirlineStruct, [253](#)
 - stdair::BasDBParams, [263](#)
 - stdair::BasLogParams, [267](#)
 - stdair::BomHolderKey, [286](#)
 - stdair::BookingRequestStruct, [330](#)
 - stdair::CancellationStruct, [343](#)
 - stdair::DemandGenerationMethod, [367](#)
 - stdair::DoWStruct, [372](#)
 - stdair::EventStruct, [394](#)
 - stdair::EventType, [397](#)
 - stdair::FareOptionStruct, [432](#)
 - stdair::ForecastingMethod, [455](#)
 - stdair::OptimisationNotificationStruct, [539](#)
 - stdair::PartnershipTechnique, [549](#)
 - stdair::PassengerType, [552](#)
 - stdair::PeriodStruct, [555](#)
 - stdair::ProgressStatus, [566](#)
 - stdair::ProgressStatusSet, [570](#)
 - stdair::RandomGeneration, [573](#)
 - stdair::RMEventStruct, [576](#)
 - stdair::SampleType, [583](#)
 - stdair::ServiceInitialisationType, [624](#)
 - stdair::SnapshotStruct, [629](#)
 - stdair::StructAbstract, [652](#)
 - stdair::TravelSolutionStruct, [663](#)
 - stdair::VirtualClassStruct, [670](#)
 - stdair::YieldRange, [679](#)
- describeKey
 - stdair::AirlineClassList, [241](#)
 - stdair::AirlineFeature, [248](#)
 - stdair::AirportPair, [255](#)
 - stdair::BomHolder, [283](#)
 - stdair::BomRoot, [305](#)
 - stdair::BookingClass, [319](#)
 - stdair::Bucket, [336](#)
 - stdair::DatePeriod, [355](#)
 - stdair::EventQueue, [382](#)
 - stdair::FareFamily, [416](#)
 - stdair::FareFeatures, [422](#)
 - stdair::FlightDate, [439](#)
 - stdair::FlightPeriod, [446](#)
 - stdair::GuillotineBlock, [465](#)
 - stdair::Inventory, [475](#)
 - stdair::LegCabin, [494](#)
 - stdair::LegDate, [509](#)
 - stdair::OnDDate, [530](#)
 - stdair::PosChannel, [559](#)
 - stdair::SegmentCabin, [592](#)
 - stdair::SegmentDate, [603](#)
 - stdair::SegmentPeriod, [614](#)
 - stdair::TimePeriod, [655](#)
 - stdair::YieldFeatures, [672](#)
 - stdair::YieldStore, [682](#)
- describeLabels
 - stdair::DemandGenerationMethod, [366](#)

- stdair::EventType, [397](#)
- stdair::ForecastingMethod, [455](#)
- stdair::PartnershipTechnique, [548](#)
- stdair::PassengerType, [552](#)
- stdair::SampleType, [583](#)
- stdair::ServiceInitialisationType, [623](#)
- describeShort
 - stdair::DoWStruct, [372](#)
 - stdair::PeriodStruct, [555](#)
- DictionaryKey_T
 - stdair, [179](#)
- display
 - stdair::BookingRequestStruct, [330](#)
 - stdair::CancellationStruct, [343](#)
 - stdair::EventQueue, [382](#)
 - stdair::FareOptionStruct, [432](#)
 - stdair::TravelSolutionStruct, [663](#)
- DISPLAY_LEVEL_STRING_ARRAY
 - stdair, [224](#)
- displayCumulativeDistribution
 - stdair::ContinuousAttributeLite, [351](#)
- displayVirtualClassList
 - stdair::LegCabin, [495](#)
- Distance_T
 - stdair, [190](#)
- DistributionPatternId_T
 - stdair, [197](#)
- doc
 - COMMAND, [348](#)
- doc/local/authors.doc, [690](#)
- doc/local/codingrules.doc, [690](#)
- doc/local/copyright.doc, [690](#)
- doc/local/documentation.doc, [690](#)
- doc/local/features.doc, [690](#)
- doc/local/help_wanted.doc, [690](#)
- doc/local/howto_release.doc, [690](#)
- doc/local/index.doc, [690](#)
- doc/local/installation.doc, [690](#)
- doc/local/linking.doc, [690](#)
- doc/local/test.doc, [690](#)
- doc/local/users_guide.doc, [690](#)
- doc/local/verification.doc, [690](#)
- doc/tutorial/tutorial.doc, [690](#)
- DOCDIR
 - stdair-paths.hpp, [1170](#)
- DocumentNotFoundException
 - stdair::DocumentNotFoundException, [370](#)
- doesExistAndIsReadable
 - stdair::BasFileMgr, [265](#)
- done
 - readline_autocomp.hpp, [1246](#)
- DOW_STR
 - stdair, [222](#)
- DOW_String_T
 - stdair, [195](#)
- DoWStruct
 - stdair::DoWStruct, [371](#)
- DTD_T
 - stdair, [202](#)
- DTDFratMap_T
 - stdair, [202](#)
- DTDProbMap_T
 - stdair, [202](#)
- dupstr
 - readline_autocomp.hpp, [1245](#)
- Duration_T
 - stdair, [194](#)
- elapsed
 - stdair::BasChronometer, [260](#)
- emptyBidPriceVector
 - stdair::LegCabin, [495](#)
- emptyClassList
 - stdair::FareOptionStruct, [431](#)
- emptyVirtualClassList
 - stdair::LegCabin, [495](#)
- emptyYieldLevelDemandMap
 - stdair::LegCabin, [496](#)
- EmsrValueList_T
 - stdair, [204](#)
- EN_DemandGenerationMethod
 - stdair::DemandGenerationMethod, [365](#)
- EN_EventType
 - stdair::EventType, [396](#)
- EN_ForecastingMethod
 - stdair::ForecastingMethod, [454](#)
- EN_LogLevel
 - stdair::LOG, [236](#)
- EN_PartnershipTechnique
 - stdair::PartnershipTechnique, [547](#)
- EN_PassengerType
 - stdair::PassengerType, [551](#)
- EN_SampleType
 - stdair::SampleType, [582](#)
- EN_ServiceInitialisationType
 - stdair::ServiceInitialisationType, [622](#)
- ERROR
 - stdair::LOG, [236](#)
- EventException

- stdair::EventException, [374](#)
- EventGeneratorKey_T
 - stdair, [199](#)
- EventList_T
 - stdair, [182](#)
- EventListElement_T
 - stdair, [182](#)
- EventName_T
 - stdair, [199](#)
- EventQueue
 - stdair::EventQueue, [378](#)
 - stdair::EventStruct, [394](#)
- EventQueueException
 - stdair::EventQueueException, [387](#)
- EventQueueID_T
 - stdair, [199](#)
- EventQueueKey
 - stdair::EventQueueKey, [389](#)
- EventQueueList_T
 - stdair, [182](#)
- EventQueueMap_T
 - stdair, [182](#)
- EventStruct
 - stdair::EventStruct, [391](#), [392](#)
- EventType
 - stdair::EventType, [396](#)
- EVT
 - stdair::SampleType, [582](#)
- EXEC_PREFIX
 - stdair-paths.hpp, [1169](#)
- execute_line
 - readline_autocomp.hpp, [1245](#)
- exponent_bits
 - FloatingPoint, [451](#)
- ExponentialDistribution_T
 - stdair, [206](#)
- ExponentialGenerator_T
 - stdair, [206](#)
- ExponentialSeed_T
 - stdair, [205](#)
- extractFlightDateKey
 - stdair::BomKeyManager, [289](#)
- extractInventoryKey
 - stdair::BomKeyManager, [288](#)
- extractKeys
 - stdair::BomKeyManager, [288](#)
- extractSegmentDateKey
 - stdair::BomKeyManager, [289](#)
- FacAbstract
 - stdair::FacAbstract, [399](#)
- FacBom
 - stdair::AirlineClassList, [241](#)
 - stdair::AirlineFeature, [248](#)
 - stdair::AirportPair, [256](#)
 - stdair::BomHolder, [283](#)
 - stdair::BomRoot, [306](#)
 - stdair::BookingClass, [320](#)
 - stdair::Bucket, [336](#)
 - stdair::DatePeriod, [356](#)
 - stdair::EventQueue, [385](#)
 - stdair::FacBom, [400](#)
 - stdair::FareFamily, [417](#)
 - stdair::FareFeatures, [425](#)
 - stdair::FlightDate, [440](#)
 - stdair::FlightPeriod, [447](#)
 - stdair::GuillotineBlock, [465](#)
 - stdair::Inventory, [475](#)
 - stdair::LegCabin, [496](#)
 - stdair::LegDate, [509](#)
 - stdair::OnDDate, [531](#)
 - stdair::PosChannel, [560](#)
 - stdair::SegmentCabin, [592](#)
 - stdair::SegmentDate, [604](#)
 - stdair::SegmentPeriod, [614](#)
 - stdair::TimePeriod, [656](#)
 - stdair::YieldFeatures, [673](#)
 - stdair::YieldStore, [682](#)
- FacBomManager
 - stdair::AirlineClassList, [241](#)
 - stdair::AirportPair, [256](#)
 - stdair::BomHolder, [283](#)
 - stdair::BomManager, [293](#)
 - stdair::BomRoot, [306](#)
 - stdair::BookingClass, [320](#)
 - stdair::Bucket, [336](#)
 - stdair::DatePeriod, [356](#)
 - stdair::EventQueue, [385](#)
 - stdair::FacBomManager, [402](#)
 - stdair::FareFamily, [417](#)
 - stdair::FareFeatures, [425](#)
 - stdair::FlightDate, [440](#)
 - stdair::FlightPeriod, [447](#)
 - stdair::GuillotineBlock, [465](#)
 - stdair::Inventory, [475](#)
 - stdair::LegCabin, [496](#)
 - stdair::LegDate, [509](#)
 - stdair::OnDDate, [531](#)
 - stdair::PosChannel, [560](#)
 - stdair::SegmentCabin, [592](#)

- stdair::SegmentDate, [604](#)
- stdair::SegmentPeriod, [614](#)
- stdair::TimePeriod, [656](#)
- stdair::YieldFeatures, [674](#)
- stdair::YieldStore, [682](#)
- FacServiceAbstract
 - stdair::FacServiceAbstract, [407](#)
- FacSTDAIRServiceContext
 - stdair::FacSTDAIRServiceContext, [409](#)
 - stdair::STDAIR_ServiceContext, [650](#)
- FacSupervisor
 - stdair::DBSessionManager, [362](#)
 - stdair::FacSupervisor, [411](#), [412](#)
 - stdair::Logger, [514](#)
- FamilyCode_T
 - stdair, [191](#)
- Fare_T
 - stdair, [194](#)
- FareFamily
 - stdair::FareFamily, [415](#)
- FareFamilyKey
 - stdair::FareFamilyKey, [418](#)
- FareFamilyKey::serialize< ba::text_iarchive >
 - stdair, [221](#)
- FareFamilyKey::serialize< ba::text_oarchive >
 - stdair, [221](#)
- FareFamilyList_T
 - stdair, [182](#)
- FareFamilyMap_T
 - stdair, [182](#)
- FareFeatures
 - stdair::FareFeatures, [422](#)
- FareFeaturesDetailedList_T
 - stdair, [183](#)
- FareFeaturesKey
 - stdair::FareFeaturesKey, [426](#)
- FareFeaturesList_T
 - stdair, [183](#)
- FareFeaturesMap_T
 - stdair, [183](#)
- FareFeaturesWithKey_T
 - stdair, [183](#)
- FareOptionList_T
 - stdair, [183](#)
- FareOptionStruct
 - stdair::FareOptionStruct, [429](#), [430](#)
- FILE_PARSING
 - stdair::ServiceInitialisationType, [622](#)
- FileAddress_T
 - stdair, [194](#)
- fileman_completion
 - readline_autocomp.hpp, [1245](#)
- Filename_T
 - stdair, [194](#)
- FileNotFoundException
 - stdair::FileNotFoundException, [433](#)
- find_command
 - readline_autocomp.hpp, [1245](#)
- FIRST
 - stdair::PassengerType, [551](#)
- Flag_T
 - stdair, [194](#)
- FlagSaver
 - stdair::FlagSaver, [434](#)
- FlightDate
 - stdair::FlightDate, [436](#)
- FlightDate::serialize< ba::text_iarchive >
 - stdair, [222](#)
- FlightDate::serialize< ba::text_oarchive >
 - stdair, [222](#)
- FlightDateKey
 - stdair::FlightDateKey, [442](#)
- FlightDateKey::serialize< ba::text_iarchive >
 - stdair, [221](#)
- FlightDateKey::serialize< ba::text_oarchive >
 - stdair, [221](#)
- FlightDateList_T
 - stdair, [183](#)
- FlightDateMap_T
 - stdair, [183](#)
- FlightNumber_T
 - stdair, [191](#)
- FlightPathCode_T
 - stdair, [200](#)
- FlightPeriod
 - stdair::FlightPeriod, [445](#)
- FlightPeriodKey
 - stdair::FlightPeriodKey, [448](#)
- FlightPeriodList_T
 - stdair, [183](#)
- FlightPeriodMap_T
 - stdair, [183](#)
- FloatDuration_T
 - stdair, [195](#)
- FloatingPoint, [449](#)
 - AlmostEquals, [452](#)

- Bits, [450](#)
- bits, [451](#)
- exponent_bits, [451](#)
- FloatingPoint, [451](#)
- fraction_bits, [451](#)
- Infinity, [451](#)
- is_nan, [452](#)
- kBitCount, [452](#)
- kExponentBitCount, [452](#)
- kExponentBitMask, [453](#)
- kFractionBitCount, [452](#)
- kFractionBitMask, [452](#)
- kMaxUlp, [453](#)
- kSignBitMask, [452](#)
- ReinterpretBits, [451](#)
- sign_bit, [451](#)
- ForecasterMode_T
 - stdair, [207](#)
- ForecastingMethod
 - stdair::ForecastingMethod, [454](#)
- FQT
 - stdair::SampleType, [582](#)
- fraction_bits
 - FloatingPoint, [451](#)
- FREQUENT_FLYER_MEMBER
 - stdair, [211](#), [229](#)
- FrequentFlyer_T
 - stdair, [198](#)
- from_base
 - soci::type_conversion< stdair::AirlineStruct
>, [664](#)
- fromStream
 - stdair::AirlineClassList, [240](#)
 - stdair::AirlineClassListKey, [244](#)
 - stdair::AirlineFeature, [247](#)
 - stdair::AirlineFeatureKey, [250](#)
 - stdair::AirlineStruct, [253](#)
 - stdair::AirportPair, [255](#)
 - stdair::AirportPairKey, [259](#)
 - stdair::BasDBParams, [264](#)
 - stdair::BasLogParams, [268](#)
 - stdair::BomAbstract, [270](#)
 - stdair::BomHolder, [282](#)
 - stdair::BomHolderKey, [285](#)
 - stdair::BomRoot, [305](#)
 - stdair::BomRootKey, [308](#)
 - stdair::BookingClass, [318](#)
 - stdair::BookingClassKey, [325](#)
 - stdair::BookingRequestStruct, [330](#)
 - stdair::Bucket, [335](#)
 - stdair::BucketKey, [339](#)
 - stdair::CancellationStruct, [343](#)
 - stdair::DatePeriod, [354](#)
 - stdair::DatePeriodKey, [358](#)
 - stdair::DbAbstract, [359](#)
 - stdair::DemandGenerationMethod, [367](#)
 - stdair::DoWStruct, [373](#)
 - stdair::EventQueue, [382](#)
 - stdair::EventQueueKey, [389](#)
 - stdair::EventStruct, [393](#)
 - stdair::EventType, [398](#)
 - stdair::FareFamily, [416](#)
 - stdair::FareFamilyKey, [419](#)
 - stdair::FareFeatures, [422](#)
 - stdair::FareFeaturesKey, [428](#)
 - stdair::FareOptionStruct, [432](#)
 - stdair::FlightDate, [439](#)
 - stdair::FlightDateKey, [442](#)
 - stdair::FlightPeriod, [446](#)
 - stdair::FlightPeriodKey, [449](#)
 - stdair::ForecastingMethod, [456](#)
 - stdair::GuillotineBlock, [464](#)
 - stdair::GuillotineBlockKey, [469](#)
 - stdair::Inventory, [474](#)
 - stdair::InventoryKey, [478](#)
 - stdair::KeyAbstract, [481](#)
 - stdair::LegCabin, [494](#)
 - stdair::LegCabinKey, [501](#)
 - stdair::LegDate, [508](#)
 - stdair::LegDateKey, [512](#)
 - stdair::OnDDate, [530](#)
 - stdair::OnDDateKey, [534](#)
 - stdair::OptimisationNotificationStruct,
[539](#)
 - stdair::ParsedKey, [541](#)
 - stdair::PartnershipTechnique, [549](#)
 - stdair::PassengerType, [553](#)
 - stdair::PeriodStruct, [556](#)
 - stdair::PosChannel, [558](#)
 - stdair::PosChannelKey, [562](#)
 - stdair::ProgressStatus, [567](#)
 - stdair::ProgressStatusSet, [569](#)
 - stdair::RandomGeneration, [574](#)
 - stdair::RMEventStruct, [576](#)
 - stdair::SampleType, [584](#)
 - stdair::SegmentCabin, [591](#)
 - stdair::SegmentCabinKey, [596](#)
 - stdair::SegmentDate, [603](#)
 - stdair::SegmentDateKey, [608](#)
 - stdair::SegmentPeriod, [614](#)

- stdair::SegmentPeriodKey, 617
- stdair::ServiceAbstract, 620
- stdair::ServiceInitialisationType, 624
- stdair::SnapshotStruct, 629
- stdair::STDAIR_ServiceContext, 650
- stdair::StructAbstract, 652
- stdair::TimePeriod, 655
- stdair::TimePeriodKey, 659
- stdair::TravelSolutionStruct, 663
- stdair::VirtualClassStruct, 669
- stdair::YieldFeatures, 672
- stdair::YieldFeaturesKey, 676
- stdair::YieldRange, 679
- stdair::YieldStore, 681
- stdair::YieldStoreKey, 684
- func
 - COMMAND, 348
- GeneratedDemandVector_T
 - stdair, 207
- GeneratedDemandVectorHolder_T
 - stdair, 207
- generateDemandSamples
 - stdair::BookingClass, 319, 320
- generateExponential
 - stdair::RandomGeneration, 573
- generateNormal
 - stdair::RandomGeneration, 572
- generateUniform
 - stdair::RandomGeneration, 572
- generateUniform01
 - stdair::RandomGeneration, 572
- getActualNb
 - stdair::ProgressStatus, 565
- getActualTotalNbOfEvents
 - stdair::EventQueue, 379, 380
- getActualTotalNumberOfEventsToBeGenerated
 - stdair::STDAIR_Service, 643
- getAdvancePurchase
 - stdair::FareFeatures, 423
 - stdair::FareFeaturesKey, 427
- getAirlineCode
 - stdair::AirlineFeatureKey, 249
 - stdair::AirlineStruct, 252
 - stdair::FlightDate, 437
 - stdair::Inventory, 473
 - stdair::InventoryKey, 478
 - stdair::LegDate, 504
 - stdair::OnDDate, 528
 - stdair::RMEventStruct, 576
 - stdair::SnapshotStruct, 628
 - stdair::YieldStore, 682
 - stdair::YieldStoreKey, 684
- getAirlineCodeList
 - stdair::AirlineClassList, 239
 - stdair::AirlineClassListKey, 243
- getAirlineName
 - stdair::AirlineStruct, 252
- getAuthorizationLevel
 - stdair::BookingClass, 314
 - stdair::LegCabin, 489
- getAvailability
 - stdair::Bucket, 334
 - stdair::FareOptionStruct, 430
 - stdair::LegCabin, 488
- getAvailabilityPool
 - stdair::LegCabin, 488
 - stdair::SegmentCabin, 589
- getAverageYield
 - stdair::YieldRange, 678
- getAvgCancellationPercentage
 - stdair::LegCabin, 490
- getBaseGenerator
 - stdair::RandomGeneration, 573
- getBidPriceVector
 - stdair::LegCabin, 489, 490
 - stdair::SegmentCabin, 589
- getBidPriceVectorHolder
 - stdair::TravelSolutionStruct, 661
- getBlockIndex
 - stdair::GuillotineBlock, 460
- getBlockNumber
 - stdair::GuillotineBlock, 460
- getBlockSpace
 - stdair::SegmentCabin, 588
- getBoardingDate
 - stdair::LegDate, 506
 - stdair::SegmentDate, 600
- getBoardingDateOffset
 - stdair::SegmentPeriod, 612
- getBoardingPoint
 - stdair::AirportPair, 256
 - stdair::AirportPairKey, 258
 - stdair::LegDate, 504
 - stdair::LegDateKey, 512
 - stdair::SegmentDate, 599
 - stdair::SegmentDateKey, 607
 - stdair::SegmentPeriod, 611
 - stdair::SegmentPeriodKey, 617
- getBoardingTime

- stdair::LegDate, [506](#)
- stdair::ParsedKey, [541](#)
- stdair::SegmentDate, [600](#)
- stdair::SegmentPeriod, [611](#)
- getBomHolderPtr
 - stdair::FacBomManager, [402](#)
- getBomRoot
 - stdair::STDAIR_Service, [648](#)
- getBookingChannel
 - stdair::BookingRequestStruct, [329](#)
- getBookingCounter
 - stdair::SegmentCabin, [588](#)
- getBookingRequest
 - stdair::EventStruct, [392](#)
- getCabinBookingClassMap
 - stdair::SegmentPeriod, [612](#)
- getCabinClassPairList
 - stdair::OnDDate, [529](#)
- getCabinCode
 - stdair::LegCabin, [487](#)
 - stdair::LegCabinKey, [501](#)
 - stdair::SegmentCabin, [587](#)
 - stdair::SegmentCabinKey, [596](#)
 - stdair::YieldFeatures, [673](#)
 - stdair::YieldFeaturesKey, [675](#)
- getCancellation
 - stdair::EventStruct, [393](#)
- getCancellationDateTime
 - stdair::CancellationStruct, [342](#)
- getCancellationPercentage
 - stdair::BookingClass, [315](#)
- getCapacity
 - stdair::LegDate, [507](#)
 - stdair::SegmentCabin, [588](#)
- getChangeFees
 - stdair::FareFeatures, [424](#)
 - stdair::FareFeaturesKey, [427](#)
 - stdair::FareOptionStruct, [430](#)
- getChannel
 - stdair::PosChannel, [559](#)
 - stdair::PosChannelKey, [562](#)
- getChosenFareOption
 - stdair::TravelSolutionStruct, [662](#)
- getClassAvailabilityMapHolder
 - stdair::TravelSolutionStruct, [661](#)
- getClassBpvMapHolder
 - stdair::TravelSolutionStruct, [661](#)
- getClassCode
 - stdair::BookingClass, [313](#)
 - stdair::BookingClassKey, [325](#)
- getClassCodeList
 - stdair::AirlineClassList, [239](#)
 - stdair::AirlineClassListKey, [244](#)
- getClassList
 - stdair::CancellationStruct, [342](#)
- getClassPath
 - stdair::FareOptionStruct, [430](#)
- getClassYieldMapHolder
 - stdair::TravelSolutionStruct, [661](#)
- getCommittedSpace
 - stdair::LegCabin, [488](#)
 - stdair::SegmentCabin, [589](#)
- getConstSegmentCabinDTDAvailabilitySnapshotView
 - stdair::GuillotineBlock, [463](#)
- getConstSegmentCabinDTDBookingSnapshotView
 - stdair::GuillotineBlock, [461](#)
- getConstSegmentCabinDTDCancellationSnapshotView
 - stdair::GuillotineBlock, [461](#)
- getConstSegmentCabinDTDProductAndPriceOrientedBookingSnapshotView
 - stdair::GuillotineBlock, [462](#)
- getConstSegmentCabinDTDRangeAvailabilitySnapshotView
 - stdair::GuillotineBlock, [463](#)
- getConstSegmentCabinDTDRangeBookingSnapshotView
 - stdair::GuillotineBlock, [461](#)
- getConstSegmentCabinDTDRangeCancellationSnapshotView
 - stdair::GuillotineBlock, [462](#)
- getConstSegmentCabinDTDRangeProductAndPriceOrientedBookingSnapshotView
 - stdair::GuillotineBlock, [462](#)
- getCumulatedBookingLimit
 - stdair::BookingClass, [314](#)
 - stdair::VirtualClassStruct, [668](#)
- getCumulatedProtection
 - stdair::BookingClass, [314](#)
 - stdair::VirtualClassStruct, [668](#)
- getCurrentBidPrice
 - stdair::LegCabin, [488](#)
 - stdair::SegmentCabin, [589](#)
- getCurrentNb
 - stdair::ProgressStatus, [565](#)
- getCurrentNbOfEvents
 - stdair::EventQueue, [379](#), [380](#)
- getDate
 - stdair::OnDDate, [528](#)
 - stdair::OnDDateKey, [533](#)
- getDateOffset
 - stdair::LegDate, [507](#)
 - stdair::SegmentDate, [601](#)
- getDatePeriod
 - stdair::DatePeriod, [355](#)
 - stdair::DatePeriodKey, [358](#)

- getDateRange
 - stdair::PeriodStruct, 554
- getDayOfWeek
 - stdair::DoWStruct, 372
- getDBName
 - stdair::BasDBParams, 262
- getDBParams
 - stdair::STDAIR_Service, 648
- getDBSession
 - stdair::DBSessionManager, 362
- getDemandGeneratorKey
 - stdair::BookingRequestStruct, 328
- getDemandInfoMap
 - stdair::OnDDate, 529
- getDepartureDate
 - stdair::FlightDate, 437
 - stdair::FlightDateKey, 442
- getDerivativeValue
 - stdair::ContinuousAttributeLite, 350
- getDestination
 - stdair::BookingRequestStruct, 328
 - stdair::OnDDate, 528
 - stdair::OnDDateKey, 534
 - stdair::OptimisationNotificationStruct, 537
- getDistance
 - stdair::LegDate, 507
 - stdair::SegmentDate, 601
- getDoW
 - stdair::PeriodStruct, 555
- getElapsedTime
 - stdair::LegDate, 506
 - stdair::SegmentDate, 600
 - stdair::SegmentPeriod, 612
- getETB
 - stdair::BookingClass, 316
 - stdair::LegCabin, 490
- getEventQueue
 - stdair::STDAIR_Service, 648
- getEventQueueID
 - stdair::EventQueueKey, 389
- getEventType
 - stdair::EventStruct, 392
- getExpectedNb
 - stdair::ProgressStatus, 565
- getExpectedTotalNbOfEvents
 - stdair::EventQueue, 379, 380
- getExpectedTotalNumberOfEventsToBeGenerated
 - stdair::STDAIR_Service, 642, 643
- getFamilyCode
 - stdair::FareFamily, 415
 - stdair::FareFamilyKey, 419
- getFare
 - stdair::AirlineClassList, 240
 - stdair::FareOptionStruct, 430
- getFareFamilyStatus
 - stdair::SegmentCabin, 589
- getFareOptionList
 - stdair::TravelSolutionStruct, 661
- getFareOptionListRef
 - stdair::TravelSolutionStruct, 661
- getFlightDate
 - stdair::Inventory, 473, 474
- getFlightDateDescription
 - stdair::RMEEventStruct, 576
- getFlightDateKey
 - stdair::ParsedKey, 540
- getFlightNumber
 - stdair::FlightDate, 436
 - stdair::FlightDateKey, 442
 - stdair::FlightPeriod, 445
 - stdair::FlightPeriodKey, 448
- getForcedInitialisationFlag
 - stdair::BasLogParams, 266
- getFrequentFlyerType
 - stdair::BookingRequestStruct, 329
 - stdair::OptimisationNotificationStruct, 538
- getFullerKey
 - stdair::LegCabin, 487
 - stdair::SegmentCabin, 587
- getGeneratedDemandVector
 - stdair::BookingClass, 317
 - stdair::VirtualClassStruct, 668
- getGrossAvailability
 - stdair::LegCabin, 489
- getGroupNbOfSeats
 - stdair::LegCabin, 490
- getGuillotineBlock
 - stdair::SegmentCabin, 588
- getGuillotineNumber
 - stdair::GuillotineBlock, 460
 - stdair::GuillotineBlockKey, 468
- GetHistory
 - swift::SReadline, 635
- getHolderMap
 - stdair::AirlineClassList, 239
 - stdair::AirportPair, 256
 - stdair::BomRoot, 304
 - stdair::BookingClass, 313

- stdair::Bucket, [334](#)
- stdair::DatePeriod, [355](#)
- stdair::EventQueue, [379](#)
- stdair::FareFamily, [416](#)
- stdair::FareFeatures, [423](#)
- stdair::FlightDate, [437](#)
- stdair::FlightPeriod, [446](#)
- stdair::GuillotineBlock, [460](#)
- stdair::Inventory, [473](#)
- stdair::LegCabin, [487](#)
- stdair::LegDate, [505](#)
- stdair::OnDDate, [528](#)
- stdair::PosChannel, [559](#)
- stdair::SegmentCabin, [587](#)
- stdair::SegmentDate, [600](#)
- stdair::SegmentPeriod, [612](#)
- stdair::TimePeriod, [656](#)
- stdair::YieldFeatures, [673](#)
- getHost
 - stdair::BasDBParams, [262](#)
- getID
 - stdair::BomRootKey, [308](#)
- getInventory
 - stdair::BomRoot, [304](#)
- getInventoryKey
 - stdair::ParsedKey, [540](#)
- getKey
 - stdair::AirlineClassList, [239](#)
 - stdair::AirlineFeature, [247](#)
 - stdair::AirportPair, [256](#)
 - stdair::BomRoot, [304](#)
 - stdair::BookingClass, [313](#)
 - stdair::Bucket, [334](#)
 - stdair::DatePeriod, [355](#)
 - stdair::EventQueue, [378](#)
 - stdair::FareFamily, [415](#)
 - stdair::FareFeatures, [423](#)
 - stdair::FlightDate, [436](#)
 - stdair::FlightPeriod, [445](#)
 - stdair::GuillotineBlock, [459](#)
 - stdair::Inventory, [473](#)
 - stdair::LegCabin, [487](#)
 - stdair::LegDate, [504](#)
 - stdair::OnDDate, [528](#)
 - stdair::PosChannel, [559](#)
 - stdair::SegmentCabin, [587](#)
 - stdair::SegmentDate, [599](#)
 - stdair::SegmentPeriod, [611](#)
 - stdair::TimePeriod, [655](#)
 - stdair::YieldFeatures, [673](#)
 - stdair::YieldStore, [682](#)
- getLabel
 - stdair::DemandGenerationMethod, [366](#)
 - stdair::EventType, [396](#)
 - stdair::ForecastingMethod, [455](#)
 - stdair::PartnershipTechnique, [548](#)
 - stdair::PassengerType, [551](#)
 - stdair::SampleType, [582](#)
 - stdair::ServiceInitialisationType, [622](#)
- getLegCabin
 - stdair::LegDate, [505](#)
- getLegDate
 - stdair::FlightDate, [437](#), [438](#)
- GetLine
 - swift::SReadline, [634](#), [635](#)
- getList
 - stdair::BomManager, [291](#), [292](#)
- getLogLevel
 - stdair::BasLogParams, [266](#)
- getLogParams
 - stdair::STDAIR_Service, [648](#)
- getLogStream
 - stdair::BasLogParams, [266](#)
- getLowerYield
 - stdair::YieldRange, [678](#)
- getMap
 - stdair::BomManager, [291](#)
- getMarketingSegmentDateList
 - stdair::SegmentDate, [601](#)
- getMean
 - stdair::BookingClass, [317](#)
 - stdair::VirtualClassStruct, [668](#)
- getMethod
 - stdair::DemandGenerationMethod, [366](#), [367](#)
 - stdair::ForecastingMethod, [455](#)
- getMethodAsChar
 - stdair::DemandGenerationMethod, [367](#)
- getMethodAsString
 - stdair::DemandGenerationMethod, [367](#)
 - stdair::ForecastingMethod, [455](#)
- getMethodLabel
 - stdair::DemandGenerationMethod, [366](#)
 - stdair::ForecastingMethod, [455](#)
- getMethodLabelAsString
 - stdair::DemandGenerationMethod, [366](#)
 - stdair::ForecastingMethod, [455](#)
- getMIN
 - stdair::SegmentCabin, [588](#)
- getMinimumStay

- stdair::FareFeatures, [424](#)
- stdair::FareFeaturesKey, [427](#)
- getNbOfBookings
 - stdair::BookingClass, [315](#)
- getNbOfCancellations
 - stdair::BookingClass, [316](#)
- getNbOfGroupBookings
 - stdair::BookingClass, [315](#)
- getNbOfPendingGroupBookings
 - stdair::BookingClass, [315](#)
- getNbOfSegments
 - stdair::OnDDate, [529](#)
 - stdair::OnDDateKey, [534](#)
- getNbOfStaffBookings
 - stdair::BookingClass, [315](#)
- getNbOfWLBookings
 - stdair::BookingClass, [315](#)
- getNegotiatedSpace
 - stdair::BookingClass, [314](#)
- getNetAvailability
 - stdair::LegCabin, [489](#)
- getNetClassAvailability
 - stdair::BookingClass, [316](#)
- getNetRevenueAvailability
 - stdair::BookingClass, [316](#)
- getNonRefundable
 - stdair::FareOptionStruct, [430](#)
- getNoShowPercentage
 - stdair::BookingClass, [315](#)
- getNotificationDateTime
 - stdair::OptimisationNotificationStruct, [537](#)
- getObject
 - stdair::BomManager, [292](#)
- getObjectPtr
 - stdair::BomManager, [292](#)
- getOffDate
 - stdair::LegDate, [506](#)
 - stdair::SegmentDate, [600](#)
- getOffDateOffset
 - stdair::SegmentPeriod, [612](#)
- getOfferedCapacity
 - stdair::LegCabin, [487](#)
- getOffPoint
 - stdair::AirportPair, [256](#)
 - stdair::AirportPairKey, [258](#)
 - stdair::LegDate, [506](#)
 - stdair::SegmentDate, [600](#)
 - stdair::SegmentDateKey, [607](#)
 - stdair::SegmentPeriod, [611](#)
 - stdair::SegmentPeriodKey, [617](#)
- getOffTime
 - stdair::LegDate, [506](#)
 - stdair::SegmentDate, [600](#)
 - stdair::SegmentPeriod, [611](#)
- getOperatingSegmentDate
 - stdair::SegmentDate, [601](#)
- getOptimisationChannel
 - stdair::OptimisationNotificationStruct, [538](#)
- getOptimisationNotificationStruct
 - stdair::EventStruct, [393](#)
- getOrigin
 - stdair::BookingRequestStruct, [328](#)
 - stdair::OnDDate, [528](#)
 - stdair::OnDDateKey, [533](#)
 - stdair::OptimisationNotificationStruct, [537](#)
- getOverallStatus
 - stdair::ProgressStatusSet, [569](#)
- getParent
 - stdair::AirlineClassList, [239](#)
 - stdair::AirportPair, [256](#)
 - stdair::BomManager, [292](#)
 - stdair::BookingClass, [313](#)
 - stdair::Bucket, [334](#)
 - stdair::DatePeriod, [355](#)
 - stdair::EventQueue, [379](#)
 - stdair::FareFamily, [415](#)
 - stdair::FareFeatures, [423](#)
 - stdair::FlightDate, [436](#)
 - stdair::FlightPeriod, [445](#)
 - stdair::GuillotineBlock, [459](#)
 - stdair::Inventory, [473](#)
 - stdair::LegCabin, [487](#)
 - stdair::LegDate, [504](#)
 - stdair::OnDDate, [528](#)
 - stdair::PosChannel, [559](#)
 - stdair::SegmentCabin, [587](#)
 - stdair::SegmentDate, [599](#)
 - stdair::SegmentPeriod, [611](#)
 - stdair::TimePeriod, [655](#)
 - stdair::YieldFeatures, [673](#)
 - stdair::YieldStore, [681](#)
- getParentPtr
 - stdair::BomManager, [292](#)
- getPartySize
 - stdair::BookingRequestStruct, [329](#)
 - stdair::CancellationStruct, [342](#)

- stdair::OptimisationNotificationStruct, 537
- getPassword
 - stdair::BasDBParams, 262
- getPeriod
 - stdair::FlightPeriod, 445
 - stdair::FlightPeriodKey, 448
- getPhysicalCapacity
 - stdair::LegCabin, 488
- getPort
 - stdair::BasDBParams, 262
- getPOS
 - stdair::BookingRequestStruct, 328
 - stdair::OptimisationNotificationStruct, 537
- getPos
 - stdair::PosChannel, 559
 - stdair::PosChannelKey, 561
- getPreferredDepartureDate
 - stdair::BookingRequestStruct, 328
 - stdair::OptimisationNotificationStruct, 537
- getPreferredCabin
 - stdair::BookingRequestStruct, 329
 - stdair::OptimisationNotificationStruct, 537
- getPreferredDepartureTime
 - stdair::BookingRequestStruct, 328
 - stdair::OptimisationNotificationStruct, 538
- getPreviousBidPrice
 - stdair::LegCabin, 488
- getProtection
 - stdair::BookingClass, 314
- getQueueSize
 - stdair::EventQueue, 385
- getRefundableOption
 - stdair::FareFeatures, 424
 - stdair::FareFeaturesKey, 427
- getRegradeAdjustment
 - stdair::LegCabin, 489
- getRemainingProportion
 - stdair::ContinuousAttributeLite, 350
- getRequestDateTime
 - stdair::BookingRequestStruct, 329
- getRMEvent
 - stdair::EventStruct, 393
- getRMEventTime
 - stdair::RMEventStruct, 576
- getSaturdayStay
 - stdair::FareFeatures, 423
 - stdair::FareFeaturesKey, 427
 - stdair::FareOptionStruct, 430
- getSeatIndex
 - stdair::Bucket, 334
 - stdair::BucketKey, 339
- getSegmentAvailability
 - stdair::BookingClass, 316
- getSegmentCabinDTDAvailabilitySnapshotView
 - stdair::GuillotineBlock, 463
- getSegmentCabinDTDBookingSnapshotView
 - stdair::GuillotineBlock, 461
- getSegmentCabinDTDCancellationSnapshotView
 - stdair::GuillotineBlock, 462
- getSegmentCabinDTDProductAndPriceOrientedBookingSnapshotView
 - stdair::GuillotineBlock, 462
- getSegmentCabinDTDRangeAvailabilitySnapshotView
 - stdair::GuillotineBlock, 463
- getSegmentCabinDTDRangeBookingSnapshotView
 - stdair::GuillotineBlock, 461
- getSegmentCabinDTDRangeCancellationSnapshotView
 - stdair::GuillotineBlock, 462
- getSegmentCabinDTDRangeProductAndPriceOrientedBookingSnapshotView
 - stdair::GuillotineBlock, 463
- getSegmentCabinIndexMap
 - stdair::GuillotineBlock, 460
- getSegmentDate
 - stdair::FlightDate, 438
- getSegmentKey
 - stdair::ParsedKey, 541
- getSegmentPath
 - stdair::CancellationStruct, 342
 - stdair::TravelSolutionStruct, 661
- getServiceInitialisationType
 - stdair::STDAIR_Service, 648
- getSnapshotStruct
 - stdair::EventStruct, 393
- getSnapshotTime
 - stdair::SnapshotStruct, 628
- getSoldSeat
 - stdair::LegCabin, 488
- getSoldSeats
 - stdair::Bucket, 334
- getSpecificGeneratorStatus
 - stdair::ProgressStatusSet, 569
- getStaffNbOfSeats
 - stdair::LegCabin, 490
- getStandardDayOfWeek
 - stdair::DoWStruct, 372
- getStart

- stdair::BasChronometer, 260
- getStatus
 - stdair::EventQueue, 379
- getStayDuration
 - stdair::BookingRequestStruct, 329
 - stdair::OptimisationNotificationStruct, 538
- getStdDev
 - stdair::BookingClass, 317
 - stdair::VirtualClassStruct, 668
- getSubclassCode
 - stdair::BookingClass, 314
- getTechnique
 - stdair::PartnershipTechnique, 548
- getTechniqueAsChar
 - stdair::PartnershipTechnique, 548
- getTechniqueAsString
 - stdair::PartnershipTechnique, 549
- getTechniqueLabel
 - stdair::PartnershipTechnique, 548
- getTechniqueLabelAsString
 - stdair::PartnershipTechnique, 548
- getTimeOffset
 - stdair::LegDate, 507
 - stdair::SegmentDate, 601
- getTimeRangeEnd
 - stdair::TimePeriod, 656
 - stdair::TimePeriodKey, 658
- getTimeRangeStart
 - stdair::TimePeriod, 656
 - stdair::TimePeriodKey, 658
- getTotalForecast
 - stdair::OnDDate, 529
- getTotalForecastMap
 - stdair::OnDDate, 529
- getTripType
 - stdair::BookingRequestStruct, 329
 - stdair::FareFeatures, 423
 - stdair::FareFeaturesKey, 427
 - stdair::OptimisationNotificationStruct, 538
 - stdair::YieldFeatures, 673
 - stdair::YieldFeaturesKey, 675
- getType
 - stdair::EventType, 397
 - stdair::PassengerType, 552
 - stdair::SampleType, 583
 - stdair::ServiceInitialisationType, 623
- getTypeAsChar
 - stdair::ServiceInitialisationType, 623
- getTypeAsString
 - stdair::EventType, 397
 - stdair::PassengerType, 552
 - stdair::SampleType, 583
 - stdair::ServiceInitialisationType, 623
- getTypeLabel
 - stdair::EventType, 396
 - stdair::PassengerType, 551
 - stdair::SampleType, 583
 - stdair::ServiceInitialisationType, 623
- getTypeLabelAsString
 - stdair::EventType, 397
 - stdair::PassengerType, 551
 - stdair::SampleType, 583
 - stdair::ServiceInitialisationType, 623
- getTypeSpecificStatus
 - stdair::ProgressStatusSet, 568
- getUpperBound
 - stdair::ContinuousAttributeLite, 350
- getUpperYield
 - stdair::YieldRange, 678
- getUPR
 - stdair::LegCabin, 489
 - stdair::SegmentCabin, 588
- getUser
 - stdair::BasDBParams, 262
- getValue
 - stdair::ContinuousAttributeLite, 350
- getValueOfTime
 - stdair::BookingRequestStruct, 330
 - stdair::OptimisationNotificationStruct, 538
- getValueTypeIndexMap
 - stdair::GuillotineBlock, 460
- getVirtualClassList
 - stdair::LegCabin, 490
- getwd
 - readline_autocomp.hpp, 1243
- getWLNbOfSeats
 - stdair::LegCabin, 490
- getWTP
 - stdair::BookingRequestStruct, 329
 - stdair::OptimisationNotificationStruct, 538
- getYield
 - stdair::AirlineClassList, 240
 - stdair::BookingClass, 316
 - stdair::VirtualClassStruct, 668
- getYieldLevelDemandMap
 - stdair::LegCabin, 491

- getYieldRangeUpperValue
 - stdair::Bucket, [334](#)
- GuillotineBlock
 - stdair::GuillotineBlock, [459](#)
- GuillotineBlockKey
 - stdair::GuillotineBlockKey, [468](#)
- GuillotineBlockKey::serialize< ba::text_iarchive >
 - stdair, [221](#)
- GuillotineBlockKey::serialize< ba::text_oarchive >
 - stdair, [221](#)
- GuillotineBlockList_T
 - stdair, [184](#)
- GuillotineBlockMap_T
 - stdair, [184](#)
- GuillotineNumber_T
 - stdair, [191](#)
- hasList
 - stdair::BomManager, [291](#), [292](#)
- hasMap
 - stdair::BomManager, [291](#), [293](#)
- HistoricalDataLimit_T
 - stdair, [207](#)
- HolderMap_T
 - stdair, [181](#)
- hour_p_t
 - stdair, [179](#)
- hour_t
 - stdair, [178](#)
- HTMLDIR
 - stdair-paths.hpp, [1170](#)
- IBP_DA
 - stdair::PartnershipTechnique, [547](#)
- IBP_YP
 - stdair::PartnershipTechnique, [547](#)
- IBP_YP_U
 - stdair::PartnershipTechnique, [547](#)
- Identity_T
 - stdair, [191](#)
- INCLUDEDIR
 - stdair-paths.hpp, [1169](#)
- Infinity
 - FloatingPoint, [451](#)
- INFODIR
 - stdair-paths.hpp, [1170](#)
- init
 - stdair::AirlineFeature, [247](#)
 - stdair::DefaultDCPLList, [363](#)
 - stdair::DefaultDtdFratMap, [363](#)
 - stdair::DefaultDtdProbMap, [364](#)
 - stdair::RandomGeneration, [573](#)
- initialize_readline
 - readline_autocomp.hpp, [1245](#)
- invSnapshotBlocks
 - stdair::GuillotineBlock, [464](#)
- InputFilePath
 - stdair::InputFilePath, [470](#)
- instance
 - stdair::DBSessionManager, [362](#)
 - stdair::FacBom, [400](#)
 - stdair::FacSTDAIRServiceContext, [409](#)
 - stdair::FacSupervisor, [412](#)
 - stdair::Logger, [514](#)
- Int
 - TypeWithSize< 4 >, [666](#)
 - TypeWithSize< 8 >, [666](#)
- int1_p_t
 - stdair, [178](#)
- intDisplay
 - stdair, [220](#)
- IntDuration_T
 - stdair, [195](#)
- intersection
 - stdair::DoWStruct, [373](#)
 - stdair::PeriodStruct, [555](#)
- INV
 - stdair::SampleType, [582](#)
- Inventory
 - stdair::Inventory, [472](#)
- Inventory::serialize< ba::text_iarchive >
 - stdair, [222](#)
- Inventory::serialize< ba::text_oarchive >
 - stdair, [222](#)
- InventoryKey
 - stdair::InventoryKey, [477](#)
- InventoryKey::serialize< ba::text_iarchive >
 - stdair, [221](#)
- InventoryKey::serialize< ba::text_oarchive >
 - stdair, [221](#)
- InventoryList_T
 - stdair, [184](#)
- InventoryMap_T
 - stdair, [184](#)
- is_nan
 - FloatingPoint, [452](#)
- isAdvancePurchaseValid

- stdair::FareFeatures, [424](#)
- isDepartureDateValid
 - stdair::DatePeriod, [356](#)
- isDepartureTimeValid
 - stdair::TimePeriod, [656](#)
- isQueueDone
 - stdair::EventQueue, [383](#)
 - stdair::STDAIR_Service, [644](#)
- isQueueEmpty
 - stdair::EventQueue, [385](#)
- isStayDurationValid
 - stdair::FareFeatures, [424](#)
- isTripTypeValid
 - stdair::FareFeatures, [424](#)
 - stdair::YieldFeatures, [673](#)
- isValid
 - stdair::DoWStruct, [373](#)
 - stdair::PeriodStruct, [555](#)
- iterateOnStatement
 - stdair::DBManagerForAirlines, [361](#)
- iterator_t
 - stdair, [178](#)
- jsonBookingClassExport
 - stdair, [220](#)
- jsonBucketExport
 - stdair, [220](#)
- jsonExport
 - stdair::BomJSONExport, [286](#)
 - stdair::STDAIR_Service, [645](#)
- jsonFareFamilyExport
 - stdair, [220](#)
- jsonImportFlightDateKey
 - stdair::BomJSONImport, [287](#)
- jsonImportInventoryKey
 - stdair::BomJSONImport, [287](#)
- jsonLegCabinExport
 - stdair, [220](#)
- jsonLegDateExport
 - stdair, [219](#)
- jsonSegmentCabinExport
 - stdair, [220](#)
- jsonSegmentDateExport
 - stdair, [220](#)
- kBitCount
 - FloatingPoint, [452](#)
- kExponentBitCount
 - FloatingPoint, [452](#)
- kExponentBitMask
- FloatingPoint, [453](#)
- Key_T
 - stdair::AirlineClassList, [238](#)
 - stdair::AirlineFeature, [246](#)
 - stdair::AirportPair, [254](#)
 - stdair::BomHolder, [281](#)
 - stdair::BomRoot, [303](#)
 - stdair::BookingClass, [313](#)
 - stdair::Bucket, [333](#)
 - stdair::DatePeriod, [354](#)
 - stdair::EventQueue, [378](#)
 - stdair::FareFamily, [415](#)
 - stdair::FareFeatures, [421](#)
 - stdair::FlightDate, [436](#)
 - stdair::FlightPeriod, [445](#)
 - stdair::GuillotineBlock, [459](#)
 - stdair::Inventory, [472](#)
 - stdair::LegCabin, [486](#)
 - stdair::LegDate, [504](#)
 - stdair::OnDDate, [527](#)
 - stdair::PosChannel, [558](#)
 - stdair::SegmentCabin, [586](#)
 - stdair::SegmentDate, [599](#)
 - stdair::SegmentPeriod, [610](#)
 - stdair::TimePeriod, [654](#)
 - stdair::YieldFeatures, [671](#)
 - stdair::YieldStore, [681](#)
- KeyAbstract.hpp
 - operator<<, [997](#)
 - operator>>, [997](#)
- KeyDescription_T
 - stdair, [191](#)
- KeyList_T
 - stdair, [184](#)
- KeyNotFoundException
 - stdair::KeyNotFoundException, [483](#)
- keyToValue
 - stdair::DictionaryManager, [368](#)
- kFractionBitCount
 - FloatingPoint, [452](#)
- kFractionBitMask
 - FloatingPoint, [452](#)
- kMaxUlp
 - FloatingPoint, [453](#)
- kSignBitMask
 - FloatingPoint, [452](#)
- LAST_VALUE
 - stdair::DemandGenerationMethod, [365](#)
 - stdair::EventType, [396](#)

- stdair::ForecastingMethod, [454](#)
- stdair::LOG, [236](#)
- stdair::PartnershipTechnique, [547](#)
- stdair::PassengerType, [551](#)
- stdair::SampleType, [582](#)
- stdair::ServiceInitialisationType, [622](#)
- LegCabin
 - stdair::LegCabin, [486](#)
- LegCabinKey
 - stdair::LegCabinKey, [500](#)
- LegCabinList_T
 - stdair, [184](#)
- LegCabinMap_T
 - stdair, [185](#)
- LegDate
 - stdair::LegDate, [504](#)
- LegDateKey
 - stdair::LegDateKey, [512](#)
- LegDateList_T
 - stdair, [185](#)
- LegDateMap_T
 - stdair, [185](#)
- LEISURE
 - stdair::PassengerType, [551](#)
- LIBDIR
 - stdair-paths.hpp, [1169](#)
- LIBEXECDIR
 - stdair-paths.hpp, [1169](#)
- linkWithOperating
 - stdair::SegmentDate, [602](#)
- linkWithParent
 - stdair::FacBomManager, [405](#)
- list
 - stdair::BomDisplay, [274](#)
 - stdair::STDAIR_Service, [645](#)
- listAirportPairDateRange
 - stdair::BomDisplay, [274](#)
 - stdair::STDAIR_Service, [645](#)
- LoadHistory
 - swift::SReadline, [636](#)
- LocationCode_T
 - stdair, [190](#)
- log
 - stdair::Logger, [514](#)
- Logger
 - stdair::BasLogParams, [268](#)
- Logger.hpp
 - STDAIR_LOG_CORE, [1197](#)
 - STDAIR_LOG_CRITICAL, [1197](#)
 - STDAIR_LOG_DEBUG, [1198](#)
 - STDAIR_LOG_ERROR, [1198](#)
 - STDAIR_LOG_NOTIFICATION, [1198](#)
 - STDAIR_LOG_VERBOSE, [1198](#)
 - STDAIR_LOG_WARNING, [1198](#)
- LongDuration_T
 - stdair, [195](#)
- MANDIR
 - stdair-paths.hpp, [1170](#)
- MapKey_T
 - stdair, [184](#)
- MatchingIndicator_T
 - stdair, [198](#)
- MAXIMAL_AVAILABILITY
 - stdair, [212](#), [232](#)
- MAXIMAL_NUMBER_OF_LEGS_IN_FLIGHT
 - stdair, [218](#), [232](#)
- MAXIMAL_NUMBER_OF_SEGMENTS_IN_OND
 - stdair, [218](#), [232](#)
- MeanStdDevPair_T
 - stdair, [206](#)
- MeanValue_T
 - stdair, [206](#)
- MemoryAllocationException
 - stdair::MemoryAllocationException, [515](#)
- MILLISECONDS_IN_ONE_SECOND
 - stdair, [209](#), [230](#)
- minute_p_t
 - stdair, [179](#)
- minute_t
 - stdair, [178](#)
- MonetaryValue_T
 - stdair, [192](#)
- month_p_t
 - stdair, [179](#)
- month_t
 - stdair, [178](#)
- MUL_PK
 - stdair::ForecastingMethod, [454](#)
- Multiplier_T
 - stdair, [192](#)
- name
 - COMMAND, [348](#)
 - stdair::InputFilePath, [470](#)
 - stdair::RootFilePath, [580](#)
- NbOfAirlines_T
 - stdair, [193](#)
- NbOfBookings_T

- stdair, [193](#)
- NbOfCancellations_T
 - stdair, [193](#)
- NbOfEvents_T
 - stdair, [199](#)
- NbOfFareRules_T
 - stdair, [199](#)
- NbOfFlightDates_T
 - stdair, [200](#)
- NbOfInventoryControlRules_T
 - stdair, [202](#)
- NbOfNoShows_T
 - stdair, [198](#)
- NbOfRequests_T
 - stdair, [193](#)
- NbOfSeats_T
 - stdair, [192](#)
- NbOfSegments_T
 - stdair, [193](#)
- NbOfTravelSolutions_T
 - stdair, [193](#)
- NbOfYields_T
 - stdair, [201](#)
- NetworkID_T
 - stdair, [200](#)
- NO_ADVANCE_PURCHASE
 - stdair, [209](#), [228](#)
- NO_CHANGE_FEES
 - stdair, [210](#), [228](#)
- NO_NON_REFUNDABLE
 - stdair, [228](#)
- No_NON_REFUNDABLE
 - stdair, [210](#)
- NO_SATURDAY_STAY
 - stdair, [210](#), [228](#)
- NO_STAY_DURATION
 - stdair, [210](#), [228](#)
- NON_REFUNDABLE
 - stdair, [210](#), [228](#)
- NONE
 - stdair::PartnershipTechnique, [547](#)
- NonInitialisedContainerException
 - stdair::NonInitialisedContainerException, [516](#)
- NonInitialisedDBSessionManagerException
 - stdair::NonInitialisedDBSessionManagerException, [517](#)
- NonInitialisedLogServiceException
 - stdair::NonInitialisedLogServiceException, [519](#)
- NonInitialisedRelationShipException
 - stdair::NonInitialisedRelationShipException, [520](#)
- NonInitialisedServiceException
 - stdair::NonInitialisedServiceException, [521](#)
- NonRefundable_T
 - stdair, [196](#)
- NonRefundableRatio_T
 - stdair, [196](#)
- NormalDistribution_T
 - stdair, [206](#)
- NormalGenerator_T
 - stdair, [206](#)
- NOT_YET_INITIALISED
 - stdair::ServiceInitialisationType, [622](#)
- NOTIFICATION
 - stdair::LOG, [236](#)
- NULL_BOOST_TIME_DURATION
 - stdair, [208](#), [230](#)
- ObjectCreationgDuplicationException
 - stdair::ObjectCreationgDuplicationException, [522](#)
- ObjectLinkingException
 - stdair::ObjectLinkingException, [524](#)
- ObjectNotFoundException
 - stdair::ObjectNotFoundException, [525](#)
- OnDDate
 - stdair::OnDDate, [527](#)
- OnDDateKey
 - stdair::OnDDateKey, [533](#)
- OnDDateKey::serialize< ba::text_iarchive>
 - stdair, [221](#)
- OnDDateKey::serialize< ba::text_oarchive>
 - stdair, [221](#)
- OnDDateList_T
 - stdair, [185](#)
- OnDDateMap_T
 - stdair, [185](#)
- OnDString_T
 - stdair, [199](#)
- OnDStringList_T
 - stdair, [199](#)
- operator<<
 - BomAbstract.hpp, [808](#)
 - DbAbstract.hpp, [1172](#)
 - KeyAbstract.hpp, [997](#)

- ServiceAbstract.hpp, [1201](#)
- StructAbstract.hpp, [779](#)
- operator>>
 - BomAbstract.hpp, [808](#)
 - DbAAbstract.hpp, [1172](#)
 - KeyAbstract.hpp, [997](#)
 - ServiceAbstract.hpp, [1201](#)
 - StructAbstract.hpp, [779](#)
- operator*
 - stdair, [219](#)
- operator()
 - stdair::RandomGeneration, [572](#)
- operator+
 - stdair, [219](#)
- operator++
 - stdair::ProgressStatus, [566](#)
- operator+=
 - stdair::ProgressStatus, [566](#)
- operator=
 - stdair::ContinuousAttributeLite, [351](#)
 - swift::SKeymap, [627](#)
- operator==
 - stdair::DemandGenerationMethod, [367](#)
 - stdair::EventType, [397](#)
 - stdair::ForecastingMethod, [456](#)
 - stdair::PartnershipTechnique, [549](#)
 - stdair::PassengerType, [552](#)
 - stdair::SampleType, [583](#)
 - stdair::ServiceInitialisationType, [624](#)
- OPT_NOT_4_FD
 - stdair::EventType, [396](#)
- OPT_NOT_4_NET
 - stdair::EventType, [396](#)
- OptimisationNotificationPtr_T
 - stdair, [186](#)
- OptimisationNotificationStruct
 - stdair::OptimisationNotificationStruct, [536](#)
- OptimizerMode_T
 - stdair, [207](#)
- OverbookingRate_T
 - stdair, [203](#)
- PACKAGE
 - stdair-paths.hpp, [1169](#)
- PACKAGE_NAME
 - stdair-paths.hpp, [1169](#)
- PACKAGE_VERSION
 - stdair-paths.hpp, [1169](#)
- ParsedKey
 - stdair::ParsedKey, [540](#)
- ParserException
 - stdair::ParserException, [543](#)
- ParsingFileFailedException
 - stdair::ParsingFileFailedException, [545](#)
- Part of the Business Object Model (BOM)
 - handling, [149](#)
- PartnershipTechnique
 - stdair::PartnershipTechnique, [547](#)
- PartySize_T
 - stdair, [193](#)
- PassengerType
 - stdair::PassengerType, [551](#)
- PassengerType_T
 - stdair, [196](#)
- PDFDIR
 - stdair-paths.hpp, [1170](#)
- Percentage_T
 - stdair, [192](#)
- PeriodStruct
 - stdair::PeriodStruct, [554](#)
- POI_PRO
 - stdair::DemandGenerationMethod, [365](#)
- PolicyDemand_T
 - stdair, [207](#)
- popEvent
 - stdair::EventQueue, [383](#)
 - stdair::STDAIR_Service, [644](#)
- POS_LHR
 - stdair, [209](#), [228](#)
- POS_SIN
 - stdair, [211](#), [229](#)
- PosChannel
 - stdair::PosChannel, [558](#)
- PosChannelDetailedList_T
 - stdair, [186](#)
- PosChannelKey
 - stdair::PosChannelKey, [561](#)
- PosChannelList_T
 - stdair, [186](#)
- PosChannelMap_T
 - stdair, [186](#)
- PosChannelWithKey_T
 - stdair, [186](#)
- PREFIXDIR
 - stdair-paths.hpp, [1169](#)
- prepareSelectStatement
 - stdair::DBManagerForAirlines, [361](#)
- PriceCurrency_T
 - stdair, [192](#)

- PriceValue_T
 - stdair, [192](#)
- Probability_T
 - stdair, [206](#)
- progress
 - stdair::ProgressStatus, [565](#)
- ProgressPercentage_T
 - stdair, [194](#)
- ProgressStatus
 - stdair::ProgressStatus, [564](#)
- ProgressStatusMap_T
 - stdair::EventQueue, [378](#)
- ProgressStatusSet
 - stdair::ProgressStatusSet, [568](#)
- ProportionFactor_T
 - stdair, [199](#)
- ProportionFactorList_T
 - stdair, [199](#)
- ProtectionLevel_T
 - stdair, [203](#)
- ProtectionLevelVector_T
 - stdair, [204](#)
- pt2Func
 - readline_autocomp.hpp, [1243](#)
- ptree
 - bpt, [158](#)
- RAC
 - stdair::SampleType, [582](#)
- RAE_DA
 - stdair::PartnershipTechnique, [547](#)
- RAE_YP
 - stdair::PartnershipTechnique, [547](#)
- RandomGeneration
 - stdair::RandomGeneration, [571](#)
- RandomSeed_T
 - stdair, [205](#)
- readline_autocomp.hpp
 - com_cd, [1244](#)
 - com_delete, [1244](#)
 - com_help, [1244](#)
 - com_list, [1243](#)
 - com_pwd, [1244](#)
 - com_quit, [1244](#)
 - com_rename, [1244](#)
 - com_stat, [1244](#)
 - com_view, [1244](#)
 - command_generator, [1245](#)
 - commands, [1246](#)
 - done, [1246](#)
 - dupstr, [1245](#)
 - execute_line, [1245](#)
 - fileman_completion, [1245](#)
 - find_command, [1245](#)
 - getwd, [1243](#)
 - initialize_readline, [1245](#)
 - pt2Func, [1243](#)
 - stripwhite, [1244](#)
 - syscom, [1246](#)
 - too_dangerous, [1246](#)
 - valid_argument, [1246](#)
 - xmalloc, [1243](#)
- RealNumber_T
 - stdair, [192](#)
- registerBomFactory
 - stdair::FacSupervisor, [412](#)
- RegisterCompletions
 - swift::SReadline, [636](#)
- registerServiceFactory
 - stdair::FacSupervisor, [412](#)
- ReinterpretBits
 - FloatingPoint, [451](#)
- ReplicationNumber_T
 - stdair, [205](#)
- RequestStatus_T
 - stdair, [198](#)
- reset
 - stdair::EventQueue, [382](#)
 - stdair::ProgressStatus, [566](#)
 - stdair::STDAIR_Service, [644](#)
- restore
 - stdair::BomArchive, [272](#)
- retrieveAirline
 - stdair::DBManagerForAirlines, [361](#)
- retrieveAirportPairFromKeySet
 - stdair::BomRetriever, [300](#)
- retrieveBookingClassFromLongKey
 - stdair::BomRetriever, [299](#)
- retrieveDatePeriodListFromKey
 - stdair::BomRetriever, [300](#)
- retrieveDatePeriodListFromKeySet
 - stdair::BomRetriever, [301](#)
- retrieveDummyLegCabin
 - stdair::BomRetriever, [301](#)
- retrieveDummySegmentCabin
 - stdair::BomRetriever, [301](#)
- retrieveFlightDateFromKey
 - stdair::BomRetriever, [297](#)
- retrieveFlightDateFromKeySet
 - stdair::BomRetriever, [296](#)

- retrieveFlightDateFromLongKey
 - stdair::BomRetriever, [295](#), [296](#)
- retrieveInventoryFromKey
 - stdair::BomRetriever, [294](#), [295](#)
- retrieveInventoryFromLongKey
 - stdair::BomRetriever, [294](#)
- retrieveSegmentDateFromKey
 - stdair::BomRetriever, [299](#)
- retrieveSegmentDateFromLongKey
 - stdair::BomRetriever, [297](#), [298](#)
- Revenue_T
 - stdair, [192](#)
- RM
 - stdair::EventType, [396](#)
- RMC
 - stdair::PartnershipTechnique, [547](#)
- RMEventList_T
 - stdair, [187](#)
- RMEventPtr_T
 - stdair, [186](#)
- RMEventStruct
 - stdair::RMEventStruct, [575](#)
- RMS
 - stdair::SampleType, [582](#)
- RootException
 - stdair::RootException, [578](#)
- RootFilePath
 - stdair::RootFilePath, [579](#), [580](#)
- SampleType
 - stdair::SampleType, [582](#)
- SATURDAY_STAY
 - stdair, [210](#), [228](#)
- SaturdayStay_T
 - stdair, [195](#)
- SaturdayStayRatio_T
 - stdair, [196](#)
- SaveHistory
 - swift::SReadline, [635](#)
- SBINDIR
 - stdair-paths.hpp, [1169](#)
- SCH
 - stdair::SampleType, [582](#)
- SeatIndex_T
 - stdair, [203](#)
- second_p_t
 - stdair, [179](#)
- second_t
 - stdair, [178](#)
- SECONDS_IN_ONE_DAY
 - stdair, [209](#), [230](#)
- SegmentCabin
 - stdair::SegmentCabin, [587](#)
- SegmentCabin::serialize< ba::text_iarchive >
 - stdair, [222](#)
- SegmentCabin::serialize< ba::text_oarchive >
 - stdair, [222](#)
- SegmentCabinDTDRangeSnapshotView_-T
 - stdair, [204](#)
- SegmentCabinDTDSnapshotView_T
 - stdair, [204](#)
- SegmentCabinIndexMap_T
 - stdair, [184](#)
- SegmentCabinKey
 - stdair::SegmentCabinKey, [595](#)
- SegmentCabinKey::serialize< ba::text_iarchive >
 - stdair, [222](#)
- SegmentCabinKey::serialize< ba::text_oarchive >
 - stdair, [221](#)
- SegmentCabinList_T
 - stdair, [187](#)
- SegmentCabinMap_T
 - stdair, [187](#)
- SegmentDate
 - stdair::SegmentDate, [599](#)
- SegmentDate::serialize< ba::text_iarchive >
 - stdair, [222](#)
- SegmentDate::serialize< ba::text_oarchive >
 - stdair, [222](#)
- SegmentDateKey
 - stdair::SegmentDateKey, [607](#)
- SegmentDateKey::serialize< ba::text_iarchive >
 - stdair, [222](#)
- SegmentDateKey::serialize< ba::text_oarchive >
 - stdair, [222](#)
- SegmentDateList_T
 - stdair, [187](#)
- SegmentDateMap_T
 - stdair, [187](#)
- SegmentPath_T
 - stdair, [188](#)

- SegmentPathList_T
 - stdair, [188](#)
- SegmentPeriod
 - stdair::SegmentPeriod, [610](#)
- SegmentPeriodDetailedList_T
 - stdair, [187](#)
- SegmentPeriodKey
 - stdair::SegmentPeriodKey, [616](#)
- SegmentPeriodList_T
 - stdair, [187](#)
- SegmentPeriodMap_T
 - stdair, [187](#)
- SegmentPeriodWithKey_T
 - stdair, [187](#)
- sell
 - stdair::BookingClass, [319](#)
- SellupFactorHolder_T
 - stdair, [207](#)
- SellupProbability_T
 - stdair, [207](#)
- SellupProbabilityVector_T
 - stdair, [207](#)
- SerialisationException
 - stdair::SerialisationException, [619](#)
- serialiseHelper
 - stdair, [222](#)
- serialize
 - stdair::AirlineClassList, [241](#)
 - stdair::AirlineClassListKey, [244](#)
 - stdair::BomRoot, [305](#)
 - stdair::BomRootKey, [309](#)
 - stdair::Bucket, [336](#)
 - stdair::BucketKey, [339](#)
 - stdair::FareFamily, [417](#)
 - stdair::FareFamilyKey, [420](#)
 - stdair::FlightDate, [440](#)
 - stdair::FlightDateKey, [443](#)
 - stdair::GuillotineBlock, [465](#)
 - stdair::GuillotineBlockKey, [469](#)
 - stdair::Inventory, [475](#)
 - stdair::InventoryKey, [478](#)
 - stdair::LegCabinKey, [501](#)
 - stdair::OnDDate, [531](#)
 - stdair::OnDDateKey, [535](#)
 - stdair::SegmentCabin, [592](#)
 - stdair::SegmentCabinKey, [597](#)
 - stdair::SegmentDate, [603](#)
 - stdair::SegmentDateKey, [608](#)
- ServiceAbstract
 - stdair::ServiceAbstract, [620](#)
- ServiceAbstract.hpp
 - operator<<, [1201](#)
 - operator>>, [1201](#)
- ServiceFactoryPool_T
 - stdair::FacSupervisor, [411](#)
- ServiceInitialisationType
 - stdair::ServiceInitialisationType, [622](#)
- ServicePool_T
 - stdair::FacServiceAbstract, [407](#)
 - stdair::FacSTDAIRServiceContext, [409](#)
- setActualNb
 - stdair::ProgressStatus, [566](#)
- setActualTotalNbOfEvents
 - stdair::EventQueue, [381](#)
- setAirlineCode
 - stdair::AirlineStruct, [252](#)
- setAirlineFeature
 - stdair::Inventory, [474](#)
- setAirlineName
 - stdair::AirlineStruct, [252](#)
- setAuthorizationLevel
 - stdair::BookingClass, [317](#)
 - stdair::LegCabin, [492](#)
- setAvailability
 - stdair::Bucket, [335](#)
 - stdair::FareOptionStruct, [431](#)
 - stdair::LegCabin, [491](#)
- setAvailabilityPool
 - stdair::LegCabin, [491](#)
 - stdair::SegmentCabin, [590](#)
- setAverageYield
 - stdair::YieldRange, [678](#)
- setAvgCancellationPercentage
 - stdair::LegCabin, [493](#)
- setBidPriceVector
 - stdair::SegmentCabin, [591](#)
- setBlockSpace
 - stdair::SegmentCabin, [590](#)
- setBoardingDate
 - stdair::LegDate, [507](#)
 - stdair::SegmentDate, [601](#)
- setBoardingDateOffset
 - stdair::SegmentPeriod, [613](#)
- setBoardingTime
 - stdair::LegDate, [508](#)
 - stdair::SegmentDate, [602](#)
 - stdair::SegmentPeriod, [612](#)
- setBookingCounter
 - stdair::SegmentCabin, [590](#)
- setCapacities

- stdair::LegCabin, [491](#)
- setCapacity
 - stdair::SegmentCabin, [589](#)
- setChangeFees
 - stdair::FareOptionStruct, [431](#)
- setChosenFareOption
 - stdair::TravelSolutionStruct, [662](#)
- setCommittedSpace
 - stdair::LegCabin, [491](#)
 - stdair::SegmentCabin, [590](#)
- setCumulatedBookingLimit
 - stdair::BookingClass, [317](#)
 - stdair::VirtualClassStruct, [669](#)
- setCumulatedProtection
 - stdair::BookingClass, [317](#)
 - stdair::VirtualClassStruct, [669](#)
- setCurrentBidPrice
 - stdair::LegCabin, [492](#)
- setCurrentNb
 - stdair::ProgressStatus, [565](#)
- setCurrentNbOfEvents
 - stdair::EventQueue, [381](#)
- setDateRange
 - stdair::PeriodStruct, [555](#)
- setDayOfWeek
 - stdair::DoWStruct, [372](#)
- setDBName
 - stdair::BasDBParams, [263](#)
- setDemandInformation
 - stdair::OnDDate, [529](#)
- setDistance
 - stdair::SegmentDate, [602](#)
- setDoW
 - stdair::PeriodStruct, [555](#)
- setElapsedTime
 - stdair::LegDate, [508](#)
 - stdair::SegmentDate, [602](#)
 - stdair::SegmentPeriod, [613](#)
- setETB
 - stdair::LegCabin, [493](#)
- setExpectedNb
 - stdair::ProgressStatus, [566](#)
- setExpectedTotalNbOfEvents
 - stdair::EventQueue, [381](#)
- setFare
 - stdair::AirlineClassList, [240](#)
 - stdair::FareOptionStruct, [431](#)
- setForcedInitialisationFlag
 - stdair::BasLogParams, [267](#)
- setGrossAvailability
 - stdair::LegCabin, [493](#)
- setGroupNbOfSeats
 - stdair::LegCabin, [493](#)
- setGuillotineBlock
 - stdair::SegmentCabin, [589](#)
- setHost
 - stdair::BasDBParams, [263](#)
- SetKeymap
 - swift::SReadline, [637](#)
- setLowerYield
 - stdair::YieldRange, [679](#)
- setMean
 - stdair::BookingClass, [318](#)
 - stdair::VirtualClassStruct, [669](#)
- setMIN
 - stdair::SegmentCabin, [590](#)
- setNetAvailability
 - stdair::LegCabin, [493](#)
- setNonRefundable
 - stdair::FareOptionStruct, [431](#)
- setOffDate
 - stdair::LegDate, [508](#)
 - stdair::SegmentDate, [602](#)
- setOffDateOffset
 - stdair::SegmentPeriod, [613](#)
- setOffPoint
 - stdair::LegDate, [507](#)
- setOffTime
 - stdair::LegDate, [508](#)
 - stdair::SegmentDate, [602](#)
 - stdair::SegmentPeriod, [613](#)
- setOverallStatus
 - stdair::ProgressStatusSet, [569](#)
- setPassword
 - stdair::BasDBParams, [262](#)
- setPort
 - stdair::BasDBParams, [263](#)
- setPreviousBidPrice
 - stdair::LegCabin, [492](#)
- setProtection
 - stdair::BookingClass, [317](#)
- setRegradeAdjustment
 - stdair::LegCabin, [492](#)
- setSaturdayStay
 - stdair::FareOptionStruct, [431](#)
- setSegmentAvailability
 - stdair::BookingClass, [318](#)
- setSoldSeat
 - stdair::LegCabin, [491](#)
- setSoldSeats

- stdair::Bucket, [335](#)
- setSpecificGeneratorStatus
 - stdair::ProgressStatusSet, [569](#)
- setStaffNbOfSeats
 - stdair::LegCabin, [493](#)
- setStatus
 - stdair::EventQueue, [380](#), [381](#)
- setStdDev
 - stdair::BookingClass, [318](#)
 - stdair::VirtualClassStruct, [669](#)
- setTotalForecast
 - stdair::OnDDate, [530](#)
- setTypeSpecificStatus
 - stdair::ProgressStatusSet, [569](#)
- setUpperYield
 - stdair::YieldRange, [678](#)
- setUPR
 - stdair::LegCabin, [492](#)
 - stdair::SegmentCabin, [590](#)
- setUser
 - stdair::BasDBParams, [262](#)
- setWLNbOfSeats
 - stdair::LegCabin, [493](#)
- setYield
 - stdair::AirlineClassList, [240](#)
 - stdair::BookingClass, [318](#)
 - stdair::VirtualClassStruct, [669](#)
- setYieldRangeUpperValue
 - stdair::Bucket, [334](#)
- shift
 - stdair::DoWStruct, [372](#)
- sign_bit
 - FloatingPoint, [451](#)
- SKD_CHG
 - stdair::EventType, [396](#)
- SKeymap
 - swift::SKeymap, [626](#)
- SNAPSHOT
 - stdair::EventType, [396](#)
- SnapshotBlock_T
 - stdair, [204](#)
- SnapshotBlockRange_T
 - stdair, [204](#)
- SnapshotPtr_T
 - stdair, [188](#)
- SnapshotStruct
 - stdair::SnapshotStruct, [628](#)
- soci, [158](#)
- soci::type_conversion< stdair::AirlineStruct
>, [664](#)
- base_type, [664](#)
- from_base, [664](#)
- to_base, [664](#)
- SQLDatabaseConnectionImpossibleException
 - stdair::SQLDatabaseConnectionImpossibleException,
[630](#)
- SQLDatabaseException
 - stdair::SQLDatabaseException, [631](#)
- SReadline
 - swift::SKeymap, [627](#)
 - swift::SReadline, [633](#)
- STA_ORD
 - stdair::DemandGenerationMethod, [365](#)
- start
 - stdair::BasChronometer, [260](#)
- stdair, [158](#)
 - AIRLINE_CODE_BA, [210](#), [228](#)
 - AirlineClassListDetailedList_T, [180](#)
 - AirlineClassListKey::serialize< ba::text_
iarchive >, [219](#)
 - AirlineClassListKey::serialize< ba::text_
oarchive >, [219](#)
 - AirlineClassListList_T, [179](#)
 - AirlineClassListMap_T, [180](#)
 - AirlineClassListWithKey_T, [180](#)
 - AirlineCode_T, [191](#)
 - AirlineCodeList_T, [200](#)
 - AirlineFeatureList_T, [180](#)
 - AirlineFeatureMap_T, [180](#)
 - AirlinePreferenceId_T, [197](#)
 - AIRPORT_BKK, [211](#), [229](#)
 - AIRPORT_LHR, [209](#), [227](#)
 - AIRPORT_SIN, [211](#), [229](#)
 - AIRPORT_SYD, [209](#), [228](#)
 - AirportCode_T, [190](#)
 - AirportPairDetailedList_T, [180](#)
 - AirportPairList_T, [180](#)
 - AirportPairMap_T, [180](#)
 - AirportPairWithKey_T, [180](#)
 - AuthorizationLevel_T, [201](#)
 - Availability_T, [193](#)
 - AvailabilityStatus_T, [201](#)
 - base_iterator_t, [178](#)
 - BaseGenerator_T, [205](#)
 - BidPrice_T, [203](#)
 - BidPriceVector_T, [203](#)
 - BidPriceVectorHolder_T, [189](#)
 - BlockIndex_T, [205](#)
 - BlockNumber_T, [205](#)
 - BlockSpace_T, [201](#)

- BomRoot::serialize< ba::text_iarchive
>, [222](#)
- BomRoot::serialize< ba::text_oarchive
>, [222](#)
- BomRootKey::serialize< ba::text_iarchive
>, [220](#)
- BomRootKey::serialize< ba::text_oarchive
>, [220](#)
- BookingClassList_T, [181](#)
- BookingClassMap_T, [181](#)
- BookingLimit_T, [201](#)
- BookingLimitVector_T, [204](#)
- BookingRatio_T, [202](#)
- BookingRequestPtr_T, [181](#)
- BookingTSIDMap_T, [198](#)
- BOOST_DEFAULT_DATE_PERIOD, [208](#),
[233](#)
- BucketAvailabilities_T, [201](#)
- BucketKey::serialize< ba::text_iarchive
>, [221](#)
- BucketKey::serialize< ba::text_oarchive
>, [221](#)
- BucketList_T, [181](#)
- BucketMap_T, [181](#)
- CABIN_ECO, [211](#), [229](#)
- CABIN_Y, [210](#), [228](#)
- CabinBookingClassMap_T, [200](#)
- CabinCapacity_T, [200](#)
- CabinClassPair_T, [198](#)
- CabinClassPairList_T, [198](#)
- CabinCode_T, [191](#)
- CabinForecastMap_T, [186](#)
- CabinForecastPair_T, [186](#)
- CancellationNoShowRatePair_T, [197](#)
- CancellationPtr_T, [181](#)
- CancellationRateCurve_T, [197](#)
- CapacityAdjustment_T, [201](#)
- CensorshipFlag_T, [202](#)
- CensorshipFlagList_T, [202](#)
- CHANGE_FEES, [210](#), [228](#)
- ChangeFees_T, [196](#)
- ChangeFeesRatio_T, [196](#)
- CHANNEL_DN, [216](#), [234](#)
- CHANNEL_IN, [216](#), [234](#)
- ChannelLabel_T, [198](#)
- CharacteristicsIndex_T, [197](#)
- CharacteristicsPatternId_T, [197](#)
- CharacteristicsWTP_tuple_T, [197](#)
- CityCode_T, [190](#)
- CLASS_CODE_Q, [211](#), [229](#)
- CLASS_CODE_Y, [210](#), [228](#)
- ClassAvailabilityMap_T, [188](#)
- ClassAvailabilityMapHolder_T, [189](#)
- ClassBpvMap_T, [189](#)
- ClassBpvMapHolder_T, [189](#)
- ClassCode_T, [191](#)
- ClassCodeList_T, [200](#)
- ClassList_String_T, [193](#)
- ClassList_StringList_T, [200](#)
- ClassYieldMap_T, [189](#)
- ClassYieldMapHolder_T, [189](#)
- CommittedSpace_T, [201](#)
- ConstSegmentCabinDTDRangeSnapshotView_
T, [205](#)
- ConstSegmentCabinDTDSnapshotView_
T, [204](#)
- ControlMode_T, [203](#)
- Count_T, [192](#)
- DATE_20110115, [209](#)
- DATE_20111231, [209](#)
- Date_T, [194](#)
- DateOffset_T, [195](#)
- DatePeriod_T, [195](#)
- DatePeriodDetailedList_T, [182](#)
- DatePeriodList_T, [182](#)
- DatePeriodMap_T, [182](#)
- DatePeriodWithKey_T, [182](#)
- DateTime_T, [195](#)
- day_p_t, [179](#)
- day_t, [178](#)
- DayDuration_T, [195](#)
- DBConnectionName_T, [196](#)
- DBRequestStatement_T, [196](#)
- DBSession_T, [196](#)
- DCP_T, [202](#)
- DCPList_T, [202](#)
- DEFAULT_ADVANCE_PURCHASE, [215](#),
[233](#)
- DEFAULT_AIRLINE_CODE, [217](#), [231](#)
- DEFAULT_AIRLINE_CODE_LIST, [223](#)
- DEFAULT_AIRPORT_CODE, [217](#), [231](#)
- DEFAULT_AVAILABILITY, [212](#), [226](#)
- DEFAULT_AVAILABILITY_STATUS, [217](#),
[235](#)
- DEFAULT_BID_PRICE, [218](#), [232](#)
- DEFAULT_BID_PRICE_VECTOR, [223](#)
- DEFAULT_BLOCK_SPACE, [212](#), [226](#)
- DEFAULT_BOM_ROOT_KEY, [208](#), [229](#)
- DEFAULT_BOM_TREE_CHANGE_FEES,
[210](#)

- DEFAULT_BOM_TREE_NON_REFUNDABLE, 210
 DEFAULT_BOM_TREE_SATURDAY_STAY, 210
 DEFAULT_BOOKING_NUMBER, 214
 DEFAULT_CABIN_CAPACITY, 212, 226
 DEFAULT_CABIN_CODE, 218, 232
 DEFAULT_CHANNEL, 216, 231
 DEFAULT_CLASS_AUTHORIZATION_LEVEL, 213, 226
 DEFAULT_CLASS_AVAILABILITY, 235
 DEFAULT_CLASS_BOOKING_LIMIT, 213, 226
 DEFAULT_CLASS_CENSORSHIPFLAG, 213, 226
 DEFAULT_CLASS_CENSORSHIPFLAG_LIST, 223
 DEFAULT_CLASS_CODE, 218, 232
 DEFAULT_CLASS_CODE_LIST, 223
 DEFAULT_CLASS_FARE_VALUE, 230
 DEFAULT_CLASS_MAX_AUTHORIZATION_LEVEL, 213, 226
 DEFAULT_CLASS_MIN_AUTHORIZATION_LEVEL, 213, 227
 DEFAULT_CLASS_NB_OF_BOOKINGS, 211, 225
 DEFAULT_CLASS_NB_OF_CANCELLATIONS, 212, 226
 DEFAULT_CLASS_NB_OF_NOSHOWS, 212, 226
 DEFAULT_CLASS_OVERBOOKING_RATE, 213, 227
 DEFAULT_CLASS_REMAINING_DEMAND_MEAN, 212, 225
 DEFAULT_CLASS_REMAINING_DEMAND_STANDARD_DEVIATION, 212, 225
 DEFAULT_CLASS_TOTAL_NB_OF_BOOKINGS, 211, 225
 DEFAULT_CLASS_UNCONSTRAINED_DEMAND, 212, 225
 DEFAULT_CLASS_YIELD_VALUE, 213
 DEFAULT_CLOSED_CLASS_CODE, 211, 225
 DEFAULT_COMMITTED_SPACE, 212, 226
 DEFAULT_CURRENCY, 217, 227
 DEFAULT_DATE, 209, 230
 DEFAULT_DATE_OFFSET, 209, 233
 DEFAULT_DATETIME, 209, 230
 DEFAULT_DAY_DURATION, 208, 227
 DEFAULT_DCP_LIST, 224
 DEFAULT_DEPARTURE_DATE, 217, 231
 DEFAULT_DESTINATION, 217, 232
 DEFAULT_DICO_STUDIED_AIRLINE, 235
 DEFAULT_DICO_STUDIED_DATE, 223
 DEFAULT_DISTANCE_VALUE, 211, 225
 DEFAULT_DOW_STRING, 208, 233
 DEFAULT_DTD_FRAT5COEF_MAP, 224
 DEFAULT_DTD_PROB_MAP, 224
 DEFAULT_EPSILON_DURATION, 209, 230
 DEFAULT_EPSILON_VALUE, 208, 227
 DEFAULT_EVENT_OLDEST_DATE, 215, 229
 DEFAULT_EVENT_OLDEST_DATETIME, 215, 229
 DEFAULT_EVENT_QUEUE_ID, 214, 229
 DEFAULT_FAMILY_CODE, 211
 DEFAULT_FARE_FAMILY_CODE, 218, 232
 DEFAULT_FARE_FAMILY_VALUE_TYPE, 218, 232
 DEFAULT_FARE_VALUE, 213, 227
 DEFAULT_FF_TIER, 216, 234
 DEFAULT_FLIGHT_NUMBER, 217, 231
 DEFAULT_FLIGHT_SPEED, 208, 230
 DEFAULT_FLIGHTPATH_CODE, 211, 235
 DEFAULT_GUILLOTINE_NUMBER, 217, 231
 DEFAULT_KEY_FLD_DELIMITER, 219, 224
 DEFAULT_KEY_SUB_FLD_DELIMITER, 219, 224
 DEFAULT_KEY_TOKEN_DELIMITER, 219, 225
 DEFAULT_LOAD_FACTOR_VALUE, 213, 227
 DEFAULT_MATCHING_INDICATOR, 217, 235
 DEFAULT_MAX_DTD, 223
 DEFAULT_MAXIMAL_CONNECTION_TIME, 216, 234
 DEFAULT_MINIMAL_CONNECTION_TIME, 216, 234
 DEFAULT_NB_OF_DAYS_IN_A_YEAR, 208, 230

- DEFAULT_NB_OF_FLIGHTDATES, 208, 229
- DEFAULT_NBOFAIRLINES, 211, 230
- DEFAULT_NULL_AIRLINE_CODE, 217, 231
- DEFAULT_NULL_AIRPORT_CODE, 217, 231
- DEFAULT_NULL_AVAILABILITY, 212, 226
- DEFAULT_NULL_CLASS_CODE, 218, 232
- DEFAULT_NULL_FARE_FAMILY_CODE, 218, 232
- DEFAULT_NUMBER_OF_REQUIRED_SEATS, 235
- DEFAULT_NUMBER_OF_SUBDIVISIONS, 208, 231
- DEFAULT_OND_BOOKING_RATE, 213, 230
- DEFAULT_OND_FARE_VALUE, 214
- DEFAULT_OND_STRING_LIST, 224
- DEFAULT_ORIGIN, 217, 231
- DEFAULT_PARTY_SIZE, 215, 233
- DEFAULT_POS, 216, 233
- DEFAULT_PREFERRED_CABIN, 216, 234
- DEFAULT_PREFERRED_DEPARTURE_DATE, 215, 233
- DEFAULT_PREFERRED_DEPARTURE_TIME, 215, 233
- DEFAULT_PROGRESS_STATUS, 214, 229
- DEFAULT_RANDOM_SEED, 209, 230
- DEFAULT_REQUEST_DATE, 215, 233
- DEFAULT_REQUEST_DATE_TIME, 215, 234
- DEFAULT_REQUEST_TIME, 215, 233
- DEFAULT_REVENUE_VALUE, 213, 227
- DEFAULT_SEAT_INDEX, 218, 232
- DEFAULT_SEGMENT_CABIN_VALUE_TYPE, 218, 233
- DEFAULT_STAY_DURATION, 215, 233
- DEFAULT_VALUE_OF_TIME, 216, 234
- DEFAULT_WTP, 215, 233
- DEFAULT_YIELD_AVAILABILITY, 214
- DEFAULT_YIELD_BOOKING_LIMIT, 214
- DEFAULT_YIELD_CENSORSHIPFLAG, 214
- DEFAULT_YIELD_MAX_VALUE, 214, 235
- DEFAULT_YIELD_NB_OF_BOOKINGS, 214
- DEFAULT_YIELD_NB_OF_CANCELLATIONS, 214
- DEFAULT_YIELD_NB_OF_NOSHOWS, 214
- DEFAULT_YIELD_OVERBOOKING_RATE, 214
- DEFAULT_YIELD_VALUE, 214, 235
- DemandGeneratorKey_T, 181
- DemandStreamKeyStr_T, 198
- DictionaryKey_T, 179
- DISPLAY_LEVEL_STRING_ARRAY, 224
- Distance_T, 190
- DistributionPatternId_T, 197
- DOW_STR, 222
- DOW_String_T, 195
- DTD_T, 202
- DTDFratMap_T, 202
- DTDProbMap_T, 202
- Duration_T, 194
- EmsrValueList_T, 204
- EventGeneratorKey_T, 199
- EventList_T, 182
- EventListElement_T, 182
- EventName_T, 199
- EventQueueID_T, 199
- EventQueueList_T, 182
- EventQueueMap_T, 182
- ExponentialDistribution_T, 206
- ExponentialGenerator_T, 206
- ExponentialSeed_T, 205
- FamilyCode_T, 191
- Fare_T, 194
- FareFamilyKey::serialize< ba::text_iarchive >, 221
- FareFamilyKey::serialize< ba::text_oarchive >, 221
- FareFamilyList_T, 182
- FareFamilyMap_T, 182
- FareFeaturesDetailedList_T, 183
- FareFeaturesList_T, 183
- FareFeaturesMap_T, 183
- FareFeaturesWithKey_T, 183
- FareOptionList_T, 183
- FileAddress_T, 194
- Filename_T, 194
- Flag_T, 194
- FlightDate::serialize< ba::text_iarchive >, 222

- FlightDate::serialize< ba::text_oarchive >, [222](#)
- FlightDateKey::serialize< ba::text_iarchive >, [221](#)
- FlightDateKey::serialize< ba::text_oarchive >, [221](#)
- FlightDateList_T, [183](#)
- FlightDateMap_T, [183](#)
- FlightNumber_T, [191](#)
- FlightPathCode_T, [200](#)
- FlightPeriodList_T, [183](#)
- FlightPeriodMap_T, [183](#)
- FloatDuration_T, [195](#)
- ForecasterMode_T, [207](#)
- FREQUENT_FLYER_MEMBER, [211](#), [229](#)
- FrequentFlyer_T, [198](#)
- GeneratedDemandVector_T, [207](#)
- GeneratedDemandVectorHolder_T, [207](#)
- GuillotineBlockKey::serialize< ba::text_iarchive >, [221](#)
- GuillotineBlockKey::serialize< ba::text_oarchive >, [221](#)
- GuillotineBlockList_T, [184](#)
- GuillotineBlockMap_T, [184](#)
- GuillotineNumber_T, [191](#)
- HistoricalDataLimit_T, [207](#)
- HolderMap_T, [181](#)
- hour_p_t, [179](#)
- hour_t, [178](#)
- Identity_T, [191](#)
- int1_p_t, [178](#)
- intDisplay, [220](#)
- IntDuration_T, [195](#)
- Inventory::serialize< ba::text_iarchive >, [222](#)
- Inventory::serialize< ba::text_oarchive >, [222](#)
- InventoryKey::serialize< ba::text_iarchive >, [221](#)
- InventoryKey::serialize< ba::text_oarchive >, [221](#)
- InventoryList_T, [184](#)
- InventoryMap_T, [184](#)
- iterator_t, [178](#)
- jsonBookingClassExport, [220](#)
- jsonBucketExport, [220](#)
- jsonFareFamilyExport, [220](#)
- jsonLegCabinExport, [220](#)
- jsonLegDateExport, [219](#)
- jsonSegmentCabinExport, [220](#)
- jsonSegmentDateExport, [220](#)
- KeyDescription_T, [191](#)
- KeyList_T, [184](#)
- LegCabinList_T, [184](#)
- LegCabinMap_T, [185](#)
- LegDateList_T, [185](#)
- LegDateMap_T, [185](#)
- LocationCode_T, [190](#)
- LongDuration_T, [195](#)
- MapKey_T, [184](#)
- MatchingIndicator_T, [198](#)
- MAXIMAL_AVAILABILITY, [212](#), [232](#)
- MAXIMAL_NUMBER_OF_LEGS_IN_FLIGHT, [218](#), [232](#)
- MAXIMAL_NUMBER_OF_SEGMENTS_IN_OND, [218](#), [232](#)
- MeanStdDevPair_T, [206](#)
- MeanValue_T, [206](#)
- MILLISECONDS_IN_ONE_SECOND, [209](#), [230](#)
- minute_p_t, [179](#)
- minute_t, [178](#)
- MonetaryValue_T, [192](#)
- month_p_t, [179](#)
- month_t, [178](#)
- Multiplier_T, [192](#)
- NbOfAirlines_T, [193](#)
- NbOfBookings_T, [193](#)
- NbOfCancellations_T, [193](#)
- NbOfEvents_T, [199](#)
- NbOfFareRules_T, [199](#)
- NbOfFlightDates_T, [200](#)
- NbOfInventoryControlRules_T, [202](#)
- NbOfNoShows_T, [198](#)
- NbOfRequests_T, [193](#)
- NbOfSeats_T, [192](#)
- NbOfSegments_T, [193](#)
- NbOfTravelSolutions_T, [193](#)
- NbOfYields_T, [201](#)
- NetworkID_T, [200](#)
- NO_ADVANCE_PURCHASE, [209](#), [228](#)
- NO_CHANGE_FEES, [210](#), [228](#)
- NO_NON_REFUNDABLE, [228](#)
- No_NON_REFUNDABLE, [210](#)
- NO_SATURDAY_STAY, [210](#), [228](#)
- NO_STAY_DURATION, [210](#), [228](#)
- NON_REFUNDABLE, [210](#), [228](#)
- NonRefundable_T, [196](#)
- NonRefundableRatio_T, [196](#)

- NormalDistribution_T, 206
- NormalGenerator_T, 206
- NULL_BOOST_TIME_DURATION, 208, 230
- OnDDateKey::serialize< ba::text_iarchive >, 221
- OnDDateKey::serialize< ba::text_oarchive >, 221
- OnDDateList_T, 185
- OnDDateMap_T, 185
- OnDString_T, 199
- OnDStringList_T, 199
- operator*, 219
- operator+, 219
- OptimisationNotificationPtr_T, 186
- OptimizerMode_T, 207
- OverbookingRate_T, 203
- PartySize_T, 193
- PassengerType_T, 196
- Percentage_T, 192
- PolicyDemand_T, 207
- POS_LHR, 209, 228
- POS_SIN, 211, 229
- PosChannelDetailedList_T, 186
- PosChannelList_T, 186
- PosChannelMap_T, 186
- PosChannelWithKey_T, 186
- PriceCurrency_T, 192
- PriceValue_T, 192
- Probability_T, 206
- ProgressPercentage_T, 194
- ProportionFactor_T, 199
- ProportionFactorList_T, 199
- ProtectionLevel_T, 203
- ProtectionLevelVector_T, 204
- RandomSeed_T, 205
- RealNumber_T, 192
- ReplicationNumber_T, 205
- RequestStatus_T, 198
- Revenue_T, 192
- RMEventList_T, 187
- RMEventPtr_T, 186
- SATURDAY_STAY, 210, 228
- SaturdayStay_T, 195
- SaturdayStayRatio_T, 196
- SeatIndex_T, 203
- second_p_t, 179
- second_t, 178
- SECONDS_IN_ONE_DAY, 209, 230
- SegmentCabin::serialize< ba::text_iarchive >, 222
- SegmentCabin::serialize< ba::text_oarchive >, 222
- SegmentCabinDTDRangeSnapshotView_T, 204
- SegmentCabinDTDSnapshotView_T, 204
- SegmentCabinIndexMap_T, 184
- SegmentCabinKey::serialize< ba::text_iarchive >, 222
- SegmentCabinKey::serialize< ba::text_oarchive >, 221
- SegmentCabinList_T, 187
- SegmentCabinMap_T, 187
- SegmentDate::serialize< ba::text_iarchive >, 222
- SegmentDate::serialize< ba::text_oarchive >, 222
- SegmentDateKey::serialize< ba::text_iarchive >, 222
- SegmentDateKey::serialize< ba::text_oarchive >, 222
- SegmentDateList_T, 187
- SegmentDateMap_T, 187
- SegmentPath_T, 188
- SegmentPathList_T, 188
- SegmentPeriodDetailedList_T, 187
- SegmentPeriodList_T, 187
- SegmentPeriodMap_T, 187
- SegmentPeriodWithKey_T, 187
- SellupFactorHolder_T, 207
- SellupProbability_T, 207
- SellupProbabilityVector_T, 207
- serialiseHelper, 222
- SnapshotBlock_T, 204
- SnapshotBlockRange_T, 204
- SnapshotPtr_T, 188
- STDAIR_ServicePtr_T, 208
- StdDevValue_T, 206
- StringCabinClassPair_T, 186
- StringCabinClassPairListMap_T, 185
- StringDemandStructMap_T, 185
- StringDemandStructPair_T, 185
- SubclassCode_T, 200
- Time_T, 194
- TimePeriodDetailedList_T, 188
- TimePeriodList_T, 188
- TimePeriodMap_T, 188
- TimePeriodWithKey_T, 188

- Tokeniser_T, 181
- TokeniserDashSeparator, 221
- TokeniserTimeSeparator, 221
- TravelSolutionList_T, 188
- TRIP_TYPE_INBOUND, 216, 234
- TRIP_TYPE_ONE_WAY, 216, 234
- TRIP_TYPE_OUTBOUND, 216, 234
- TRIP_TYPE_ROUND_TRIP, 216, 234
- TripType_T, 191
- uint1_4_p_t, 179
- uint2_p_t, 179
- uint4_p_t, 179
- UniformDistribution_T, 205
- UniformGenerator_T, 206
- UniformSeed_T, 205
- UnsignedIndex_T, 194
- UPR_T, 201
- ValueTypeIndexMap_T, 184
- VirtualClassList_T, 189
- VirtualClassMap_T, 189
- WTP_T, 197
- WTPDemandPair_T, 197
- year_p_t, 179
- year_t, 178
- Yield_T, 202
- YieldDemandPair_T, 203
- YieldFeaturesDetailedList_T, 190
- YieldFeaturesList_T, 190
- YieldFeaturesMap_T, 190
- YieldFeaturesWithKey_T, 190
- YieldLevel_T, 203
- YieldLevelDemandMap_T, 203
- YieldStoreList_T, 190
- YieldStoreMap_T, 190
- YieldValue_T, 192
- stdair-paths.hpp
 - BINDIR, 1169
 - DATADIR, 1170
 - DATAROOTDIR, 1170
 - DOCDIR, 1170
 - EXEC_PREFIX, 1169
 - HTMLDIR, 1170
 - INCLUDEDIR, 1169
 - INFODIR, 1170
 - LIBDIR, 1169
 - LIBEXECDIR, 1169
 - MANDIR, 1170
 - PACKAGE, 1169
 - PACKAGE_NAME, 1169
 - PACKAGE_VERSION, 1169
 - PDFDIR, 1170
 - PREFIXDIR, 1169
 - SBINDIR, 1169
 - STDAIR_SAMPLE_DIR, 1170
 - SYSCONFDIR, 1169
- stdair/ Directory Reference, 157
- stdair/basic/ Directory Reference, 150
- stdair/basic/BasChronometer.cpp, 690
- stdair/basic/BasChronometer.hpp, 691, 692
- stdair/basic/BasConst.cpp, 692, 696
- stdair/basic/BasConst_BomDisplay.hpp, 703, 704
- stdair/basic/BasConst_BookingClass.hpp, 704, 705
- stdair/basic/BasConst_DefaultObject.hpp, 706, 707
- stdair/basic/BasConst_Event.hpp, 708, 709
- stdair/basic/BasConst_General.hpp, 709, 710
- stdair/basic/BasConst_Inventory.hpp, 711, 712
- stdair/basic/BasConst_Period_BOM.hpp, 713
- stdair/basic/BasConst_Request.hpp, 714, 715
- stdair/basic/BasConst_TravelSolution.hpp, 716
- stdair/basic/BasConst_Yield.hpp, 717
- stdair/basic/BasDBParams.cpp, 717, 718
- stdair/basic/BasDBParams.hpp, 719
- stdair/basic/BasFileMgr.cpp, 721
- stdair/basic/BasFileMgr.hpp, 722
- stdair/basic/BasLogParams.cpp, 723
- stdair/basic/BasLogParams.hpp, 724
- stdair/basic/BasParserHelperTypes.hpp, 726, 727
- stdair/basic/BasParserTypes.hpp, 728
- stdair/basic/BasTypes.hpp, 729
- stdair/basic/ContinuousAttributeLite.hpp, 730
- stdair/basic/DemandGenerationMethod.cpp, 734
- stdair/basic/DemandGenerationMethod.hpp, 737
- stdair/basic/DictionaryManager.cpp, 738, 739
- stdair/basic/DictionaryManager.hpp, 739, 740
- stdair/basic/EventType.cpp, 740
- stdair/basic/EventType.hpp, 742, 743
- stdair/basic/float_utils.hpp, 744
- stdair/basic/float_utils_google.hpp, 744
- stdair/basic/ForecastingMethod.cpp, 749
- stdair/basic/ForecastingMethod.hpp, 751

- stdair/basic/PartnershipTechnique.cpp, 752, 753
- stdair/basic/PartnershipTechnique.hpp, 755, 756
- stdair/basic/PassengerType.cpp, 757
- stdair/basic/PassengerType.hpp, 759
- stdair/basic/ProgressStatus.cpp, 760
- stdair/basic/ProgressStatus.hpp, 762
- stdair/basic/ProgressStatusSet.cpp, 764
- stdair/basic/ProgressStatusSet.hpp, 765, 766
- stdair/basic/RandomGeneration.cpp, 767
- stdair/basic/RandomGeneration.hpp, 769
- stdair/basic/SampleType.cpp, 770, 771
- stdair/basic/SampleType.hpp, 773
- stdair/basic/ServiceInitialisationType.cpp, 774
- stdair/basic/ServiceInitialisationType.hpp, 775
- stdair/basic/StructAbstract.hpp, 778, 779
- stdair/basic/YieldRange.cpp, 780, 781
- stdair/basic/YieldRange.hpp, 782
- stdair/bom/ Directory Reference, 151
- stdair/bom/AirlineClassList.cpp, 783, 784
- stdair/bom/AirlineClassList.hpp, 785
- stdair/bom/AirlineClassListKey.cpp, 788
- stdair/bom/AirlineClassListKey.hpp, 790
- stdair/bom/AirlineClassListTypes.hpp, 792
- stdair/bom/AirlineFeature.cpp, 793
- stdair/bom/AirlineFeature.hpp, 794
- stdair/bom/AirlineFeatureKey.cpp, 795
- stdair/bom/AirlineFeatureKey.hpp, 796
- stdair/bom/AirlineFeatureTypes.hpp, 797, 798
- stdair/bom/AirlineStruct.cpp, 798
- stdair/bom/AirlineStruct.hpp, 799, 800
- stdair/bom/AirportPair.cpp, 801
- stdair/bom/AirportPair.hpp, 802
- stdair/bom/AirportPairKey.cpp, 804
- stdair/bom/AirportPairKey.hpp, 805
- stdair/bom/AirportPairTypes.hpp, 806, 807
- stdair/bom/BomAbstract.hpp, 807, 808
- stdair/bom/BomArchive.cpp, 809, 810
- stdair/bom/BomArchive.hpp, 811, 812
- stdair/bom/BomDisplay.cpp, 812
- stdair/bom/BomDisplay.hpp, 831
- stdair/bom/BomHolder.hpp, 833
- stdair/bom/BomHolderKey.cpp, 834, 835
- stdair/bom/BomHolderKey.hpp, 835, 836
- stdair/bom/BomJSONExport.cpp, 836, 837
- stdair/bom/BomJSONExport.hpp, 845
- stdair/bom/BomJSONImport.cpp, 846
- stdair/bom/BomJSONImport.hpp, 847, 848
- stdair/bom/BomKeyManager.cpp, 848, 849
- stdair/bom/BomKeyManager.hpp, 851
- stdair/bom/BomManager.hpp, 852
- stdair/bom/BomRetriever.cpp, 857, 858
- stdair/bom/BomRetriever.hpp, 866
- stdair/bom/BomRoot.cpp, 868
- stdair/bom/BomRoot.hpp, 869, 870
- stdair/bom/BomRootKey.cpp, 871, 872
- stdair/bom/BomRootKey.hpp, 873, 874
- stdair/bom/BookingClass.cpp, 875
- stdair/bom/BookingClass.hpp, 877
- stdair/bom/BookingClassKey.cpp, 881, 882
- stdair/bom/BookingClassKey.hpp, 883
- stdair/bom/BookingClassTypes.hpp, 884
- stdair/bom/BookingRequestStruct.cpp, 885
- stdair/bom/BookingRequestStruct.hpp, 889
- stdair/bom/BookingRequestTypes.hpp, 892
- stdair/bom/Bucket.cpp, 893
- stdair/bom/Bucket.hpp, 894, 895
- stdair/bom/BucketKey.cpp, 897
- stdair/bom/BucketKey.hpp, 899
- stdair/bom/BucketTypes.hpp, 900, 901
- stdair/bom/CancellationStruct.cpp, 901, 902
- stdair/bom/CancellationStruct.hpp, 903
- stdair/bom/CancellationTypes.hpp, 905
- stdair/bom/DatePeriod.cpp, 905, 906
- stdair/bom/DatePeriod.hpp, 907
- stdair/bom/DatePeriodKey.cpp, 908, 909
- stdair/bom/DatePeriodKey.hpp, 910
- stdair/bom/DatePeriodTypes.hpp, 911
- stdair/bom/DoWStruct.cpp, 912
- stdair/bom/DoWStruct.hpp, 914, 915
- stdair/bom/EventQueue.cpp, 916
- stdair/bom/EventQueue.hpp, 922
- stdair/bom/EventQueueKey.cpp, 925, 926
- stdair/bom/EventQueueKey.hpp, 926, 927
- stdair/bom/EventQueueTypes.hpp, 927, 928
- stdair/bom/EventStruct.cpp, 928, 929
- stdair/bom/EventStruct.hpp, 933
- stdair/bom/EventTypes.hpp, 935, 936
- stdair/bom/FareFamily.cpp, 936, 937
- stdair/bom/FareFamily.hpp, 938
- stdair/bom/FareFamilyKey.cpp, 940, 941
- stdair/bom/FareFamilyKey.hpp, 942
- stdair/bom/FareFamilyTypes.hpp, 944
- stdair/bom/FareFeatures.cpp, 944, 945
- stdair/bom/FareFeatures.hpp, 947
- stdair/bom/FareFeaturesKey.cpp, 949
- stdair/bom/FareFeaturesKey.hpp, 950, 951
- stdair/bom/FareFeaturesTypes.hpp, 952, 953
- stdair/bom/FareOptionStruct.cpp, 953

- stdair/bom/FareOptionStruct.hpp, 955, 956
stdair/bom/FareOptionTypes.hpp, 957, 958
stdair/bom/FlightDate.cpp, 958, 959
stdair/bom/FlightDate.hpp, 960, 961
stdair/bom/FlightDateKey.cpp, 963
stdair/bom/FlightDateKey.hpp, 965
stdair/bom/FlightDateTypes.hpp, 967
stdair/bom/FlightPeriod.cpp, 967, 968
stdair/bom/FlightPeriod.hpp, 968, 969
stdair/bom/FlightPeriodKey.cpp, 970
stdair/bom/FlightPeriodKey.hpp, 971
stdair/bom/FlightPeriodTypes.hpp, 972
stdair/bom/GuillotineBlock.cpp, 973
stdair/bom/GuillotineBlock.hpp, 979
stdair/bom/GuillotineBlockKey.cpp, 983
stdair/bom/GuillotineBlockKey.hpp, 985
stdair/bom/GuillotineBlockTypes.hpp, 986, 987
stdair/bom/Inventory.cpp, 987, 988
stdair/bom/Inventory.hpp, 989
stdair/bom/InventoryKey.cpp, 991, 992
stdair/bom/InventoryKey.hpp, 993, 994
stdair/bom/InventoryTypes.hpp, 995
stdair/bom/key_types.hpp, 996
stdair/bom/KeyAbstract.hpp, 996, 998
stdair/bom/LegCabin.cpp, 999
stdair/bom/LegCabin.hpp, 1001, 1002
stdair/bom/LegCabinKey.cpp, 1007, 1008
stdair/bom/LegCabinKey.hpp, 1009
stdair/bom/LegCabinTypes.hpp, 1010, 1011
stdair/bom/LegDate.cpp, 1011, 1012
stdair/bom/LegDate.hpp, 1013, 1014
stdair/bom/LegDateKey.cpp, 1017
stdair/bom/LegDateKey.hpp, 1018
stdair/bom/LegDateTypes.hpp, 1019, 1020
stdair/bom/OnDDate.cpp, 1020
stdair/bom/OnDDate.hpp, 1022, 1023
stdair/bom/OnDDateKey.cpp, 1025, 1026
stdair/bom/OnDDateKey.hpp, 1028
stdair/bom/OnDDateTypes.hpp, 1030
stdair/bom/OptimisationNotificationStruct.cpp, 1031
stdair/bom/OptimisationNotificationStruct.hpp, 1033
stdair/bom/OptimisationNotificationTypes.hpp, 1036
stdair/bom/ParsedKey.cpp, 1036, 1037
stdair/bom/ParsedKey.hpp, 1039, 1040
stdair/bom/PeriodStruct.cpp, 1041
stdair/bom/PeriodStruct.hpp, 1042, 1043
stdair/bom/PosChannel.cpp, 1044
stdair/bom/PosChannel.hpp, 1045
stdair/bom/PosChannelKey.cpp, 1047
stdair/bom/PosChannelKey.hpp, 1048
stdair/bom/PosChannelTypes.hpp, 1049, 1050
stdair/bom/RMEventStruct.cpp, 1050
stdair/bom/RMEventStruct.hpp, 1051, 1052
stdair/bom/RMEventTypes.hpp, 1053
stdair/bom/SegmentCabin.cpp, 1054
stdair/bom/SegmentCabin.hpp, 1055, 1056
stdair/bom/SegmentCabinKey.cpp, 1060
stdair/bom/SegmentCabinKey.hpp, 1062
stdair/bom/SegmentCabinTypes.hpp, 1063, 1064
stdair/bom/SegmentDate.cpp, 1064
stdair/bom/SegmentDate.hpp, 1066
stdair/bom/SegmentDateKey.cpp, 1069, 1070
stdair/bom/SegmentDateKey.hpp, 1071, 1072
stdair/bom/SegmentDateTypes.hpp, 1073
stdair/bom/SegmentPeriod.cpp, 1074
stdair/bom/SegmentPeriod.hpp, 1075
stdair/bom/SegmentPeriodKey.cpp, 1077
stdair/bom/SegmentPeriodKey.hpp, 1078
stdair/bom/SegmentPeriodTypes.hpp, 1079, 1080
stdair/bom/SnapshotStruct.cpp, 1080
stdair/bom/SnapshotStruct.hpp, 1081, 1082
stdair/bom/SnapshotTypes.hpp, 1083
stdair/bom/TimePeriod.cpp, 1083, 1084
stdair/bom/TimePeriod.hpp, 1085
stdair/bom/TimePeriodKey.cpp, 1087
stdair/bom/TimePeriodKey.hpp, 1088
stdair/bom/TimePeriodTypes.hpp, 1089, 1090
stdair/bom/TravelSolutionStruct.cpp, 1090, 1091
stdair/bom/TravelSolutionStruct.hpp, 1093, 1094
stdair/bom/TravelSolutionTypes.hpp, 1096
stdair/bom/VirtualClassStruct.cpp, 1097, 1098
stdair/bom/VirtualClassStruct.hpp, 1099
stdair/bom/VirtualClassTypes.hpp, 1101
stdair/bom/YieldFeatures.cpp, 1102
stdair/bom/YieldFeatures.hpp, 1103, 1104
stdair/bom/YieldFeaturesKey.cpp, 1105
stdair/bom/YieldFeaturesKey.hpp, 1106, 1107
stdair/bom/YieldFeaturesTypes.hpp, 1108
stdair/bom/YieldStore.cpp, 1109
stdair/bom/YieldStore.hpp, 1109, 1110
stdair/bom/YieldStoreKey.cpp, 1111
stdair/bom/YieldStoreKey.hpp, 1112

- stdair/bom/YieldStoreTypes.hpp, 1113
- stdair/command/ Directory Reference, 155
- stdair/command/CmdAbstract.cpp, 1114
- stdair/command/CmdAbstract.hpp, 1114
- stdair/command/CmdBomManager.cpp, 1115
- stdair/command/CmdBomManager.hpp, 1158
- stdair/command/CmdBomSerialiser.cpp, 1158, 1160
- stdair/command/CmdBomSerialiser.hpp, 1164
- stdair/command/DBManagerForAirlines.cpp, 1164, 1165
- stdair/command/DBManagerForAirlines.hpp, 1167
- stdair/config/ Directory Reference, 155
- stdair/config/stdair-paths.hpp, 1168, 1170
- stdair/dbadaptor/ Directory Reference, 155
- stdair/dbadaptor/DbAbstract.cpp, 1171
- stdair/dbadaptor/DbAbstract.hpp, 1171, 1172
- stdair/dbadaptor/DbAirline.cpp, 1173
- stdair/dbadaptor/DbAirline.hpp, 1174, 1175
- stdair/factory/ Directory Reference, 156
- stdair/factory/FacAbstract.cpp, 1175
- stdair/factory/FacAbstract.hpp, 1176
- stdair/factory/FacBom.hpp, 1176, 1177
- stdair/factory/FacBomManager.hpp, 1179
- stdair/service/ Directory Reference, 156
- stdair/service/DBSessionManager.cpp, 1185
- stdair/service/DBSessionManager.hpp, 1187
- stdair/service/FacServiceAbstract.cpp, 1188
- stdair/service/FacServiceAbstract.hpp, 1189
- stdair/service/FacSTDAIRServiceContext.cpp, 1190
- stdair/service/FacSTDAIRServiceContext.hpp, 1191
- stdair/service/FacSupervisor.cpp, 1192
- stdair/service/FacSupervisor.hpp, 1194
- stdair/service/Logger.cpp, 1195
- stdair/service/Logger.hpp, 1197, 1198
- stdair/service/ServiceAbstract.cpp, 1200
- stdair/service/ServiceAbstract.hpp, 1200, 1201
- stdair/service/STDAIR_Service.cpp, 1202, 1203
- stdair/service/STDAIR_ServiceContext.cpp, 1213
- stdair/service/STDAIR_ServiceContext.hpp, 1214, 1215
- stdair/stdair_basic_types.hpp, 1216, 1218
- stdair/stdair_date_time_types.hpp, 1219, 1220
- stdair/stdair_db.hpp, 1220, 1221
- stdair/stdair_demand_types.hpp, 1221, 1223
- stdair/stdair_event_types.hpp, 1224
- stdair/stdair_exceptions.hpp, 1225, 1226
- stdair/stdair_fare_types.hpp, 1228
- stdair/stdair_file.hpp, 1229
- stdair/stdair_inventory_types.hpp, 1230, 1232
- stdair/stdair_log.hpp, 1234
- stdair/stdair_maths_types.hpp, 1235, 1236
- stdair/stdair_rm_types.hpp, 1237
- stdair/STDAIR_Service.hpp, 1238
- stdair/stdair_service_types.hpp, 1241
- stdair/stdair_types.hpp, 1241, 1242
- stdair/ui/ Directory Reference, 157
- stdair/ui/cmdline/ Directory Reference, 155
- stdair/ui/cmdline/readline_autocomp.hpp, 1242, 1247
- stdair/ui/cmdline/SReadline.hpp, 1252, 1253
- stdair::AirlineClassList, 237
- ~AirlineClassList, 239
- _fare, 242
- _holderMap, 242
- _key, 241
- _parent, 242
- _yield, 242
- AirlineClassList, 239
- boost::serialization::access, 241
- describeKey, 241
- FacBom, 241
- FacBomManager, 241
- fromStream, 240
- getAirlineCodeList, 239
- getClassCodeList, 239
- getFare, 240
- getHolderMap, 239
- getKey, 239
- getParent, 239
- getYield, 240
- Key_T, 238
- serialize, 241
- setFare, 240
- setYield, 240
- toStream, 240
- toString, 241
- stdair::AirlineClassListKey, 242
- ~AirlineClassListKey, 243
- AirlineClassListKey, 243
- boost::serialization::access, 245
- fromStream, 244
- getAirlineCodeList, 243
- getClassCodeList, 244

- serialize, [244](#)
- toStream, [244](#)
- toString, [244](#)
- stdair::AirlineFeature, [245](#)
 - ~AirlineFeature, [246](#)
 - _controlMode, [248](#)
 - _forecasterMode, [248](#)
 - _historicalDataLimit, [248](#)
 - _key, [248](#)
 - AirlineFeature, [246](#)
 - describeKey, [248](#)
 - FacBom, [248](#)
 - fromStream, [247](#)
 - getKey, [247](#)
 - init, [247](#)
 - Key_T, [246](#)
 - toStream, [247](#)
 - toString, [247](#)
- stdair::AirlineFeatureKey, [249](#)
 - ~AirlineFeatureKey, [249](#)
 - AirlineFeatureKey, [249](#)
 - fromStream, [250](#)
 - getAirlineCode, [249](#)
 - toStream, [250](#)
 - toString, [250](#)
- stdair::AirlineStruct, [251](#)
 - ~AirlineStruct, [252](#)
 - AirlineStruct, [251](#)
 - describe, [253](#)
 - fromStream, [253](#)
 - getAirlineCode, [252](#)
 - getAirlineName, [252](#)
 - setAirlineCode, [252](#)
 - setAirlineName, [252](#)
 - toStream, [252](#)
- stdair::AirportPair, [253](#)
 - ~AirportPair, [255](#)
 - _holderMap, [257](#)
 - _key, [257](#)
 - _parent, [257](#)
 - AirportPair, [255](#)
 - describeKey, [255](#)
 - FacBom, [256](#)
 - FacBomManager, [256](#)
 - fromStream, [255](#)
 - getBoardingPoint, [256](#)
 - getHolderMap, [256](#)
 - getKey, [256](#)
 - getOffPoint, [256](#)
 - getParent, [256](#)
 - Key_T, [254](#)
 - toStream, [255](#)
 - toString, [255](#)
- stdair::AirportPairKey, [257](#)
 - ~AirportPairKey, [258](#)
 - AirportPairKey, [258](#)
 - fromStream, [259](#)
 - getBoardingPoint, [258](#)
 - getOffPoint, [258](#)
 - toStream, [259](#)
 - toString, [259](#)
- stdair::BasChronometer, [259](#)
 - BasChronometer, [260](#)
 - elapsed, [260](#)
 - getStart, [260](#)
 - start, [260](#)
- stdair::BasDBParams, [260](#)
 - ~BasDBParams, [262](#)
 - BasDBParams, [261](#), [262](#)
 - check, [263](#)
 - describe, [263](#)
 - fromStream, [264](#)
 - getDBName, [262](#)
 - getHost, [262](#)
 - getPassword, [262](#)
 - getPort, [262](#)
 - getUser, [262](#)
 - setDBName, [263](#)
 - setHost, [263](#)
 - setPassword, [262](#)
 - setPort, [263](#)
 - setUser, [262](#)
 - toShortString, [263](#)
 - toStream, [263](#)
 - toString, [263](#)
- stdair::BasFileMgr, [264](#)
 - doesExistAndIsReadable, [265](#)
- stdair::BasLogParams, [265](#)
 - ~BasLogParams, [266](#)
 - BasLogParams, [266](#)
 - check, [267](#)
 - describe, [267](#)
 - fromStream, [268](#)
 - getForcedInitialisationFlag, [266](#)
 - getLogLevel, [266](#)
 - getLogStream, [266](#)
 - Logger, [268](#)
 - setForcedInitialisationFlag, [267](#)
 - toShortString, [267](#)
 - toStream, [267](#)

- toString, 267
- stdair::BomAbstract, 268
 - ~BomAbstract, 270
 - BomAbstract, 270
 - fromStream, 270
 - toStream, 270
 - toString, 271
- stdair::BomArchive, 271
 - archive, 271, 272
 - restore, 272
- stdair::BomDisplay, 272
 - csvAirlineClassDisplay, 280
 - csvAirportPairDisplay, 278
 - csvBookingClassDisplay, 277
 - csvBucketDisplay, 277
 - csvDateDisplay, 279
 - csvDisplay, 273–275, 277, 278
 - csvFareFamilyDisplay, 276
 - csvFeatureListDisplay, 279
 - csvFeaturesDisplay, 279
 - csvLegCabinDisplay, 276
 - csvLegDateDisplay, 275
 - csvPosChannelDisplay, 279
 - csvSegmentCabinDisplay, 276
 - csvSegmentDateDisplay, 276
 - csvSimFQTAirRACDisplay, 278
 - csvTimeDisplay, 279
 - list, 274
 - listAirportPairDateRange, 274
- stdair::BomHolder, 280
 - ~BomHolder, 282
 - _bomList, 283
 - _bomMap, 284
 - _key, 283
 - BomHolder, 282
 - BomList_T, 281
 - BomMap_T, 281
 - describeKey, 283
 - FacBom, 283
 - FacBomManager, 283
 - fromStream, 282
 - Key_T, 281
 - toStream, 282
 - toString, 283
- stdair::BomHolderKey, 284
 - ~BomHolderKey, 285
 - BomHolderKey, 285
 - describe, 286
 - fromStream, 285
 - toStream, 285
- toString, 285
- stdair::BomJSONExport, 286
 - jsonExport, 286
- stdair::BomJSONImport, 287
 - jsonImportFlightDateKey, 287
 - jsonImportInventoryKey, 287
- stdair::BomKeyManager, 288
 - extractFlightDateKey, 289
 - extractInventoryKey, 288
 - extractKeys, 288
 - extractSegmentDateKey, 289
- stdair::BomManager, 290
 - FacBomManager, 293
 - getList, 291, 292
 - getMap, 291
 - getObject, 292
 - getObjectPtr, 292
 - getParent, 292
 - getParentPtr, 292
 - hasList, 291, 292
 - hasMap, 291, 293
- stdair::BomRetriever, 293
 - retrieveAirportPairFromKeySet, 300
 - retrieveBookingClassFromLongKey, 299
 - retrieveDatePeriodListFromKey, 300
 - retrieveDatePeriodListFromKeySet, 301
 - retrieveDummyLegCabin, 301
 - retrieveDummySegmentCabin, 301
 - retrieveFlightDateFromKey, 297
 - retrieveFlightDateFromKeySet, 296
 - retrieveFlightDateFromLongKey, 295, 296
 - retrieveInventoryFromKey, 294, 295
 - retrieveInventoryFromLongKey, 294
 - retrieveSegmentDateFromKey, 299
 - retrieveSegmentDateFromLongKey, 297, 298
- stdair::BomRoot, 302
 - ~BomRoot, 303
 - _holderMap, 306
 - _key, 306
 - BomRoot, 303
 - boost::serialization::access, 306
 - describeKey, 305
 - FacBom, 306
 - FacBomManager, 306
 - fromStream, 305
 - getHolderMap, 304
 - getInventory, 304
 - getKey, 304

- Key_T, 303
- serialize, 305
- toStream, 305
- toString, 305
- stdair::BomRootKey, 307
 - ~BomRootKey, 308
 - BomRootKey, 307, 308
 - boost::serialization::access, 309
 - fromStream, 308
 - getID, 308
 - serialize, 309
 - toStream, 308
 - toString, 308
- stdair::BookingClass, 310
 - ~BookingClass, 313
 - _au, 321
 - _cancellationPercentage, 321
 - _cumulatedBookingLimit, 321
 - _cumulatedProtection, 320
 - _etb, 322
 - _generatedDemandVector, 323
 - _groupNbOfBookings, 322
 - _groupPendingNbOfBookings, 322
 - _holderMap, 320
 - _key, 320
 - _mean, 323
 - _nbOfBookings, 321
 - _nbOfCancellations, 322
 - _nego, 321
 - _netClassAvailability, 323
 - _netRevenueAvailability, 323
 - _noShowPercentage, 321
 - _parent, 320
 - _protection, 321
 - _segmentAvailability, 323
 - _staffNbOfBookings, 322
 - _stdDev, 323
 - _subclassCode, 320
 - _wlnNbOfBookings, 322
 - _yield, 323
- BookingClass, 313
- cancel, 319
- describeKey, 319
- FacBom, 320
- FacBomManager, 320
- fromStream, 318
- generateDemandSamples, 319, 320
- getAuthorizationLevel, 314
- getCancellationPercentage, 315
- getClassCode, 313
- getCumulatedBookingLimit, 314
- getCumulatedProtection, 314
- getETB, 316
- getGeneratedDemandVector, 317
- getHolderMap, 313
- getKey, 313
- getMean, 317
- getNbOfBookings, 315
- getNbOfCancellations, 316
- getNbOfGroupBookings, 315
- getNbOfPendingGroupBookings, 315
- getNbOfStaffBookings, 315
- getNbOfWLBookings, 315
- getNegotiatedSpace, 314
- getNetClassAvailability, 316
- getNetRevenueAvailability, 316
- getNoShowPercentage, 315
- getParent, 313
- getProtection, 314
- getSegmentAvailability, 316
- getStdDev, 317
- getSubclassCode, 314
- getYield, 316
- Key_T, 313
- sell, 319
- setAuthorizationLevel, 317
- setCumulatedBookingLimit, 317
- setCumulatedProtection, 317
- setMean, 318
- setProtection, 317
- setSegmentAvailability, 318
- setStdDev, 318
- setYield, 318
- toStream, 318
- toString, 319
- stdair::BookingClassKey, 324
 - ~BookingClassKey, 324
 - BookingClassKey, 324
 - fromStream, 325
 - getClassCode, 325
 - toStream, 325
 - toString, 325
- stdair::BookingRequestStruct, 326
 - ~BookingRequestStruct, 328
 - BookingRequestStruct, 327
 - describe, 330
 - display, 330
 - fromStream, 330
 - getBookingChannel, 329
 - getDemandGeneratorKey, 328

- getDestination, [328](#)
- getFrequentFlyerType, [329](#)
- getOrigin, [328](#)
- getPartySize, [329](#)
- getPOS, [328](#)
- getPreferredDepartureDate, [328](#)
- getPreferredCabin, [329](#)
- getPreferredDepartureTime, [328](#)
- getRequestDateTime, [329](#)
- getStayDuration, [329](#)
- getTripType, [329](#)
- getValueOfTime, [330](#)
- getWTP, [329](#)
- toStream, [330](#)
- stdair::Bucket, [332](#)
 - ~Bucket, [333](#)
 - _availability, [337](#)
 - _holderMap, [337](#)
 - _key, [336](#)
 - _parent, [336](#)
 - _soldSeats, [337](#)
 - _yieldRangeUpperValue, [337](#)
- boost::serialization::access, [336](#)
- Bucket, [333](#)
- describeKey, [336](#)
- FacBom, [336](#)
- FacBomManager, [336](#)
- fromStream, [335](#)
- getAvailability, [334](#)
- getHolderMap, [334](#)
- getKey, [334](#)
- getParent, [334](#)
- getSeatIndex, [334](#)
- getSoldSeats, [334](#)
- getYieldRangeUpperValue, [334](#)
- Key_T, [333](#)
- serialize, [336](#)
- setAvailability, [335](#)
- setSoldSeats, [335](#)
- setYieldRangeUpperValue, [334](#)
- toStream, [335](#)
- toString, [335](#)
- stdair::BucketKey, [337](#)
 - ~BucketKey, [338](#)
 - boost::serialization::access, [340](#)
 - BucketKey, [338](#)
 - fromStream, [339](#)
 - getSeatIndex, [339](#)
 - serialize, [339](#)
 - toStream, [339](#)
 - toString, [339](#)
- stdair::CancellationStruct, [341](#)
 - ~CancellationStruct, [342](#)
 - CancellationStruct, [342](#)
 - describe, [343](#)
 - display, [343](#)
 - fromStream, [343](#)
 - getCancellationDateTime, [342](#)
 - getClassList, [342](#)
 - getPartySize, [342](#)
 - getSegmentPath, [342](#)
 - toStream, [343](#)
- stdair::CmdAbstract, [344](#)
- stdair::CmdBomManager, [344](#)
 - STDAIR_Service, [344](#)
- stdair::CmdBomSerialiser, [345](#)
- stdair::CodeConversionException, [345](#)
 - _what, [346](#)
 - CodeConversionException, [346](#)
 - what, [346](#)
- stdair::CodeDuplicationException, [346](#)
 - _what, [347](#)
 - CodeDuplicationException, [347](#)
 - what, [347](#)
- stdair::ContinuousAttributeLite, [349](#)
 - ~ContinuousAttributeLite, [350](#)
 - ContinuousAttributeLite, [349, 350](#)
 - ContinuousDistribution_T, [349](#)
 - displayCumulativeDistribution, [351](#)
 - getDerivativeValue, [350](#)
 - getRemainingProportion, [350](#)
 - getUpperBound, [350](#)
 - getValue, [350](#)
 - operator=, [351](#)
- stdair::date_time_element, [351](#)
 - _value, [352](#)
 - check, [352](#)
 - date_time_element, [352](#)
- stdair::DatePeriod, [352](#)
 - ~DatePeriod, [354](#)
 - _holderMap, [356](#)
 - _key, [356](#)
 - _parent, [356](#)
 - DatePeriod, [354](#)
 - describeKey, [355](#)
 - FacBom, [356](#)
 - FacBomManager, [356](#)
 - fromStream, [354](#)
 - getDatePeriod, [355](#)
 - getHolderMap, [355](#)

- getKey, 355
- getParent, 355
- isDepartureDateValid, 356
- Key_T, 354
- toStream, 354
- toString, 355
- stdair::DatePeriodKey, 357
 - ~DatePeriodKey, 357
 - DatePeriodKey, 357
 - fromStream, 358
 - getDatePeriod, 358
 - toStream, 358
 - toString, 358
- stdair::DbAbstract, 359
 - ~DbAbstract, 359
 - DbAbstract, 359
 - fromStream, 359
 - toStream, 359
- stdair::DBManagerForAirlines, 360
 - iterateOnStatement, 361
 - prepareSelectStatement, 361
 - retrieveAirline, 361
 - updateAirlineInDB, 360
- stdair::DBSessionManager, 361
 - FacSupervisor, 362
 - getDBSession, 362
 - instance, 362
 - STDAIR_Service, 362
- stdair::DefaultDCPList, 362
 - init, 363
- stdair::DefaultDtdFratMap, 363
 - init, 363
- stdair::DefaultDtdProbMap, 363
 - init, 364
- stdair::DemandGenerationMethod, 364
 - DemandGenerationMethod, 365, 366
 - describe, 367
 - describeLabels, 366
 - EN_DemandGenerationMethod, 365
 - fromStream, 367
 - getLabel, 366
 - getMethod, 366, 367
 - getMethodAsChar, 367
 - getMethodAsString, 367
 - getMethodLabel, 366
 - getMethodLabelAsString, 366
 - LAST_VALUE, 365
 - operator==, 367
 - POI_PRO, 365
 - STA_ORD, 365
 - toStream, 367
- stdair::DictionaryManager, 368
 - keyToValue, 368
 - valueToKey, 369
- stdair::DocumentNotFoundException, 369
 - _what, 370
 - DocumentNotFoundException, 370
 - what, 370
- stdair::DoWStruct, 370
 - ~DoWStruct, 371
 - BooleanList_T, 371
 - describe, 372
 - describeShort, 372
 - DoWStruct, 371
 - fromStream, 373
 - getDayOfWeek, 372
 - getStandardDayOfWeek, 372
 - intersection, 373
 - isValid, 373
 - setDayOfWeek, 372
 - shift, 372
 - toStream, 373
- stdair::EventException, 374
 - _what, 375
 - EventException, 374
 - what, 374
- stdair::EventQueue, 375
 - ~EventQueue, 378
 - _eventList, 386
 - _holderMap, 386
 - _key, 386
 - _parent, 386
 - _progressStatus, 386
 - _progressStatusMap, 386
 - addEvent, 383
 - addStatus, 384
 - calculateProgress, 385
 - describeKey, 382
 - display, 382
 - EventQueue, 378
 - FacBom, 385
 - FacBomManager, 385
 - fromStream, 382
 - getActualTotalNbOfEvents, 379, 380
 - getCurrentNbOfEvents, 379, 380
 - getExpectedTotalNbOfEvents, 379, 380
 - getHolderMap, 379
 - getKey, 378
 - getParent, 379
 - getQueueSize, 385

- getStatus, 379
- isQueueDone, 383
- isQueueEmpty, 385
- Key_T, 378
- popEvent, 383
- ProgressStatusMap_T, 378
- reset, 382
- setActualTotalNbOfEvents, 381
- setCurrentNbOfEvents, 381
- setExpectedTotalNbOfEvents, 381
- setStatus, 380, 381
- toStream, 381
- toString, 382
- updateStatus, 384
- stdair::EventQueueException, 387
 - _what, 388
 - EventQueueException, 387
 - what, 388
- stdair::EventQueueKey, 388
 - ~EventQueueKey, 389
 - EventQueueKey, 389
 - fromStream, 389
 - getEventQueueID, 389
 - toStream, 389
 - toString, 390
- stdair::EventStruct, 390
 - ~EventStruct, 392
 - describe, 394
 - EventQueue, 394
 - EventStruct, 391, 392
 - fromStream, 393
 - getBookingRequest, 392
 - getCancellation, 393
 - getEventType, 392
 - getOptimisationNotificationStruct, 393
 - getRMEvent, 393
 - getSnapshotStruct, 393
 - toStream, 394
- stdair::EventType, 395
 - BKG_REQ, 396
 - BRK_PT, 396
 - CX, 396
 - describe, 397
 - describeLabels, 397
 - EN_EventType, 396
 - EventType, 396
 - fromStream, 398
 - getLabel, 396
 - getType, 397
 - getTypeAsString, 397
 - getTypeLabel, 396
 - getTypeLabelAsString, 397
 - LAST_VALUE, 396
 - operator==, 397
 - OPT_NOT_4_FD, 396
 - OPT_NOT_4_NET, 396
 - RM, 396
 - SKD_CHG, 396
 - SNAPSHOT, 396
 - toStream, 397
- stdair::FacAbstract, 398
 - ~FacAbstract, 399
 - FacAbstract, 399
- stdair::FacBom, 399
 - ~FacBom, 400
 - clean, 401
 - create, 400, 401
 - FacBom, 400
 - instance, 400
- stdair::FacBomManager, 401
 - ~FacBomManager, 402
 - addBomHolder, 403
 - addToList, 403, 406
 - addToListAndMap, 404
 - addToMap, 403, 404
 - cloneHolder, 405
 - FacBomManager, 402
 - getBomHolderPtr, 402
 - linkWithParent, 405
- stdair::FacServiceAbstract, 406
 - ~FacServiceAbstract, 407
 - _pool, 407
 - clean, 407
 - FacServiceAbstract, 407
 - ServicePool_T, 407
- stdair::FacSTDAIRServiceContext, 408
 - ~FacSTDAIRServiceContext, 409
 - _pool, 410
 - clean, 410
 - create, 409
 - FacSTDAIRServiceContext, 409
 - instance, 409
 - ServicePool_T, 409
- stdair::FacSupervisor, 410
 - ~FacSupervisor, 411
 - BomFactoryPool_T, 411
 - cleanAll, 413
 - cleanBomLayer, 413
 - cleanDBSessionManager, 413
 - cleanLoggerService, 413

- cleanServiceLayer, [413](#)
- FacSupervisor, [411](#), [412](#)
- instance, [412](#)
- registerBomFactory, [412](#)
- registerServiceFactory, [412](#)
- ServiceFactoryPool_T, [411](#)
- stdair::FareFamily, [413](#)
 - ~FareFamily, [415](#)
 - _holderMap, [417](#)
 - _key, [417](#)
 - _parent, [417](#)
 - boost::serialization::access, [417](#)
 - describeKey, [416](#)
 - FacBom, [417](#)
 - FacBomManager, [417](#)
 - FareFamily, [415](#)
 - fromStream, [416](#)
 - getFamilyCode, [415](#)
 - getHolderMap, [416](#)
 - getKey, [415](#)
 - getParent, [415](#)
 - Key_T, [415](#)
 - serialize, [417](#)
 - toStream, [416](#)
 - toString, [416](#)
- stdair::FareFamilyKey, [418](#)
 - ~FareFamilyKey, [419](#)
 - boost::serialization::access, [420](#)
 - FareFamilyKey, [418](#)
 - fromStream, [419](#)
 - getFamilyCode, [419](#)
 - serialize, [420](#)
 - toStream, [419](#)
 - toString, [419](#)
- stdair::FareFeatures, [420](#)
 - ~FareFeatures, [422](#)
 - _holderMap, [425](#)
 - _key, [425](#)
 - _parent, [425](#)
 - describeKey, [422](#)
 - FacBom, [425](#)
 - FacBomManager, [425](#)
 - FareFeatures, [422](#)
 - fromStream, [422](#)
 - getAdvancePurchase, [423](#)
 - getChangeFees, [424](#)
 - getHolderMap, [423](#)
 - getKey, [423](#)
 - getMinimumStay, [424](#)
 - getParent, [423](#)
 - getRefundableOption, [424](#)
 - getSaturdayStay, [423](#)
 - getTripType, [423](#)
 - isAdvancePurchaseValid, [424](#)
 - isStayDurationValid, [424](#)
 - isTripTypeValid, [424](#)
 - Key_T, [421](#)
 - toStream, [422](#)
 - toString, [422](#)
- stdair::FareFeaturesKey, [426](#)
 - ~FareFeaturesKey, [427](#)
 - FareFeaturesKey, [426](#)
 - fromStream, [428](#)
 - getAdvancePurchase, [427](#)
 - getChangeFees, [427](#)
 - getMinimumStay, [427](#)
 - getRefundableOption, [427](#)
 - getSaturdayStay, [427](#)
 - getTripType, [427](#)
 - toStream, [428](#)
 - toString, [428](#)
- stdair::FareOptionStruct, [428](#)
 - ~FareOptionStruct, [430](#)
 - addClassList, [431](#)
 - describe, [432](#)
 - display, [432](#)
 - emptyClassList, [431](#)
 - FareOptionStruct, [429](#), [430](#)
 - fromStream, [432](#)
 - getAvailability, [430](#)
 - getChangeFees, [430](#)
 - getClassPath, [430](#)
 - getFare, [430](#)
 - getNonRefundable, [430](#)
 - getSaturdayStay, [430](#)
 - setAvailability, [431](#)
 - setChangeFees, [431](#)
 - setFare, [431](#)
 - setNonRefundable, [431](#)
 - setSaturdayStay, [431](#)
 - toStream, [431](#)
- stdair::FileNotFoundException, [432](#)
 - _what, [433](#)
 - FileNotFoundException, [433](#)
 - what, [433](#)
- stdair::FlagSaver, [434](#)
 - ~FlagSaver, [434](#)
 - FlagSaver, [434](#)
- stdair::FlightDate, [434](#)
 - ~FlightDate, [436](#)

- [_holderMap](#), 440
 - [_key](#), 440
 - [_parent](#), 440
 - [boost::serialization::access](#), 440
 - [describeKey](#), 439
 - [FacBom](#), 440
 - [FacBomManager](#), 440
 - [FlightDate](#), 436
 - [fromStream](#), 439
 - [getAirlineCode](#), 437
 - [getDepartureDate](#), 437
 - [getFlightNumber](#), 436
 - [getHolderMap](#), 437
 - [getKey](#), 436
 - [getLegDate](#), 437, 438
 - [getParent](#), 436
 - [getSegmentDate](#), 438
 - [Key_T](#), 436
 - [serialize](#), 440
 - [toStream](#), 439
 - [toString](#), 439
- [stdair::FlightDateKey](#), 441
 - [~FlightDateKey](#), 442
 - [boost::serialization::access](#), 443
 - [FlightDateKey](#), 442
 - [fromStream](#), 442
 - [getDepartureDate](#), 442
 - [getFlightNumber](#), 442
 - [serialize](#), 443
 - [toStream](#), 442
 - [toString](#), 443
- [stdair::FlightPeriod](#), 443
 - [~FlightPeriod](#), 445
 - [_holderMap](#), 447
 - [_key](#), 447
 - [_parent](#), 447
 - [describeKey](#), 446
 - [FacBom](#), 447
 - [FacBomManager](#), 447
 - [FlightPeriod](#), 445
 - [fromStream](#), 446
 - [getFlightNumber](#), 445
 - [getHolderMap](#), 446
 - [getKey](#), 445
 - [getParent](#), 445
 - [getPeriod](#), 445
 - [Key_T](#), 445
 - [toStream](#), 446
 - [toString](#), 446
- [stdair::FlightPeriodKey](#), 447
 - [~FlightPeriodKey](#), 448
 - [FlightPeriodKey](#), 448
 - [fromStream](#), 449
 - [getFlightNumber](#), 448
 - [getPeriod](#), 448
 - [toStream](#), 449
 - [toString](#), 449
- [stdair::ForecastingMethod](#), 453
 - [ADD_PK](#), 454
 - [describe](#), 455
 - [describeLabels](#), 455
 - [EN_ForecastingMethod](#), 454
 - [ForecastingMethod](#), 454
 - [fromStream](#), 456
 - [getLabel](#), 455
 - [getMethod](#), 455
 - [getMethodAsString](#), 455
 - [getMethodLabel](#), 455
 - [getMethodLabelAsString](#), 455
 - [LAST_VALUE](#), 454
 - [MUL_PK](#), 454
 - [operator==](#), 456
 - [toStream](#), 456
- [stdair::GuillotineBlock](#), 457
 - [~GuillotineBlock](#), 459
 - [_availabilitySnapshotBlock](#), 467
 - [_bookingSnapshotBlock](#), 466
 - [_cancellationSnapshotBlock](#), 466
 - [_holderMap](#), 466
 - [_key](#), 465
 - [_parent](#), 465
 - [_productAndPriceOrientedBookingSnapshotBlock](#), 467
 - [_segmentCabinIndexMap](#), 466
 - [_valueTypesIndexMap](#), 466
 - [boost::serialization::access](#), 465
 - [describeKey](#), 465
 - [FacBom](#), 465
 - [FacBomManager](#), 465
 - [fromStream](#), 464
 - [getBlockIndex](#), 460
 - [getBlockNumber](#), 460
 - [getConstSegmentCabinDTDAvailabilitySnapshotView](#), 463
 - [getConstSegmentCabinDTDBookingSnapshotView](#), 461
 - [getConstSegmentCabinDTDCancellationSnapshotView](#), 461
 - [getConstSegmentCabinDTDProductAndPriceOrientedBookingSnapshotView](#), 461

- 462
- getConstSegmentCabinDTRangeAvailabilitySnapshotView, 463
- getConstSegmentCabinDTRangeBookingSnapshotView, 461
- getConstSegmentCabinDTRangeCancellationSnapshotView, 462
- getConstSegmentCabinDTRangeProductAndPriceOrientedBookingSnapshotView, 462
- getGuillotineNumber, 460
- getHolderMap, 460
- getKey, 459
- getParent, 459
- getSegmentCabinDTRangeAvailabilitySnapshotView, 463
- getSegmentCabinDTRangeBookingSnapshotView, 461
- getSegmentCabinDTRangeCancellationSnapshotView, 462
- getSegmentCabinDTRangeProductAndPriceOrientedBookingSnapshotView, 462
- getSegmentCabinDTRangeAvailabilitySnapshotView, 463
- getSegmentCabinDTRangeBookingSnapshotView, 461
- getSegmentCabinDTRangeCancellationSnapshotView, 462
- getSegmentCabinDTRangeProductAndPriceOrientedBookingSnapshotView, 463
- getSegmentCabinIndexMap, 460
- getValueTypeIndexMap, 460
- GuillotineBlock, 459
- initSnapshotBlocks, 464
- Key_T, 459
- serialize, 465
- toStream, 464
- toString, 464
- stdair::GuillotineBlockKey, 467
- ~GuillotineBlockKey, 468
- boost::serialization::access, 469
- fromStream, 469
- getGuillotineNumber, 468
- GuillotineBlockKey, 468
- serialize, 469
- toStream, 468
- toString, 469
- stdair::InputFilePath, 470
- _filename, 471
- InputFilePath, 470
- name, 470
- stdair::Inventory, 471
- ~Inventory, 472
- _airlineFeature, 476
- _holderMap, 476
- _key, 476
- _parent, 476
- boost::serialization::access, 475
- describeKey, 475
- FacBom, 475
- FacBomManager, 475
- fromStream, 474
- getAirlineCode, 473
- getFlightDate, 473, 474
- getHolderMap, 473
- getKey, 473
- getParent, 473
- Inventory, 472
- Key_T, 472
- serialize, 475
- setAirlineFeature, 474
- toStream, 474
- toString, 475
- stdair::InventoryKey, 476
- ~InventoryKey, 477
- boost::serialization::access, 479
- fromStream, 478
- getAirlineCode, 478
- InventoryKey, 477
- serialize, 478
- toStream, 478
- toString, 478
- stdair::KeyAbstract, 479
- ~KeyAbstract, 481
- fromStream, 481
- toStream, 481
- toString, 482
- stdair::KeyNotFoundException, 482
- _what, 483
- KeyNotFoundException, 483
- what, 483
- stdair::LegCabin, 484
- ~LegCabin, 486
- _acp, 499
- _au, 498
- _availability, 497
- _availabilityPool, 497
- _bidPriceVector, 497
- _committedSpace, 497

- [_currentBidPrice](#), 497
- [_dcsRegrade](#), 498
- [_etb](#), 499
- [_gav](#), 498
- [_groupNbOfBookings](#), 499
- [_holderMap](#), 496
- [_key](#), 496
- [_nav](#), 498
- [_offeredCapacity](#), 496
- [_parent](#), 496
- [_physicalCapacity](#), 496
- [_previousBidPrice](#), 497
- [_soldSeat](#), 497
- [_staffNbOfBookings](#), 499
- [_upr](#), 498
- [_virtualClassList](#), 498
- [_wlnNbOfBookings](#), 499
- [_yieldLevelDemandMap](#), 498
- [addDemandInformation](#), 495
- [addVirtualClass](#), 495
- [describeKey](#), 494
- [displayVirtualClassList](#), 495
- [emptyBidPriceVector](#), 495
- [emptyVirtualClassList](#), 495
- [emptyYieldLevelDemandMap](#), 496
- [FacBom](#), 496
- [FacBomManager](#), 496
- [fromStream](#), 494
- [getAuthorizationLevel](#), 489
- [getAvailability](#), 488
- [getAvailabilityPool](#), 488
- [getAvgCancellationPercentage](#), 490
- [getBidPriceVector](#), 489, 490
- [getCabinCode](#), 487
- [getCommittedSpace](#), 488
- [getCurrentBidPrice](#), 488
- [getETB](#), 490
- [getFullerKey](#), 487
- [getGrossAvailability](#), 489
- [getGroupNbOfSeats](#), 490
- [getHolderMap](#), 487
- [getKey](#), 487
- [getNetAvailability](#), 489
- [getOfferedCapacity](#), 487
- [getParent](#), 487
- [getPhysicalCapacity](#), 488
- [getPreviousBidPrice](#), 488
- [getRegradeAdjustment](#), 489
- [getSoldSeat](#), 488
- [getStaffNbOfSeats](#), 490
- [getUPR](#), 489
- [getVirtualClassList](#), 490
- [getWlnNbOfSeats](#), 490
- [getYieldLevelDemandMap](#), 491
- [Key_T](#), 486
- [LegCabin](#), 486
- [setAuthorizationLevel](#), 492
- [setAvailability](#), 491
- [setAvailabilityPool](#), 491
- [setAvgCancellationPercentage](#), 493
- [setCapacities](#), 491
- [setCommittedSpace](#), 491
- [setCurrentBidPrice](#), 492
- [setETB](#), 493
- [setGrossAvailability](#), 493
- [setGroupNbOfSeats](#), 493
- [setNetAvailability](#), 493
- [setPreviousBidPrice](#), 492
- [setRegradeAdjustment](#), 492
- [setSoldSeat](#), 491
- [setStaffNbOfSeats](#), 493
- [setUPR](#), 492
- [setWlnNbOfSeats](#), 493
- [toStream](#), 494
- [toString](#), 494
- [updateCurrentBidPrice](#), 494
- [updateFromReservation](#), 495
- [updatePreviousBidPrice](#), 492
- [stdair::LegCabinKey](#), 499
 - [~LegCabinKey](#), 500
 - [boost::serialization::access](#), 502
 - [fromStream](#), 501
 - [getCabinCode](#), 501
 - [LegCabinKey](#), 500
 - [serialize](#), 501
 - [toStream](#), 501
 - [toString](#), 501
- [stdair::LegDate](#), 502
 - [~LegDate](#), 504
 - [_boardingDate](#), 510
 - [_boardingTime](#), 510
 - [_capacity](#), 511
 - [_distance](#), 511
 - [_elapsedTime](#), 510
 - [_holderMap](#), 510
 - [_key](#), 509
 - [_offDate](#), 510
 - [_offPoint](#), 510
 - [_offTime](#), 510
 - [_parent](#), 509

- describeKey, [509](#)
- FacBom, [509](#)
- FacBomManager, [509](#)
- fromStream, [508](#)
- getAirlineCode, [504](#)
- getBoardingDate, [506](#)
- getBoardingPoint, [504](#)
- getBoardingTime, [506](#)
- getCapacity, [507](#)
- getDateOffset, [507](#)
- getDistance, [507](#)
- getElapsedTime, [506](#)
- getHolderMap, [505](#)
- getKey, [504](#)
- getLegCabin, [505](#)
- getOffDate, [506](#)
- getOffPoint, [506](#)
- getOffTime, [506](#)
- getParent, [504](#)
- getTimeOffset, [507](#)
- Key_T, [504](#)
- LegDate, [504](#)
- setBoardingDate, [507](#)
- setBoardingTime, [508](#)
- setElapsedTime, [508](#)
- setOffDate, [508](#)
- setOffPoint, [507](#)
- setOffTime, [508](#)
- toStream, [508](#)
- toString, [509](#)
- stdair::LegDateKey, [511](#)
- ~LegDateKey, [512](#)
- fromStream, [512](#)
- getBoardingPoint, [512](#)
- LegDateKey, [512](#)
- toStream, [512](#)
- toString, [513](#)
- stdair::LOG, [235](#)
- _logLevels, [236](#)
- CRITICAL, [236](#)
- DEBUG, [236](#)
- EN_LogLevel, [236](#)
- ERROR, [236](#)
- LAST_VALUE, [236](#)
- NOTIFICATION, [236](#)
- VERBOSE, [236](#)
- WARNING, [236](#)
- stdair::Logger, [513](#)
- FacSupervisor, [514](#)
- instance, [514](#)
- log, [514](#)
- STDAIR_Service, [514](#)
- stdair::MemoryAllocationException, [514](#)
- _what, [515](#)
- MemoryAllocationException, [515](#)
- what, [515](#)
- stdair::NonInitialisedContainerException, [515](#)
- _what, [516](#)
- NonInitialisedContainerException, [516](#)
- what, [516](#)
- stdair::NonInitialisedDBSessionManagerException, [517](#)
- _what, [518](#)
- NonInitialisedDBSessionManagerException, [517](#)
- what, [517](#)
- stdair::NonInitialisedLogServiceException, [518](#)
- _what, [519](#)
- NonInitialisedLogServiceException, [519](#)
- what, [519](#)
- stdair::NonInitialisedRelationShipException, [519](#)
- _what, [520](#)
- NonInitialisedRelationShipException, [520](#)
- what, [520](#)
- stdair::NonInitialisedServiceException, [520](#)
- _what, [521](#)
- NonInitialisedServiceException, [521](#)
- what, [521](#)
- stdair::ObjectCreationDuplicationException, [522](#)
- _what, [523](#)
- ObjectCreationDuplicationException, [522](#)
- what, [523](#)
- stdair::ObjectLinkingException, [523](#)
- _what, [524](#)
- ObjectLinkingException, [524](#)
- what, [524](#)
- stdair::ObjectNotFoundException, [524](#)
- _what, [525](#)
- ObjectNotFoundException, [525](#)
- what, [525](#)
- stdair::OnDDate, [526](#)
- ~OnDDate, [527](#)
- _cabinForecastMap, [532](#)
- _classPathDemandMap, [531](#)
- _holderMap, [531](#)
- _key, [531](#)

- [_parent](#), [531](#)
 - [_stringCabinClassPairListMap](#), [532](#)
 - [boost::serialization::access](#), [531](#)
 - [describeKey](#), [530](#)
 - [FacBom](#), [531](#)
 - [FacBomManager](#), [531](#)
 - [fromStream](#), [530](#)
 - [getAirlineCode](#), [528](#)
 - [getCabinClassPairList](#), [529](#)
 - [getDate](#), [528](#)
 - [getDemandInfoMap](#), [529](#)
 - [getDestination](#), [528](#)
 - [getHolderMap](#), [528](#)
 - [getKey](#), [528](#)
 - [getNbOfSegments](#), [529](#)
 - [getOrigin](#), [528](#)
 - [getParent](#), [528](#)
 - [getTotalForecast](#), [529](#)
 - [getTotalForecastMap](#), [529](#)
 - [Key_T](#), [527](#)
 - [OnDDate](#), [527](#)
 - [serialize](#), [531](#)
 - [setDemandInformation](#), [529](#)
 - [setTotalForecast](#), [530](#)
 - [toStream](#), [530](#)
 - [toString](#), [530](#)
- [stdair::OnDDateKey](#), [532](#)
 - [~OnDDateKey](#), [533](#)
 - [boost::serialization::access](#), [535](#)
 - [fromStream](#), [534](#)
 - [getDate](#), [533](#)
 - [getDestination](#), [534](#)
 - [getNbOfSegments](#), [534](#)
 - [getOrigin](#), [533](#)
 - [OnDDateKey](#), [533](#)
 - [serialize](#), [535](#)
 - [toStream](#), [534](#)
 - [toString](#), [534](#)
- [stdair::OptimisationNotificationStruct](#), [535](#)
 - [~OptimisationNotificationStruct](#), [536](#)
 - [describe](#), [539](#)
 - [fromStream](#), [539](#)
 - [getDestination](#), [537](#)
 - [getFrequentFlyerType](#), [538](#)
 - [getNotificationDateTime](#), [537](#)
 - [getOptimisationChannel](#), [538](#)
 - [getOrigin](#), [537](#)
 - [getPartySize](#), [537](#)
 - [getPOS](#), [537](#)
 - [getPreferredDepartureDate](#), [537](#)
 - [getPreferredCabin](#), [537](#)
 - [getPreferredDepartureTime](#), [538](#)
 - [getStayDuration](#), [538](#)
 - [getTripType](#), [538](#)
 - [getValueOfTime](#), [538](#)
 - [getWTP](#), [538](#)
 - [OptimisationNotificationStruct](#), [536](#)
 - [toStream](#), [538](#)
- [stdair::ParsedKey](#), [539](#)
 - [~ParsedKey](#), [540](#)
 - [_airlineCode](#), [542](#)
 - [_boardingPoint](#), [542](#)
 - [_boardingTime](#), [543](#)
 - [_departureDate](#), [542](#)
 - [_flightNumber](#), [542](#)
 - [_fullKey](#), [542](#)
 - [_offPoint](#), [542](#)
 - [fromStream](#), [541](#)
 - [getBoardingTime](#), [541](#)
 - [getFlightDateKey](#), [540](#)
 - [getInventoryKey](#), [540](#)
 - [getSegmentKey](#), [541](#)
 - [ParsedKey](#), [540](#)
 - [toStream](#), [541](#)
 - [toString](#), [541](#)
- [stdair::ParserException](#), [543](#)
 - [_what](#), [544](#)
 - [ParserException](#), [543](#)
 - [what](#), [544](#)
- [stdair::ParsingFileFailedException](#), [544](#)
 - [_what](#), [545](#)
 - [ParsingFileFailedException](#), [545](#)
 - [what](#), [545](#)
- [stdair::PartnershipTechnique](#), [545](#)
 - [A_RMC](#), [547](#)
 - [describe](#), [549](#)
 - [describeLabels](#), [548](#)
 - [EN_PartnershipTechnique](#), [547](#)
 - [fromStream](#), [549](#)
 - [getLabel](#), [548](#)
 - [getTechnique](#), [548](#)
 - [getTechniqueAsChar](#), [548](#)
 - [getTechniqueAsString](#), [549](#)
 - [getTechniqueLabel](#), [548](#)
 - [getTechniqueLabelAsString](#), [548](#)
 - [IBP_DA](#), [547](#)
 - [IBP_YP](#), [547](#)
 - [IBP_YP_U](#), [547](#)
 - [LAST_VALUE](#), [547](#)
 - [NONE](#), [547](#)

- operator==, [549](#)
- PartnershipTechnique, [547](#)
- RAE_DA, [547](#)
- RAE_YP, [547](#)
- RMC, [547](#)
- toStream, [549](#)
- stdair::PassengerType, [550](#)
 - BUSINESS, [551](#)
 - describe, [552](#)
 - describeLabels, [552](#)
 - EN_PassengerType, [551](#)
 - FIRST, [551](#)
 - fromStream, [553](#)
 - getLabel, [551](#)
 - getType, [552](#)
 - getTypeAsString, [552](#)
 - getTypeLabel, [551](#)
 - getTypeLabelAsString, [551](#)
 - LAST_VALUE, [551](#)
 - LEISURE, [551](#)
 - operator==, [552](#)
 - PassengerType, [551](#)
 - toStream, [552](#)
- stdair::PeriodStruct, [553](#)
 - ~PeriodStruct, [554](#)
 - addDateOffset, [555](#)
 - describe, [555](#)
 - describeShort, [555](#)
 - fromStream, [556](#)
 - getDateRange, [554](#)
 - getDoW, [555](#)
 - intersection, [555](#)
 - isValid, [555](#)
 - PeriodStruct, [554](#)
 - setDateRange, [555](#)
 - setDoW, [555](#)
 - toStream, [556](#)
- stdair::PosChannel, [556](#)
 - ~PosChannel, [558](#)
 - _holderMap, [560](#)
 - _key, [560](#)
 - _parent, [560](#)
 - describeKey, [559](#)
 - FacBom, [560](#)
 - FacBomManager, [560](#)
 - fromStream, [558](#)
 - getChannel, [559](#)
 - getHolderMap, [559](#)
 - getKey, [559](#)
 - getParent, [559](#)
 - getPos, [559](#)
 - Key_T, [558](#)
 - PosChannel, [558](#)
 - toStream, [558](#)
 - toString, [558](#)
- stdair::PosChannelKey, [560](#)
 - ~PosChannelKey, [561](#)
 - fromStream, [562](#)
 - getChannel, [562](#)
 - getPos, [561](#)
 - PosChannelKey, [561](#)
 - toStream, [562](#)
 - toString, [562](#)
- stdair::ProgressStatus, [563](#)
 - count, [565](#)
 - describe, [566](#)
 - fromStream, [567](#)
 - getActualNb, [565](#)
 - getCurrentNb, [565](#)
 - getExpectedNb, [565](#)
 - operator++, [566](#)
 - operator+=, [566](#)
 - progress, [565](#)
 - ProgressStatus, [564](#)
 - reset, [566](#)
 - setActualNb, [566](#)
 - setCurrentNb, [565](#)
 - setExpectedNb, [566](#)
 - toStream, [566](#)
- stdair::ProgressStatusSet, [567](#)
 - ~ProgressStatusSet, [568](#)
 - describe, [570](#)
 - fromStream, [569](#)
 - getOverallStatus, [569](#)
 - getSpecificGeneratorStatus, [569](#)
 - getTypeSpecificStatus, [568](#)
 - ProgressStatusSet, [568](#)
 - setOverallStatus, [569](#)
 - setSpecificGeneratorStatus, [569](#)
 - setTypeSpecificStatus, [569](#)
 - toStream, [570](#)
- stdair::RandomGeneration, [570](#)
 - ~RandomGeneration, [572](#)
 - _generator, [574](#)
 - describe, [573](#)
 - fromStream, [574](#)
 - generateExponential, [573](#)
 - generateNormal, [572](#)
 - generateUniform, [572](#)
 - generateUniform01, [572](#)

- getBaseGenerator, 573
- init, 573
- operator(), 572
- RandomGeneration, 571
- toStream, 573
- stdair::RMEventStruct, 574
 - ~RMEventStruct, 575
 - describe, 576
 - fromStream, 576
 - getAirlineCode, 576
 - getFlightDateDescription, 576
 - getRMEventTime, 576
 - RMEventStruct, 575
 - toStream, 576
- stdair::RootException, 577
 - ~RootException, 578
 - _what, 578
 - RootException, 578
 - what, 578
- stdair::RootFilePath, 579
 - ~RootFilePath, 580
 - _filename, 580
 - name, 580
 - RootFilePath, 579, 580
- stdair::SampleType, 580
 - A4P, 582
 - ALL, 582
 - CCM, 582
 - CRS, 582
 - DEM, 582
 - describe, 583
 - describeLabels, 583
 - EN_SampleType, 582
 - EVT, 582
 - FQT, 582
 - fromStream, 584
 - getLabel, 582
 - getType, 583
 - getTypeAsString, 583
 - getTypeLabel, 583
 - getTypeLabelAsString, 583
 - INV, 582
 - LAST_VALUE, 582
 - operator==, 583
 - RAC, 582
 - RMS, 582
 - SampleType, 582
 - SCH, 582
 - toStream, 583
- stdair::SegmentCabin, 584
 - ~SegmentCabin, 587
 - _availabilityPool, 594
 - _bidPriceVector, 594
 - _blockSpace, 593
 - _bookingCounter, 593
 - _capacity, 593
 - _committedSpace, 594
 - _currentBidPrice, 594
 - _fareFamilyActivation, 594
 - _guillotineBlock, 593
 - _holderMap, 593
 - _key, 592
 - _min, 593
 - _parent, 592
 - _upr, 593
 - activateFareFamily, 591
 - boost::serialization::access, 592
 - describeKey, 592
 - FacBom, 592
 - FacBomManager, 592
 - fromStream, 591
 - getAvailabilityPool, 589
 - getBidPriceVector, 589
 - getBlockSpace, 588
 - getBookingCounter, 588
 - getCabinCode, 587
 - getCapacity, 588
 - getCommittedSpace, 589
 - getCurrentBidPrice, 589
 - getFareFamilyStatus, 589
 - getFullerKey, 587
 - getGuillotineBlock, 588
 - getHolderMap, 587
 - getKey, 587
 - getMIN, 588
 - getParent, 587
 - getUPR, 588
 - Key_T, 586
 - SegmentCabin, 587
 - serialize, 592
 - setAvailabilityPool, 590
 - setBidPriceVector, 591
 - setBlockSpace, 590
 - setBookingCounter, 590
 - setCapacity, 589
 - setCommittedSpace, 590
 - setGuillotineBlock, 589
 - setMIN, 590
 - setUPR, 590
 - toStream, 591

- toString, 591
- updateFromReservation, 591
- stdair::SegmentCabinKey, 595
 - ~SegmentCabinKey, 596
 - boost::serialization::access, 597
 - fromStream, 596
 - getCabinCode, 596
 - SegmentCabinKey, 595
 - serialize, 597
 - toStream, 596
 - toString, 596
- stdair::SegmentDate, 597
 - ~SegmentDate, 599
 - _boardingDate, 605
 - _boardingTime, 605
 - _distance, 606
 - _elapsedTime, 605
 - _holderMap, 604
 - _key, 604
 - _marketingSegmentDateList, 605
 - _offDate, 605
 - _offTime, 605
 - _operatingSegmentDate, 604
 - _parent, 604
 - boost::serialization::access, 604
 - describeKey, 603
 - FacBom, 604
 - FacBomManager, 604
 - fromStream, 603
 - getBoardingDate, 600
 - getBoardingPoint, 599
 - getBoardingTime, 600
 - getDateOffset, 601
 - getDistance, 601
 - getElapsedTime, 600
 - getHolderMap, 600
 - getKey, 599
 - getMarketingSegmentDateList, 601
 - getOffDate, 600
 - getOffPoint, 600
 - getOffTime, 600
 - getOperatingSegmentDate, 601
 - getParent, 599
 - getTimeOffset, 601
 - Key_T, 599
 - linkWithOperating, 602
 - SegmentDate, 599
 - serialize, 603
 - setBoardingDate, 601
 - setBoardingTime, 602
 - setDistance, 602
 - setElapsedTime, 602
 - setOffDate, 602
 - setOffTime, 602
 - toStream, 603
 - toString, 603
- stdair::SegmentDateKey, 606
 - ~SegmentDateKey, 607
 - boost::serialization::access, 608
 - fromStream, 608
 - getBoardingPoint, 607
 - getOffPoint, 607
 - SegmentDateKey, 607
 - serialize, 608
 - toStream, 607
 - toString, 608
- stdair::SegmentPeriod, 609
 - ~SegmentPeriod, 611
 - _boardingDateOffset, 615
 - _boardingTime, 615
 - _cabinBookingClassMap, 615
 - _elapsedTime, 615
 - _holderMap, 615
 - _key, 614
 - _offDateOffset, 615
 - _offTime, 615
 - _parent, 615
 - addCabinBookingClassList, 613
 - describeKey, 614
 - FacBom, 614
 - FacBomManager, 614
 - fromStream, 614
 - getBoardingDateOffset, 612
 - getBoardingPoint, 611
 - getBoardingTime, 611
 - getCabinBookingClassMap, 612
 - getElapsedTime, 612
 - getHolderMap, 612
 - getKey, 611
 - getOffDateOffset, 612
 - getOffPoint, 611
 - getOffTime, 611
 - getParent, 611
 - Key_T, 610
 - SegmentPeriod, 610
 - setBoardingDateOffset, 613
 - setBoardingTime, 612
 - setElapsedTime, 613
 - setOffDateOffset, 613
 - setOffTime, 613

- toStream, [613](#)
- toString, [614](#)
- stdair::SegmentPeriodKey, [616](#)
 - ~SegmentPeriodKey, [616](#)
 - fromStream, [617](#)
 - getBoardingPoint, [617](#)
 - getOffPoint, [617](#)
 - SegmentPeriodKey, [616](#)
 - toStream, [617](#)
 - toString, [617](#)
- stdair::SerialisationException, [618](#)
 - _what, [619](#)
 - SerialisationException, [619](#)
 - what, [619](#)
- stdair::ServiceAbstract, [619](#)
 - ~ServiceAbstract, [620](#)
 - fromStream, [620](#)
 - ServiceAbstract, [620](#)
 - toStream, [620](#)
- stdair::ServiceInitialisationType, [621](#)
 - BUILTIN_SAMPLE, [622](#)
 - describe, [624](#)
 - describeLabels, [623](#)
 - EN_ServiceInitialisationType, [622](#)
 - FILE_PARSING, [622](#)
 - fromStream, [624](#)
 - getLabel, [622](#)
 - getType, [623](#)
 - getTypeAsChar, [623](#)
 - getTypeAsString, [623](#)
 - getTypeLabel, [623](#)
 - getTypeLabelAsString, [623](#)
 - LAST_VALUE, [622](#)
 - NOT_YET_INITIALISED, [622](#)
 - operator==, [624](#)
 - ServiceInitialisationType, [622](#)
 - toStream, [624](#)
- stdair::SnapshotStruct, [627](#)
 - ~SnapshotStruct, [628](#)
 - describe, [629](#)
 - fromStream, [629](#)
 - getAirlineCode, [628](#)
 - getSnapshotTime, [628](#)
 - SnapshotStruct, [628](#)
 - toStream, [628](#)
- stdair::SQLDatabaseConnectionImpossibleException, [629](#)
 - _what, [630](#)
 - SQLDatabaseConnectionImpossibleException, [630](#)
- what, [630](#)
- stdair::SQLDatabaseException, [631](#)
 - _what, [632](#)
 - SQLDatabaseException, [631](#)
 - what, [631](#)
- stdair::STDAIR_Service, [637](#)
 - ~STDAIR_Service, [639](#)
 - buildDummyInventory, [640](#)
 - buildSampleBom, [639](#)
 - buildSampleBookingRequest, [641](#)
 - buildSampleTravelSolutionForPricing, [640](#)
 - buildSampleTravelSolutions, [641](#)
 - check, [646](#)
 - csvDisplay, [646](#), [647](#)
 - getActualTotalNumberOfEventsToBeGenerated, [643](#)
 - getBomRoot, [648](#)
 - getDBParams, [648](#)
 - getEventQueue, [648](#)
 - getExpectedTotalNumberOfEventsToBeGenerated, [642](#), [643](#)
 - getLogParams, [648](#)
 - getServiceInitialisationType, [648](#)
 - isQueueDone, [644](#)
 - jsonExport, [645](#)
 - list, [645](#)
 - listAirportPairDateRange, [645](#)
 - popEvent, [644](#)
 - reset, [644](#)
 - STDAIR_Service, [639](#)
- stdair::STDAIR_ServiceContext, [649](#)
 - FacSTDAIRServiceContext, [650](#)
 - fromStream, [650](#)
 - STDAIR_Service, [650](#)
 - toStream, [649](#)
- stdair::StructAbstract, [650](#)
 - ~StructAbstract, [652](#)
 - describe, [652](#)
 - fromStream, [652](#)
 - StructAbstract, [652](#)
 - toStream, [652](#)
- stdair::TimePeriod, [653](#)
 - ~TimePeriod, [654](#)
 - _holderMap, [657](#)
 - _key, [657](#)
 - _parent, [657](#)
 - describeKey, [655](#)
 - FacBom, [656](#)
 - FacBomManager, [656](#)

- fromStream, 655
- getHolderMap, 656
- getKey, 655
- getParent, 655
- getTimeRangeEnd, 656
- getTimeRangeStart, 656
- isDepartureTimeValid, 656
- Key_T, 654
- TimePeriod, 654
- toStream, 654
- toString, 655
- stdair::TimePeriodKey, 657
 - ~TimePeriodKey, 658
 - fromStream, 659
 - getTimeRangeEnd, 658
 - getTimeRangeStart, 658
 - TimePeriodKey, 658
 - toStream, 658
 - toString, 659
- stdair::TravelSolutionStruct, 659
 - ~TravelSolutionStruct, 660
 - addBidPriceVector, 662
 - addClassAvailabilityMap, 662
 - addClassBpvMap, 662
 - addClassYieldMap, 662
 - addFareOption, 662
 - addSegment, 662
 - describe, 663
 - display, 663
 - fromStream, 663
 - getBidPriceVectorHolder, 661
 - getChosenFareOption, 662
 - getClassAvailabilityMapHolder, 661
 - getClassBpvMapHolder, 661
 - getClassYieldMapHolder, 661
 - getFareOptionList, 661
 - getFareOptionListRef, 661
 - getSegmentPath, 661
 - setChosenFareOption, 662
 - toStream, 663
 - TravelSolutionStruct, 660
- stdair::VirtualClassStruct, 667
 - ~VirtualClassStruct, 668
 - describe, 670
 - fromStream, 669
 - getCumulatedBookingLimit, 668
 - getCumulatedProtection, 668
 - getGeneratedDemandVector, 668
 - getMean, 668
 - getStdDev, 668
 - getYield, 668
 - setCumulatedBookingLimit, 669
 - setCumulatedProtection, 669
 - setMean, 669
 - setStdDev, 669
 - setYield, 669
 - toStream, 669
 - VirtualClassStruct, 667
- stdair::YieldFeatures, 670
 - ~YieldFeatures, 671
 - _holderMap, 674
 - _key, 674
 - _parent, 674
 - describeKey, 672
 - FacBom, 673
 - FacBomManager, 674
 - fromStream, 672
 - getCabinCode, 673
 - getHolderMap, 673
 - getKey, 673
 - getParent, 673
 - getTripType, 673
 - isTripTypeValid, 673
 - Key_T, 671
 - toStream, 672
 - toString, 672
 - YieldFeatures, 671
- stdair::YieldFeaturesKey, 674
 - ~YieldFeaturesKey, 675
 - fromStream, 676
 - getCabinCode, 675
 - getTripType, 675
 - toStream, 676
 - toString, 676
 - YieldFeaturesKey, 675
- stdair::YieldRange, 676
 - ~YieldRange, 678
 - describe, 679
 - fromStream, 679
 - getAverageYield, 678
 - getLowerYield, 678
 - getUpperYield, 678
 - setAverageYield, 678
 - setLowerYield, 679
 - setUpperYield, 678
 - toStream, 679
 - YieldRange, 677, 678
- stdair::YieldStore, 679
 - ~YieldStore, 681
 - _key, 682

- [_parent](#), [683](#)
 - [describeKey](#), [682](#)
 - [FacBom](#), [682](#)
 - [FacBomManager](#), [682](#)
 - [fromStream](#), [681](#)
 - [getAirlineCode](#), [682](#)
 - [getKey](#), [682](#)
 - [getParent](#), [681](#)
 - [Key_T](#), [681](#)
 - [toStream](#), [681](#)
 - [toString](#), [682](#)
 - [YieldStore](#), [681](#)
- [stdair::YieldStoreKey](#), [683](#)
- [~YieldStoreKey](#), [684](#)
 - [fromStream](#), [684](#)
 - [getAirlineCode](#), [684](#)
 - [toStream](#), [684](#)
 - [toString](#), [684](#)
 - [YieldStoreKey](#), [684](#)
- [STDAIR_LOG_CORE](#)
- [Logger.hpp](#), [1197](#)
- [STDAIR_LOG_CRITICAL](#)
- [Logger.hpp](#), [1197](#)
- [STDAIR_LOG_DEBUG](#)
- [Logger.hpp](#), [1198](#)
- [STDAIR_LOG_ERROR](#)
- [Logger.hpp](#), [1198](#)
- [STDAIR_LOG_NOTIFICATION](#)
- [Logger.hpp](#), [1198](#)
- [STDAIR_LOG_VERBOSE](#)
- [Logger.hpp](#), [1198](#)
- [STDAIR_LOG_WARNING](#)
- [Logger.hpp](#), [1198](#)
- [STDAIR_SAMPLE_DIR](#)
- [stdair-paths.hpp](#), [1170](#)
- [STDAIR_Service](#)
- [stdair::CmdBomManager](#), [344](#)
 - [stdair::DBSessionManager](#), [362](#)
 - [stdair::Logger](#), [514](#)
 - [stdair::STDAIR_Service](#), [639](#)
 - [stdair::STDAIR_ServiceContext](#), [650](#)
- [STDAIR_ServicePtr_T](#)
- [stdair](#), [208](#)
- [stdair_test](#), [236](#)
- [stdair_test::BookingClass](#), [309](#)
- [_classCode](#), [310](#)
 - [BookingClass](#), [310](#)
 - [toString](#), [310](#)
- [stdair_test::Cabin](#), [340](#)
- [_bookingClass](#), [341](#)
- [Cabin](#), [341](#)
 - [child](#), [340](#)
 - [toString](#), [341](#)
- [StdDevValue_T](#)
- [stdair](#), [206](#)
- [StringCabinClassPair_T](#)
- [stdair](#), [186](#)
- [StringCabinClassPairListMap_T](#)
- [stdair](#), [185](#)
- [StringDemandStructMap_T](#)
- [stdair](#), [185](#)
- [StringDemandStructPair_T](#)
- [stdair](#), [185](#)
- [stripwhite](#)
- [readline_autocomp.hpp](#), [1244](#)
- [StructAbstract](#)
- [stdair::StructAbstract](#), [652](#)
- [StructAbstract.hpp](#)
- [operator<<](#), [779](#)
 - [operator>>](#), [779](#)
- [SubclassCode_T](#)
- [stdair](#), [200](#)
- [swift](#), [237](#)
- [swift::SKeymap](#), [625](#)
- [~SKeymap](#), [626](#)
 - [Bind](#), [626](#)
 - [operator=](#), [627](#)
 - [SKeymap](#), [626](#)
 - [SReadline](#), [627](#)
 - [Unbind](#), [627](#)
- [swift::SReadline](#), [632](#)
- [~SReadline](#), [633](#)
 - [ClearHistory](#), [636](#)
 - [GetHistory](#), [635](#)
 - [GetLine](#), [634](#), [635](#)
 - [LoadHistory](#), [636](#)
 - [RegisterCompletions](#), [636](#)
 - [SaveHistory](#), [635](#)
 - [SetKeymap](#), [637](#)
 - [SReadline](#), [633](#)
- [syscom](#)
- [readline_autocomp.hpp](#), [1246](#)
- [SYSCONFDIR](#)
- [stdair-paths.hpp](#), [1169](#)
- [test/ Directory Reference](#), [157](#)
- [test/stdair/ Directory Reference](#), [156](#)
- [test/stdair/MPBomRoot.cpp](#), [1260](#)
- [test/stdair/MPBomRoot.hpp](#), [1261](#)
- [test/stdair/MPInventory.cpp](#), [1261](#), [1262](#)

- test/stdair/MPInventory.hpp, 1262
- test/stdair/StandardAirlineITTestSuite.cpp, 1263
- test/stdair/StdairTestLib.hpp, 1268
- Time_T
 - stdair, 194
- TimePeriod
 - stdair::TimePeriod, 654
- TimePeriodDetailedList_T
 - stdair, 188
- TimePeriodKey
 - stdair::TimePeriodKey, 658
- TimePeriodList_T
 - stdair, 188
- TimePeriodMap_T
 - stdair, 188
- TimePeriodWithKey_T
 - stdair, 188
- to_base
 - soci::type_conversion< stdair::AirlineStruct
>, 664
- Tokeniser_T
 - stdair, 181
- TokeniserDashSeparator
 - stdair, 221
- TokeniserTimeSeparator
 - stdair, 221
- too_dangerous
 - readline_autocomp.hpp, 1246
- toShortString
 - stdair::BasDBParams, 263
 - stdair::BasLogParams, 267
- toStream
 - stdair::AirlineClassList, 240
 - stdair::AirlineClassListKey, 244
 - stdair::AirlineFeature, 247
 - stdair::AirlineFeatureKey, 250
 - stdair::AirlineStruct, 252
 - stdair::AirportPair, 255
 - stdair::AirportPairKey, 259
 - stdair::BasDBParams, 263
 - stdair::BasLogParams, 267
 - stdair::BomAbstract, 270
 - stdair::BomHolder, 282
 - stdair::BomHolderKey, 285
 - stdair::BomRoot, 305
 - stdair::BomRootKey, 308
 - stdair::BookingClass, 318
 - stdair::BookingClassKey, 325
 - stdair::BookingRequestStruct, 330
 - stdair::Bucket, 335
 - stdair::BucketKey, 339
 - stdair::CancellationStruct, 343
 - stdair::DatePeriod, 354
 - stdair::DatePeriodKey, 358
 - stdair::DbAbstract, 359
 - stdair::DemandGenerationMethod, 367
 - stdair::DoWStruct, 373
 - stdair::EventQueue, 381
 - stdair::EventQueueKey, 389
 - stdair::EventStruct, 394
 - stdair::EventType, 397
 - stdair::FareFamily, 416
 - stdair::FareFamilyKey, 419
 - stdair::FareFeatures, 422
 - stdair::FareFeaturesKey, 428
 - stdair::FareOptionStruct, 431
 - stdair::FlightDate, 439
 - stdair::FlightDateKey, 442
 - stdair::FlightPeriod, 446
 - stdair::FlightPeriodKey, 449
 - stdair::ForecastingMethod, 456
 - stdair::GuillotineBlock, 464
 - stdair::GuillotineBlockKey, 468
 - stdair::Inventory, 474
 - stdair::InventoryKey, 478
 - stdair::KeyAbstract, 481
 - stdair::LegCabin, 494
 - stdair::LegCabinKey, 501
 - stdair::LegDate, 508
 - stdair::LegDateKey, 512
 - stdair::OnDDate, 530
 - stdair::OnDDateKey, 534
 - stdair::OptimisationNotificationStruct,
538
 - stdair::ParsedKey, 541
 - stdair::PartnershipTechnique, 549
 - stdair::PassengerType, 552
 - stdair::PeriodStruct, 556
 - stdair::PosChannel, 558
 - stdair::PosChannelKey, 562
 - stdair::ProgressStatus, 566
 - stdair::ProgressStatusSet, 570
 - stdair::RandomGeneration, 573
 - stdair::RMEventStruct, 576
 - stdair::SampleType, 583
 - stdair::SegmentCabin, 591
 - stdair::SegmentCabinKey, 596
 - stdair::SegmentDate, 603
 - stdair::SegmentDateKey, 607
 - stdair::SegmentPeriod, 613

- stdair::SegmentPeriodKey, [617](#)
- stdair::ServiceAbstract, [620](#)
- stdair::ServiceInitialisationType, [624](#)
- stdair::SnapshotStruct, [628](#)
- stdair::STDAIR_ServiceContext, [649](#)
- stdair::StructAbstract, [652](#)
- stdair::TimePeriod, [654](#)
- stdair::TimePeriodKey, [658](#)
- stdair::TravelSolutionStruct, [663](#)
- stdair::VirtualClassStruct, [669](#)
- stdair::YieldFeatures, [672](#)
- stdair::YieldFeaturesKey, [676](#)
- stdair::YieldRange, [679](#)
- stdair::YieldStore, [681](#)
- stdair::YieldStoreKey, [684](#)
- toString
 - stdair::AirlineClassList, [241](#)
 - stdair::AirlineClassListKey, [244](#)
 - stdair::AirlineFeature, [247](#)
 - stdair::AirlineFeatureKey, [250](#)
 - stdair::AirportPair, [255](#)
 - stdair::AirportPairKey, [259](#)
 - stdair::BasDBParams, [263](#)
 - stdair::BasLogParams, [267](#)
 - stdair::BomAbstract, [271](#)
 - stdair::BomHolder, [283](#)
 - stdair::BomHolderKey, [285](#)
 - stdair::BomRoot, [305](#)
 - stdair::BomRootKey, [308](#)
 - stdair::BookingClass, [319](#)
 - stdair::BookingClassKey, [325](#)
 - stdair::Bucket, [335](#)
 - stdair::BucketKey, [339](#)
 - stdair::DatePeriod, [355](#)
 - stdair::DatePeriodKey, [358](#)
 - stdair::EventQueue, [382](#)
 - stdair::EventQueueKey, [390](#)
 - stdair::FareFamily, [416](#)
 - stdair::FareFamilyKey, [419](#)
 - stdair::FareFeatures, [422](#)
 - stdair::FareFeaturesKey, [428](#)
 - stdair::FlightDate, [439](#)
 - stdair::FlightDateKey, [443](#)
 - stdair::FlightPeriod, [446](#)
 - stdair::FlightPeriodKey, [449](#)
 - stdair::GuillotineBlock, [464](#)
 - stdair::GuillotineBlockKey, [469](#)
 - stdair::Inventory, [475](#)
 - stdair::InventoryKey, [478](#)
 - stdair::KeyAbstract, [482](#)
 - stdair::LegCabin, [494](#)
 - stdair::LegCabinKey, [501](#)
 - stdair::LegDate, [509](#)
 - stdair::LegDateKey, [513](#)
 - stdair::OnDDate, [530](#)
 - stdair::OnDDateKey, [534](#)
 - stdair::ParsedKey, [541](#)
 - stdair::PosChannel, [558](#)
 - stdair::PosChannelKey, [562](#)
 - stdair::SegmentCabin, [591](#)
 - stdair::SegmentCabinKey, [596](#)
 - stdair::SegmentDate, [603](#)
 - stdair::SegmentDateKey, [608](#)
 - stdair::SegmentPeriod, [614](#)
 - stdair::SegmentPeriodKey, [617](#)
 - stdair::TimePeriod, [655](#)
 - stdair::TimePeriodKey, [659](#)
 - stdair::YieldFeatures, [672](#)
 - stdair::YieldFeaturesKey, [676](#)
 - stdair::YieldStore, [682](#)
 - stdair::YieldStoreKey, [684](#)
 - stdair_test::BookingClass, [310](#)
 - stdair_test::Cabin, [341](#)
- TravelSolutionList_T
 - stdair, [188](#)
- TravelSolutionStruct
 - stdair::TravelSolutionStruct, [660](#)
- TRIP_TYPE_INBOUND
 - stdair, [216](#), [234](#)
- TRIP_TYPE_ONE_WAY
 - stdair, [216](#), [234](#)
- TRIP_TYPE_OUTBOUND
 - stdair, [216](#), [234](#)
- TRIP_TYPE_ROUND_TRIP
 - stdair, [216](#), [234](#)
- TripType_T
 - stdair, [191](#)
- TypeWithSize, [665](#)
 - UInt, [665](#)
- TypeWithSize< 4 >, [665](#)
 - Int, [666](#)
 - UInt, [666](#)
- TypeWithSize< 8 >, [666](#)
 - Int, [666](#)
 - UInt, [666](#)
- UInt
 - TypeWithSize, [665](#)
 - TypeWithSize< 4 >, [666](#)
 - TypeWithSize< 8 >, [666](#)

- uint1_4_p_t
 - stdair, [179](#)
- uint2_p_t
 - stdair, [179](#)
- uint4_p_t
 - stdair, [179](#)
- Unbind
 - swift::SKeymap, [627](#)
- UniformDistribution_T
 - stdair, [205](#)
- UniformGenerator_T
 - stdair, [206](#)
- UniformSeed_T
 - stdair, [205](#)
- UnsignedIndex_T
 - stdair, [194](#)
- updateAirlineInDB
 - stdair::DBManagerForAirlines, [360](#)
- updateCurrentBidPrice
 - stdair::LegCabin, [494](#)
- updateFromReservation
 - stdair::LegCabin, [495](#)
 - stdair::SegmentCabin, [591](#)
- updatePreviousBidPrice
 - stdair::LegCabin, [492](#)
- updateStatus
 - stdair::EventQueue, [384](#)
- UPR_T
 - stdair, [201](#)
- valid_argument
 - readline_autocomp.hpp, [1246](#)
- valueToKey
 - stdair::DictionaryManager, [369](#)
- ValueTypeIndexMap_T
 - stdair, [184](#)
- VERBOSE
 - stdair::LOG, [236](#)
- VirtualClassList_T
 - stdair, [189](#)
- VirtualClassMap_T
 - stdair, [189](#)
- VirtualClassStruct
 - stdair::VirtualClassStruct, [667](#)
- WARNING
 - stdair::LOG, [236](#)
- what
 - stdair::CodeConversionException, [346](#)
 - stdair::CodeDuplicationException, [347](#)
 - stdair::DocumentNotFoundException, [370](#)
 - stdair::EventException, [374](#)
 - stdair::EventQueueException, [388](#)
 - stdair::FileNotFoundException, [433](#)
 - stdair::KeyNotFoundException, [483](#)
 - stdair::MemoryAllocationException, [515](#)
 - stdair::NonInitialisedContainerException, [516](#)
 - stdair::NonInitialisedDBSessionManagerException, [517](#)
 - stdair::NonInitialisedLogServiceException, [519](#)
 - stdair::NonInitialisedRelationshipException, [520](#)
 - stdair::NonInitialisedServiceException, [521](#)
 - stdair::ObjectCreationDuplicationException, [523](#)
 - stdair::ObjectLinkingException, [524](#)
 - stdair::ObjectNotFoundException, [525](#)
 - stdair::ParserException, [544](#)
 - stdair::ParsingFileFailedException, [545](#)
 - stdair::RootException, [578](#)
 - stdair::SerialisationException, [619](#)
 - stdair::SQLDatabaseConnectionImpossibleException, [630](#)
 - stdair::SQLDatabaseException, [631](#)
- WTP_T
 - stdair, [197](#)
- WTPDemandPair_T
 - stdair, [197](#)
- xmalloc
 - readline_autocomp.hpp, [1243](#)
- year_p_t
 - stdair, [179](#)
- year_t
 - stdair, [178](#)
- Yield_T
 - stdair, [202](#)
- YieldDemandPair_T
 - stdair, [203](#)
- YieldFeatures
 - stdair::YieldFeatures, [671](#)
- YieldFeaturesDetailedList_T
 - stdair, [190](#)
- YieldFeaturesKey
 - stdair::YieldFeaturesKey, [675](#)

YieldFeaturesList_T
 stdair, [190](#)
YieldFeaturesMap_T
 stdair, [190](#)
YieldFeaturesWithKey_T
 stdair, [190](#)
YieldLevel_T
 stdair, [203](#)
YieldLevelDemandMap_T
 stdair, [203](#)
YieldRange
 stdair::YieldRange, [677](#), [678](#)
YieldStore
 stdair::YieldStore, [681](#)
YieldStoreKey
 stdair::YieldStoreKey, [684](#)
YieldStoreList_T
 stdair, [190](#)
YieldStoreMap_T
 stdair, [190](#)
YieldValue_T
 stdair, [192](#)