RMOL Reference Manual

1.00.1

Generated by Doxygen 1.4.7

Sun Jun 14 17:19:15 2015

Contents

1 RMOL Documentation

1.1 Getting Started

1.1 Getting Started

- Main features
- Installation
- Linking with RMOL
- Users Guide
- Tutorials
- Copyright and License
- Make a Difference
- Make a new release
- People

1.2 RMOL at SourceForge

1.2 RMOL at SourceForge

- Project page
- Download RMOL
- Open a ticket for a bug or feature

- Mailing lists
- Forums
  - Discuss about Development issues
  - Ask for Help
  - Discuss RMOL

1.3 RMOL Development

1.3 RMOL Development

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

1.4 External Libraries

1.4 External Libraries

- Boost (C++ STL extensions)
- Python
- MySQL client
- SOCI (C++ DB API)

1.5 Support RMOL

1.5 Support RMOL

1.6 About RMOL

1.6 About RMOL

RMOL is a C++ library of revenue management and optimisation classes and functions. RMOL mainly targets simulation purposes. N

RMOL makes an extensive use of existing open-source libraries for increased functionality, speed and accuracy. In particular GSL (*GNU Scientific Library*) and Boost (*C++ Standard Extensions*) libraries are used.

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

RMOL 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 RMOL 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 RMOL.

2 RMOL Namespace Index

2 RMOL Namespace Index

2.1 RMOL Namespace List

2.1 RMOL Namespace List

Here is a list of all namespaces with brief descriptions:

3 RMOL Hierarchical Index

3 RMOL Hierarchical Index

3.1 RMOL Class Hierarchy

3.1 RMOL Class Hierarchy

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

4 RMOL Class Index

4.1 RMOL Class List

4.1 RMOL Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

5 RMOL File Index

5 RMOL File Index

5.1 RMOL File List

5.1 RMOL File List

Here is a list of all files with brief descriptions:

6 RMOL Page Index

6 RMOL Page Index

6.1 RMOL Related Pages

6.1 RMOL Related Pages

Here is a list of all related documentation pages:

7 RMOL Namespace Documentation

7 RMOL Namespace Documentation

7.1 RMOL Namespace Reference

7.1 RMOL Namespace Reference

Classes

- class DPOptimiser
- class EMDetruncator
- class Emsr
- class EmsrUtils
- struct HistoricalBooking

  Structure keeping track, for a given class, of the number of historical bookings and of the censorship flag.

- struct HistoricalBookingHolder
- class MCOptimiser
- class DemandGeneratorList
- class PolicyHelper
- class SegmentSnapshotTableHelper
- class Utilities
- class BasedForecasting
- class DemandInputPreparation
- class Detruncator
- class FareAdjustment
- class Forecaster
- class HybridForecasting
- class InventoryParser

  Class filling the virtual class list (representing a list of classes/buckets) from a given input inventory.

- class MarginalRevenueTransformation

- class NewQFF
- class OldQFF
- class Optimiser
- class PreOptimiser
- class QForecasting
- class FacRmolServiceContext

  Factory for the service context.

- class RMOL_Service

  Interface for the RMOL Services.

- class OverbookingException

  Overbooking-related exception.

- class UnconstrainingException

  Unconstraining-related exception.

- class EmptyNestingStructException

  Empty nesting structure in unconstrainer exception.

- class MissingDCPException

  Missing a DCP in unconstrainer exception.

- class OptimisationException

  Optimisation-related exception.

- class PolicyException

  Policy-related exception.

- class ConvexHullException

  Convex Hull-related exception.

- class EmptyConvexHullException

  Empty convex hull exception.

- class FirstPolicyNotNullException

  Missing policy NULL in convex hull exception.

- class YieldConvexHullException

  Yield convex hull exception.

- class FareFamilyException

  Fare Family-related exception.

- class EmptyBookingClassListException

  Empty Booking Class List of Fare Family exception.

- class MissingBookingClassInFareFamilyException

  Missing Booking Class in Fare Family exception.

- class FareFamilyDemandVectorSizeException

  Fare Family demand exception.

- class RMOL_ServiceContext

  Inner class holding the context for the RMOL Service object.

Typedefs

- typedef std::list< BucketHolder * > BucketHolderList_T
- typedef std::list< FldDistributionParameters > DistributionParameterList_T
- typedef std::vector< HistoricalBooking > HistoricalBookingVector_T
- typedef boost::shared_ptr< RMOL_Service > RMOL_ServicePtr_T
- typedef std::vector< stdair::Flag_T > FlagVector_T
- typedef std::map< stdair::BookingClass *, stdair::MeanStdDevPair_T > BookingClassMeanStdDevPairMap_T

Variables

- const stdair::AirlineCode_T DEFAULT_RMOL_SERVICE_AIRLINE_CODE = "BA"
- const double DEFAULT_RMOL_SERVICE_CAPACITY = 1.0
- const int DEFAULT_NUMBER_OF_DRAWS_FOR_MC_SIMULATION = 10000
- const int DEFAULT_PRECISION = 10
- const double DEFAULT_EPSILON = 0.0001
- const double DEFAULT_STOPPING_CRITERION = 0.01
- const double DEFAULT_INITIALIZER_DOUBLE_NEGATIVE = -10.0
- const int DEFAULT_NUMBER_OF_DRAWS_FOR_MC_SIMULATION
- const int DEFAULT_PRECISION
- const double DEFAULT_EPSILON
- const double DEFAULT_STOPPING_CRITERION
- const double DEFAULT_INITIALIZER_DOUBLE_NEGATIVE
- const stdair::AirlineCode_T DEFAULT_RMOL_SERVICE_AIRLINE_CODE
- const double DEFAULT_RMOL_SERVICE_CAPACITY

7.1.1 Typedef Documentation

7.1.1.1 typedef std::list<BucketHolder*> RMOL::BucketHolderList_T

Define a vector (ordered list) of N bucket/classe holders.

Definition at line 16 of file BucketHolderTypes.hpp.

7.1.1.2 typedef std::list<FldDistributionParameters> RMOL::DistributionParameterList_T

Define the set of parameters, each of one wrapping a pair of distribution parameters (i.e., mean and standard deviation).

Definition at line 16 of file DistributionParameterList.hpp.

7.1.1.3 typedef std::vector<HistoricalBooking> RMOL::HistoricalBookingVector_T

Define a vector (ordered list) of N HistoricalBookings.

Definition at line 16 of file HistoricalBookingHolder.hpp.

7.1.1.4 typedef boost::shared_ptr<RMOL_Service> RMOL::RMOL_ServicePtr_T

Pointer on the RMOL Service handler.

Definition at line 176 of file RMOL_Types.hpp.

7.1.1.5 typedef std::vector<stdair::Flag_T> RMOL::FlagVector_T

Define the vector of censorship flags.

Definition at line 179 of file RMOL_Types.hpp.

7.1.1.6 typedef std::map<stdair::BookingClass*, stdair::MeanStdDevPair_T> RMOL::BookingClassMeanStdDevPairMap_T

Define the map between booking class and demand.

Definition at line 182 of file RMOL_Types.hpp.

7.1.2 Variable Documentation

7.1.2.1 const stdair::AirlineCode_T RMOL::DEFAULT_RMOL_SERVICE_AIRLINE_CODE = "BA"

Default airline code for the RMOL_Service.

Definition at line 10 of file BasConst.cpp.

7.1.2.2 const double RMOL::DEFAULT_RMOL_SERVICE_CAPACITY = 1.0

Default capacity for the RMOL_Service.

Definition at line 13 of file BasConst.cpp.

7.1.2.3 const int RMOL::DEFAULT_NUMBER_OF_DRAWS_FOR_MC_SIMULATION = 10000

Default value for the number of draws within the Monte-Carlo Integration algorithm.

Definition at line 17 of file BasConst.cpp.

Referenced by RMOL::MCOptimiser::optimisationByMCIntegration().

7.1.2.4 const int RMOL::DEFAULT_PRECISION = 10

Default value for the precision of the integral computation in the Dynamic Programming algorithm (100 means that the precision will be 0.01).

Definition at line 22 of file BasConst.cpp.

7.1.2.5 const double RMOL::DEFAULT_EPSILON = 0.0001

Default epsilon value to qualify a denominator

Definition at line 25 of file BasConst.cpp.

7.1.2.6 const double RMOL::DEFAULT_STOPPING_CRITERION = 0.01

Default stopping value for an iterative algorithm.

Definition at line 28 of file BasConst.cpp.

7.1.2.7 const double RMOL::DEFAULT_INITIALIZER_DOUBLE_NEGATIVE = -10.0

Default negative value used to initialze a double variable.

Definition at line 31 of file BasConst.cpp.

7.1.2.8 const int RMOL::DEFAULT_NUMBER_OF_DRAWS_FOR_MC_SIMULATION

Default value for the number of draws within the Monte-Carlo Integration algorithm.

Definition at line 17 of file BasConst.cpp.

Referenced by RMOL::MCOptimiser::optimisationByMCIntegration().

7.1.2.9 const int RMOL::DEFAULT_PRECISION

Default value for the precision of the integral computation in the Dynamic Programming algorithm (100 means that the precision will be 0.01).

Definition at line 22 of file BasConst.cpp.

7.1.2.10 const double RMOL::DEFAULT_EPSILON

Default epsilon value to qualify a denominator

Definition at line 25 of file BasConst.cpp.

7.1.2.11 const double RMOL::DEFAULT_STOPPING_CRITERION

Default stopping value for an iterative algorithm.

Definition at line 28 of file BasConst.cpp.

7.1.2.12 const double RMOL::DEFAULT_INITIALIZER_DOUBLE_NEGATIVE

Default negative value used to initialze a double variable.

Definition at line 31 of file BasConst.cpp.

7.1.2.13 const stdair::AirlineCode_T RMOL::DEFAULT_RMOL_SERVICE_AIRLINE_CODE

Default airline code for the RMOL_Service.

Definition at line 10 of file BasConst.cpp.

7.1.2.14 const double RMOL::DEFAULT_RMOL_SERVICE_CAPACITY

Default capacity for the RMOL_Service.

Definition at line 13 of file BasConst.cpp.

7.2 stdair Namespace Reference

7.2 stdair Namespace Reference

Forward declarations.

7.2.1 Detailed Description

Forward declarations.

Forward declarations.

8 RMOL Class Documentation

8 RMOL Class Documentation

8.1 RMOL::BasedForecasting Class Reference

8.1 RMOL::BasedForecasting Class Reference

`#include <rmol/command/BasedForecasting.hpp>`

Static Public Member Functions

- static bool forecast (stdair::SegmentCabin &, const stdair::Date_T &, const stdair::DTD_T &, const stdair::UnconstrainingMethod &, const stdair::NbOfSegments_T &)
- static void prepareHistoricalBooking (const stdair::SegmentCabin &, const stdair::BookingClass &, const stdair::SegmentSnapshotTable &, HistoricalBookingHolder &, const stdair::DCP_T &, const stdair::DCP_T &, const stdair::NbOfSegments_T &, const stdair::NbOfSegments_T &)

8.1.1 Detailed Description

Class wrapping the forecasting algorithms.

Definition at line 21 of file BasedForecasting.hpp.

8.1.2 Member Function Documentation

8.1.2.1 bool RMOL::BasedForecasting::forecast (stdair::SegmentCabin &, const stdair::Date_T &, const stdair::DTD_T &, const stdair::UnconstrainingMethod &, const stdair::NbOfSegments_T &) [static]

Forecast demand for a segment cabin. Compute for the current DTD the mean and the standard deviation of unconstrained bookings of similar flights.

Parameters:

stdair::SegmentCabin& Current Segment Cabin

const stdair::Date_T& Current Date

const stdair::DTD_T& Current DTD

const stdair::UnconstrainingMethod& Method used for the unconstraining

const stdair::NbOfSegments_T& Number of usable historical segments

Definition at line 31 of file BasedForecasting.cpp.

References RMOL::Utilities::computeDistributionParameters(), RMOL::HistoricalBookingHolder::getNbOfFlights(), RMOL::SegmentSnapshotTableHelper::getNbOfSegmentAlreadyPassedThisDTD(), RMOL::HistoricalBookingHolder::getUnconstrainedDemand(), prepareHistoricalBooking(), and RMOL::Detruncator::unconstrain().

8.1.2.2 void RMOL::BasedForecasting::prepareHistoricalBooking (const stdair::SegmentCabin &, const stdair::BookingClass &, const stdair::SegmentSnapshotTable &, HistoricalBookingHolder &, const stdair::DCP_T &, const stdair::DCP_T &, const stdair::NbOfSegments_T &, const stdair::NbOfSegments_T &) [static]

Prepare the historical booking figures for a given cabin

Parameters:

const stdair::DCP_T& DCP range start

const stdair::DCP_T& DCP range end

const stdair::NbOfSegments_T& Segment range start index

const stdair::NbOfSegments_T& Segment range end index

Definition at line 121 of file BasedForecasting.cpp.

References RMOL::HistoricalBookingHolder::addHistoricalBooking().

Referenced by forecast().

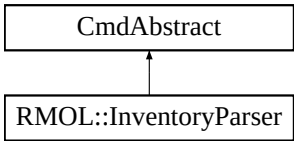
The documentation for this class was generated from the following files:

- rmol/command/BasedForecasting.hpp
- rmol/command/BasedForecasting.cpp

8.2 CmdAbstract Class Reference

8.2 CmdAbstract Class Reference

Inheritance diagram for CmdAbstract::



The documentation for this class was generated from the following file:

- rmol/command/InventoryParser.hpp

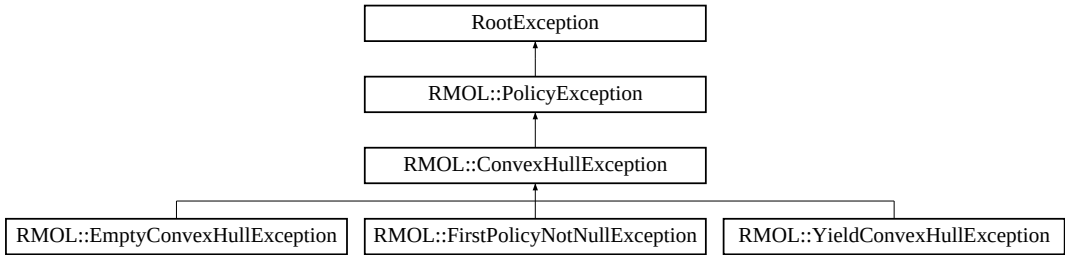
8.3 RMOL::ConvexHullException Class Reference

8.3 RMOL::ConvexHullException Class Reference

Convex Hull-related exception.

`#include <rmol/RMOL_Types.hpp>`

Inheritance diagram for RMOL::ConvexHullException::



Public Member Functions

- ConvexHullException (const std::string &iWhat)

8.3.1 Detailed Description

Convex Hull-related exception.

Definition at line 93 of file RMOL_Types.hpp.

8.3.2 Constructor & Destructor Documentation

8.3.2.1 RMOL::ConvexHullException::ConvexHullException (const std::string & *iWhat*) `[inline]`

Constructor.

Definition at line 96 of file RMOL_Types.hpp.

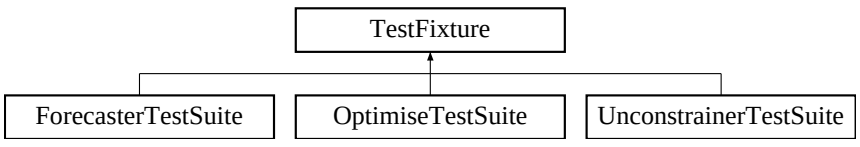
The documentation for this class was generated from the following file:

- rmol/RMOL_Types.hpp

8.4 TestFixture Class Reference

8.4 TestFixture Class Reference

Inheritance diagram for TestFixture::



The documentation for this class was generated from the following files:

- test/rmol/OptimiseTestSuite.hpp
- test/rmol/ForecasterTestSuite.hpp
- test/rmol/UnconstrainerTestSuite.hpp

8.5 RMOL::DemandGeneratorList Class Reference

8.5 RMOL::DemandGeneratorList Class Reference

`#include <rmol/bom/old/DemandGeneratorList.hpp>`

Public Member Functions

- DemandGeneratorList ()
- DemandGeneratorList (const DemandGeneratorList &)
- DemandGeneratorList (const DistributionParameterList_T &)
- virtual ~DemandGeneratorList ()
- void generateVariateList (VariateList_T &) const

Protected Types

- typedef std::list< Gaussian > DemandGeneratorList_T

8.5.1 Detailed Description

Wrapper around a set of Gaussian Random Generators.

Definition at line 17 of file DemandGeneratorList.hpp.

8.5.2 Member Typedef Documentation

8.5.2.1 typedef std::list<Gaussian> RMOL::DemandGeneratorList::DemandGeneratorList_T [protected]

Define a (ordered) set of Gaussian Random Generators.

Definition at line 20 of file DemandGeneratorList.hpp.

8.5.3 Constructor & Destructor Documentation

8.5.3.1 RMOL::DemandGeneratorList::DemandGeneratorList ()

Constructors.

Definition at line 10 of file DemandGeneratorList.cpp.

8.5.3.2 RMOL::DemandGeneratorList::DemandGeneratorList (const DemandGeneratorList &)

Definition at line 17 of file DemandGeneratorList.cpp.

8.5.3.3 RMOL::DemandGeneratorList::DemandGeneratorList (const DistributionParameterList_T &)

List of distribution parameters (mean, standard deviation).

Definition at line 25 of file DemandGeneratorList.cpp.

8.5.3.4 RMOL::DemandGeneratorList::~DemandGeneratorList () [virtual]

Destructors.

Definition at line 30 of file DemandGeneratorList.cpp.

8.5.4 Member Function Documentation

8.5.4.1 void RMOL::DemandGeneratorList::generateVariateList (VariateList_T &) const

Definition at line 50 of file DemandGeneratorList.cpp.

The documentation for this class was generated from the following files:

- rmol/bom/old/DemandGeneratorList.hpp
- rmol/bom/old/DemandGeneratorList.cpp

8.6 RMOL::DemandInputPreparation Class Reference

8.6 RMOL::DemandInputPreparation Class Reference

`#include <rmol/command/DemandInputPreparation.hpp>`

Static Public Member Functions

- static bool prepareDemandInput (const stdair::SegmentCabin &)

8.6.1 Detailed Description

Class wrapping the pre-optimisation algorithms.

Definition at line 21 of file DemandInputPreparation.hpp.

8.6.2 Member Function Documentation

8.6.2.1 bool RMOL::DemandInputPreparation::prepareDemandInput (const stdair::SegmentCabin &) [static]

Prepare the demand input for the optimser.

Definition at line 23 of file DemandInputPreparation.cpp.

The documentation for this class was generated from the following files:

- rmol/command/DemandInputPreparation.hpp
- rmol/command/DemandInputPreparation.cpp

8.7 RMOL::Detruncator Class Reference

8.7 RMOL::Detruncator Class Reference

`#include <rmol/command/Detruncator.hpp>`

Static Public Member Functions

- static void unconstrain (HistoricalBookingHolder &, const stdair::UnconstrainingMethod &)

8.7.1 Detailed Description

Class wrapping the principal unconstraining algorithms and some accessory algorithms.

Definition at line 20 of file Detruncator.hpp.

8.7.2 Member Function Documentation

8.7.2.1 void RMOL::Detruncator::unconstrain (HistoricalBookingHolder &, const stdair::UnconstrainingMethod &) [static]

Unconstrain booking figures between two DCP's.

Definition at line 17 of file Detruncator.cpp.

References RMOL::EMDetruncator::unconstrain().

Referenced by RMOL::QForecasting::forecast(), RMOL::OldQFF::forecast(), RMOL::Hybrid-Forecasting::forecast(), and RMOL::BasedForecasting::forecast().

The documentation for this class was generated from the following files:

- rmol/command/Detruncator.hpp
- rmol/command/Detruncator.cpp

8.8 RMOL::DPOptimiser Class Reference

8.8 RMOL::DPOptimiser Class Reference

`#include <rmol/bom/DPOptimiser.hpp>`

Static Public Member Functions

- static void optimalOptimisationByDP (stdair::LegCabin &)
- static double cdfGaussianQ (const double, const double)

8.8.1 Detailed Description

Utility methods for the Dynamic Programming algorithms.

Definition at line 17 of file DPOptimiser.hpp.

8.8.2 Member Function Documentation

8.8.2.1 void RMOL::DPOptimiser::optimalOptimisationByDP (stdair::LegCabin &) [static]

Dynamic Programming to compute the cumulative protection levels and booking limits (described in the book Revenue Management - Talluri & Van Ryzin, p.41-42).

Definition at line 22 of file DPOptimiser.cpp.

Referenced by RMOL::Optimiser::optimalOptimisationByDP().

8.8.2.2 static double RMOL::DPOptimiser::cdfGaussianQ (const *double*, const *double*) [static]

Compute the cdf_Q of a gaussian.

The documentation for this class was generated from the following files:

- rmol/bom/DPOptimiser.hpp
- rmol/bom/DPOptimiser.cpp

8.9 RMOL::EMDetruncator Class Reference

8.9 RMOL::EMDetruncator Class Reference

`#include <rmol/bom/EMDetruncator.hpp>`

Static Public Member Functions

- static void unconstrain (HistoricalBookingHolder &)

8.9.1 Detailed Description

Utility for the Expectation-Maximisation algorithm.

Definition at line 12 of file EMDetruncator.hpp.

8.9.2 Member Function Documentation

8.9.2.1 void RMOL::EMDetruncator::unconstrain (HistoricalBookingHolder &) [static]

Unconstrain the censored booking data using the Expection-Maximisation algorithm.

Definition at line 20 of file EMDetruncator.cpp.

References RMOL::HistoricalBookingHolder::getDemandMean(), RMOL::HistoricalBookingHolder::getListOfToBeUnconstrainedFlags(), RMOL::HistoricalBookingHolder::getNbOfFlights(), RMOL::HistoricalBookingHolder::getNbOfUncensoredBookings(), RMOL::HistoricalBookingHolder::getNbOfUncensoredData(), RMOL::HistoricalBookingHolder::getStandardDeviation(), RMOL::HistoricalBookingHolder::getUncensoredStandardDeviation(), RMOL::HistoricalBookingHolder::getUnconstrainedDemand(), and RMOL::HistoricalBookingHolder::setUnconstrainedDemand().

Referenced by RMOL::Detruncator::unconstrain().

The documentation for this class was generated from the following files:

- rmol/bom/EMDetruncator.hpp
- rmol/bom/EMDetruncator.cpp

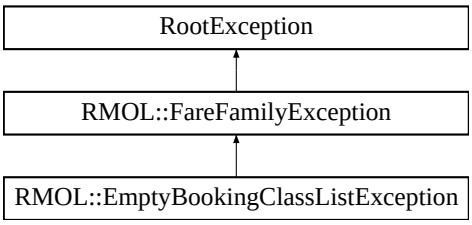
8.10 RMOL::EmptyBookingClassListException Class Reference

8.10 RMOL::EmptyBookingClassListException Class Reference

Empty Booking Class List of Fare Family exception.

`#include <rmol/RMOL_Types.hpp>`

Inheritance diagram for RMOL::EmptyBookingClassListException::



Public Member Functions

- EmptyBookingClassListException (const std::string &iWhat)

8.10.1 Detailed Description

Empty Booking Class List of Fare Family exception.

Definition at line 144 of file RMOL_Types.hpp.

8.10.2 Constructor & Destructor Documentation

8.10.2.1 RMOL::EmptyBookingClassListException::EmptyBookingClassListException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 147 of file RMOL_Types.hpp.

The documentation for this class was generated from the following file:

- rmol/RMOL_Types.hpp

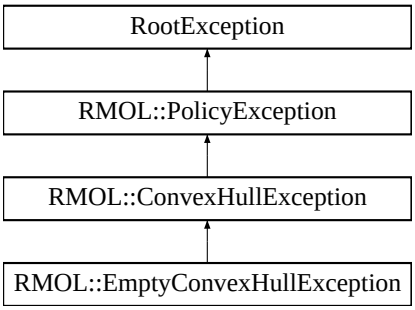
8.11 RMOL::EmptyConvexHullException Class Reference

8.11 RMOL::EmptyConvexHullException Class Reference

Empty convex hull exception.

`#include <rmol/RMOL_Types.hpp>`

Inheritance diagram for RMOL::EmptyConvexHullException::



Public Member Functions

- EmptyConvexHullException (const std::string &iWhat)

8.11.1 Detailed Description

Empty convex hull exception.

Definition at line 103 of file RMOL_Types.hpp.

8.11.2 Constructor & Destructor Documentation

8.11.2.1 RMOL::EmptyConvexHullException::EmptyConvexHullException (const std::string & *iWhat*) `[inline]`

Constructor.

Definition at line 106 of file RMOL_Types.hpp.

The documentation for this class was generated from the following file:

- rmol/RMOL_Types.hpp

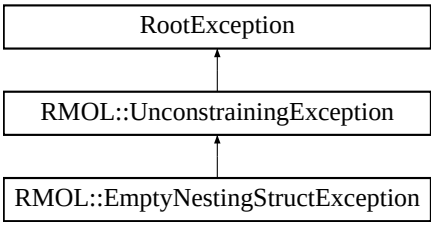
8.12 RMOL::EmptyNestingStructException Class Reference

8.12 RMOL::EmptyNestingStructException Class Reference

Empty nesting structure in unconstrainer exception.

`#include <rmol/RMOL_Types.hpp>`

Inheritance diagram for RMOL::EmptyNestingStructException::



Public Member Functions

- EmptyNestingStructException (const std::string &iWhat)

8.12.1 Detailed Description

Empty nesting structure in unconstrainer exception.

Definition at line 52 of file RMOL_Types.hpp.

8.12.2 Constructor & Destructor Documentation

8.12.2.1 RMOL::EmptyNestingStructException::EmptyNestingStructException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 55 of file RMOL_Types.hpp.

The documentation for this class was generated from the following file:

- rmol/RMOL_Types.hpp

8.13 RMOL::Emsr Class Reference

8.13 RMOL::Emsr Class Reference

`#include <rmol/bom/Emsr.hpp>`

Static Public Member Functions

- static void heuristicOptimisationByEmsr (stdair::LegCabin &)
- static void heuristicOptimisationByEmsrA (stdair::LegCabin &)
- static void heuristicOptimisationByEmsrB (stdair::LegCabin &)

8.13.1 Detailed Description

Class Implementing the EMSR algorithm for Bid-Price Vector computing.

Definition at line 18 of file Emsr.hpp.

8.13.2 Member Function Documentation

8.13.2.1 void RMOL::Emsr::heuristicOptimisationByEmsr (stdair::LegCabin &) [static]

Compute the Bid-Price Vector using the EMSR algorithm. Then compute the protection levels and booking limits by using the BPV.

For each class/bucket j with yield pj and demand Dj, compute pj∗Pr(Dj>=x) with x the capacity index. This value is called the EMSR (Expected Marginal Seat Revenue) of the class/bucket j with the remaining capacity of x. Thus, we have for each class/bucket a list of EMSR values. We merge all these lists and sort the values from high to low in order to obtain the BPV.

Definition at line 108 of file Emsr.cpp.

References RMOL::EmsrUtils::computeEmsrValue().

Referenced by RMOL::Optimiser::heuristicOptimisationByEmsr().

8.13.2.2 void RMOL::Emsr::heuristicOptimisationByEmsrA (stdair::LegCabin &) [static]

Calculate the optimal protections for the set of buckets/classes given in input, and update those buckets accordingly.

Definition at line 21 of file Emsr.cpp.

References RMOL::EmsrUtils::computeProtectionLevel().

Referenced by RMOL::Optimiser::heuristicOptimisationByEmsrA().

8.13.2.3 void RMOL::Emsr::heuristicOptimisationByEmsrB (stdair::LegCabin &) [static]

Complute the protection levels and booking limites by using the EMSR-b algorithm.

Definition at line 64 of file Emsr.cpp.

References RMOL::EmsrUtils::computeAggregatedVirtualClass(), and RMOL::EmsrUtils::computeProtectionLevel().

Referenced by RMOL::Optimiser::heuristicOptimisationByEmsrB().

The documentation for this class was generated from the following files:

- rmol/bom/Emsr.hpp
- rmol/bom/Emsr.cpp

8.14 RMOL::EmsrUtils Class Reference

8.14 RMOL::EmsrUtils Class Reference

`#include <rmol/bom/EmsrUtils.hpp>`

Static Public Member Functions

- static void computeAggregatedVirtualClass (stdair::VirtualClassStruct &, stdair::VirtualClassStruct &)
- static const stdair::ProtectionLevel_T computeProtectionLevel (stdair::VirtualClassStruct &, stdair::VirtualClassStruct &)
- static const double computeEmsrValue (double, stdair::VirtualClassStruct &)

8.14.1 Detailed Description

Forward declarations.

Definition at line 19 of file EmsrUtils.hpp.

8.14.2 Member Function Documentation

8.14.2.1 void RMOL::EmsrUtils::computeAggregatedVirtualClass (stdair::VirtualClassStruct &, stdair::VirtualClassStruct &) [static]

Compute the aggregated class/bucket of classes/buckets 1,..,j for EMSR-b algorithm.

Definition at line 19 of file EmsrUtils.cpp.

Referenced by RMOL::Emsr::heuristicOptimisationByEmsrB().

8.14.2.2 const stdair::ProtectionLevel_T RMOL::EmsrUtils::computeProtectionLevel (stdair::VirtualClassStruct &, stdair::VirtualClassStruct &) [static]

Compute the protection level using the Little-Wood formular.

Definition at line 53 of file EmsrUtils.cpp.

Referenced by RMOL::Emsr::heuristicOptimisationByEmsrA(), and RMOL::Emsr::heuristicOptimisationByEmsrB().

8.14.2.3 const double RMOL::EmsrUtils::computeEmsrValue (double, stdair::VirtualClassStruct &) [static]

Compute the EMSR value of a class/bucket.

Definition at line 80 of file EmsrUtils.cpp.

Referenced by RMOL::Emsr::heuristicOptimisationByEmsr().

The documentation for this class was generated from the following files:

- rmol/bom/EmsrUtils.hpp
- rmol/bom/EmsrUtils.cpp

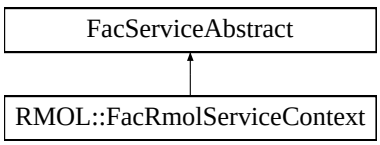
8.15 RMOL::FacRmolServiceContext Class Reference

8.15 RMOL::FacRmolServiceContext Class Reference

Factory for the service context.

`#include <rmol/factory/FacRmolServiceContext.hpp>`

Inheritance diagram for RMOL::FacRmolServiceContext::



Public Member Functions

- ~FacRmolServiceContext ()
- RMOL_ServiceContext & create ()

Static Public Member Functions

- static FacRmolServiceContext & instance ()

Protected Member Functions

- FacRmolServiceContext ()

8.15.1 Detailed Description

Factory for the service context.

Definition at line 22 of file FacRmolServiceContext.hpp.

8.15.2 Constructor & Destructor Documentation

8.15.2.1 RMOL::FacRmolServiceContext::~FacRmolServiceContext ()

Destructor.

The Destruction put the _instance to NULL in order to be clean for the next FacSimfqtService-Context::instance().

Definition at line 17 of file FacRmolServiceContext.cpp.

8.15.2.2 RMOL::FacRmolServiceContext::FacRmolServiceContext () [inline, protected]

Default Constructor.

This constructor is protected in order to ensure the singleton pattern.

Definition at line 57 of file FacRmolServiceContext.hpp.

Referenced by instance().

8.15.3 Member Function Documentation

8.15.3.1 FacRmolServiceContext & RMOL::FacRmolServiceContext::instance () [static]

Provide the unique instance.

The singleton is instantiated when first used.

Returns:

FacServiceContext&

Definition at line 22 of file FacRmolServiceContext.cpp.

References FacRmolServiceContext().

8.15.3.2 RMOL_ServiceContext & RMOL::FacRmolServiceContext::create ()

Create a new ServiceContext object.

This new object is added to the list of instantiated objects.

Returns:

ServiceContext& The newly created object.

Definition at line 34 of file FacRmolServiceContext.cpp.

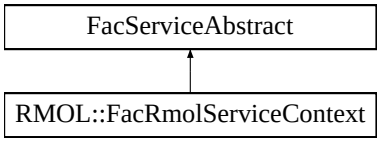
The documentation for this class was generated from the following files:

- rmol/factory/FacRmolServiceContext.hpp
- rmol/factory/FacRmolServiceContext.cpp

8.16 FacServiceAbstract Class Reference

8.16 FacServiceAbstract Class Reference

Inheritance diagram for FacServiceAbstract::



The documentation for this class was generated from the following file:

- rmol/factory/FacRmolServiceContext.hpp

8.17 RMOL::FareAdjustment Class Reference

8.17 RMOL::FareAdjustment Class Reference

`#include <rmol/command/FareAdjustment.hpp>`

Static Public Member Functions

- static bool adjustYield (const stdair::SegmentCabin &)

8.17.1 Detailed Description

Class wrapping the pre-optimisation algorithms.

Definition at line 21 of file FareAdjustment.hpp.

8.17.2 Member Function Documentation

8.17.2.1 bool RMOL::FareAdjustment::adjustYield (const stdair::SegmentCabin &) [static]

Prepare the demand input for the optimser.

Definition at line 23 of file FareAdjustment.cpp.

The documentation for this class was generated from the following files:

- rmol/command/FareAdjustment.hpp
- rmol/command/FareAdjustment.cpp

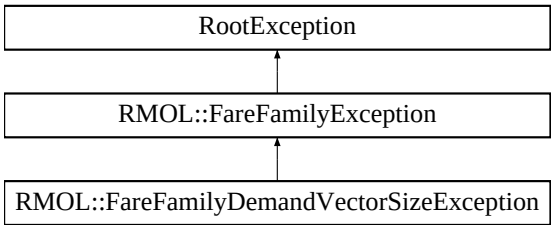
8.18 RMOL::FareFamilyDemandVectorSizeException Class Reference

8.18 RMOL::FareFamilyDemandVectorSizeException Class Reference

Fare Family demand exception.

`#include <rmol/RMOL_Types.hpp>`

Inheritance diagram for RMOL::FareFamilyDemandVectorSizeException::



Public Member Functions

- FareFamilyDemandVectorSizeException (const std::string &iWhat)

8.18.1 Detailed Description

Fare Family demand exception.

Definition at line 164 of file RMOL_Types.hpp.

8.18.2 Constructor & Destructor Documentation

8.18.2.1 RMOL::FareFamilyDemandVectorSizeException::FareFamilyDemandVectorSizeException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 167 of file RMOL_Types.hpp.

The documentation for this class was generated from the following file:

- rmol/RMOL_Types.hpp

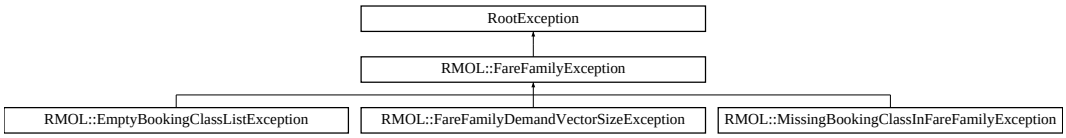
8.19 RMOL::FareFamilyException Class Reference

8.19 RMOL::FareFamilyException Class Reference

Fare Family-related exception.

`#include <rmol/RMOL_Types.hpp>`

Inheritance diagram for RMOL::FareFamilyException::



Public Member Functions

- FareFamilyException (const std::string &iWhat)

8.19.1 Detailed Description

Fare Family-related exception.

Definition at line 134 of file RMOL_Types.hpp.

8.19.2 Constructor & Destructor Documentation

8.19.2.1 RMOL::FareFamilyException::FareFamilyException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 137 of file RMOL_Types.hpp.

The documentation for this class was generated from the following file:

- rmol/RMOL_Types.hpp

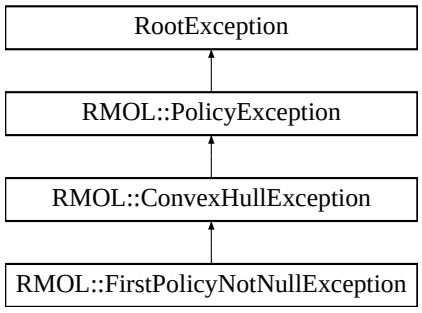
8.20 RMOL::FirstPolicyNotNullException Class Reference

8.20 RMOL::FirstPolicyNotNullException Class Reference

Missing policy NULL in convex hull exception.

`#include <rmol/RMOL_Types.hpp>`

Inheritance diagram for RMOL::FirstPolicyNotNullException::



Public Member Functions

- FirstPolicyNotNullException (const std::string &iWhat)

8.20.1 Detailed Description

Missing policy NULL in convex hull exception.

Definition at line 113 of file RMOL_Types.hpp.

8.20.2 Constructor & Destructor Documentation

8.20.2.1 RMOL::FirstPolicyNotNullException::FirstPolicyNotNullException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 116 of file RMOL_Types.hpp.

The documentation for this class was generated from the following file:

- rmol/RMOL_Types.hpp

8.21 RMOL::Forecaster Class Reference

8.21 RMOL::Forecaster Class Reference

`#include <rmol/command/Forecaster.hpp>`

Static Public Member Functions

- static bool forecast (stdair::FlightDate &, const stdair::DateTime_T &, const stdair::UnconstrainingMethod &, const stdair::ForecastingMethod &)

8.21.1 Detailed Description

Class wrapping the forecasting algorithms.

Definition at line 22 of file Forecaster.hpp.

8.21.2 Member Function Documentation

8.21.2.1 bool RMOL::Forecaster::forecast (stdair::FlightDate &, const stdair::DateTime_T &, const stdair::UnconstrainingMethod &, const stdair::ForecastingMethod &) [static]

Forecast demand for a flight-date.

Definition at line 35 of file Forecaster.cpp.

Referenced by RMOL::RMOL_Service::optimise().

The documentation for this class was generated from the following files:

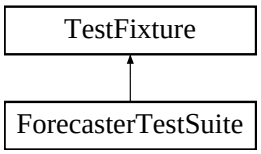
- rmol/command/Forecaster.hpp
- rmol/command/Forecaster.cpp

8.22 ForecasterTestSuite Class Reference

8.22 ForecasterTestSuite Class Reference

`#include <test/rmol/ForecasterTestSuite.hpp>`

Inheritance diagram for ForecasterTestSuite::



Public Member Functions

- void testQForecaster ()
- ForecasterTestSuite ()

Protected Attributes

- std::stringstream _describeKey

8.22.1 Detailed Description

Definition at line 6 of file ForecasterTestSuite.hpp.

8.22.2 Constructor & Destructor Documentation

8.22.2.1 ForecasterTestSuite::ForecasterTestSuite ()

Constructor.

8.22.3 Member Function Documentation

8.22.3.1 void ForecasterTestSuite::testQForecaster ()

Test Q-forecaster.

8.22.4 Member Data Documentation

8.22.4.1 std::stringstream ForecasterTestSuite::_describeKey [protected]

Definition at line 19 of file ForecasterTestSuite.hpp.

The documentation for this class was generated from the following file:

- test/rmol/ForecasterTestSuite.hpp

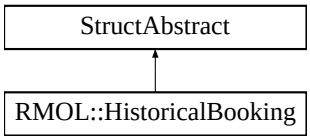
8.23 RMOL::HistoricalBooking Struct Reference

8.23 RMOL::HistoricalBooking Struct Reference

Structure keeping track, for a given class, of the number of historical bookings and of the censorship flag.

`#include <rmol/bom/HistoricalBooking.hpp>`

Inheritance diagram for RMOL::HistoricalBooking::



Public Member Functions

- const stdair::NbOfBookings_T & getNbOfBookings () const
- const stdair::NbOfBookings_T & getUnconstrainedDemand () const
- const stdair::Flag_T & getFlag () const
- void setUnconstrainedDemand (const stdair::NbOfBookings_T &iDemand)
- void setParameters (const stdair::NbOfBookings_T, const stdair::Flag_T)
- void toStream (std::ostream &ioOut) const
- const std::string describe () const
- void display () const
- HistoricalBooking (const stdair::NbOfBookings_T, const stdair::Flag_T)
- HistoricalBooking ()
- HistoricalBooking (const HistoricalBooking &)
- virtual ~HistoricalBooking ()

8.23.1 Detailed Description

Structure keeping track, for a given class, of the number of historical bookings and of the censorship flag.

Definition at line 17 of file HistoricalBooking.hpp.

8.23.2 Constructor & Destructor Documentation

8.23.2.1 RMOL::HistoricalBooking::HistoricalBooking (const stdair::NbOfBookings_T, const stdair::Flag_T)

Main constructor.

Definition at line 21 of file HistoricalBooking.cpp.

8.23.2.2 RMOL::HistoricalBooking::HistoricalBooking ()

Default constructor.

Definition at line 15 of file HistoricalBooking.cpp.

8.23.2.3 RMOL::HistoricalBooking::HistoricalBooking (const HistoricalBooking &)

Copy constructor.

Definition at line 29 of file HistoricalBooking.cpp.

8.23.2.4 RMOL::HistoricalBooking::~HistoricalBooking () [virtual]

Destructor.

Definition at line 36 of file HistoricalBooking.cpp.

8.23.3 Member Function Documentation

8.23.3.1 const stdair::NbOfBookings_T& RMOL::HistoricalBooking::getNbOfBookings () const [inline]

Getter for the booking.

Definition at line 22 of file HistoricalBooking.hpp.

Referenced by RMOL::HistoricalBookingHolder::calculateExpectedDemand(), RMOL::HistoricalBookingHolder::getHistoricalBooking(), RMOL::HistoricalBookingHolder::getUncensoredStandardDeviation(), and toStream().

8.23.3.2 const stdair::NbOfBookings_T& RMOL::HistoricalBooking::getUnconstrainedDemand () const [inline]

Getter for the unconstrained bookings.

Definition at line 26 of file HistoricalBooking.hpp.

Referenced by RMOL::HistoricalBookingHolder::getUnconstrainedDemand(), and toStream().

8.23.3.3 const stdair::Flag_T& RMOL::HistoricalBooking::getFlag () const [inline]

Getter for the flag of censorship: "false" means that the bookings are not censored.

Definition at line 31 of file HistoricalBooking.hpp.

Referenced by RMOL::HistoricalBookingHolder::getCensorshipFlag(), and toStream().

8.23.3.4 void RMOL::HistoricalBooking::setUnconstrainedDemand (const stdair::NbOfBookings_T & *iDemand*) [inline]

Setter for the unconstraining demand.

Definition at line 38 of file HistoricalBooking.hpp.

8.23.3.5 void RMOL::HistoricalBooking::setParameters (const stdair::NbOfBookings_T, const stdair::Flag_T)

Setter for all parameters.

Definition at line 41 of file HistoricalBooking.cpp.

8.23.3.6 void RMOL::HistoricalBooking::toStream (std::ostream & *ioOut*) const

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream

Returns:

ostream& the output stream.

Definition at line 57 of file HistoricalBooking.cpp.

References getFlag(), getNbOfBookings(), and getUnconstrainedDemand().

Referenced by display().

8.23.3.7 const std::string RMOL::HistoricalBooking::describe () const

Give a description of the structure (for display purposes).

Definition at line 48 of file HistoricalBooking.cpp.

8.23.3.8 void RMOL::HistoricalBooking::display () const

Display on standard output.

Definition at line 66 of file HistoricalBooking.cpp.

References toStream().

The documentation for this struct was generated from the following files:

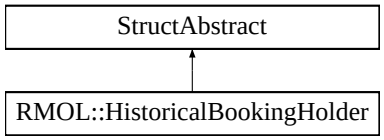
- rmol/bom/HistoricalBooking.hpp
- rmol/bom/HistoricalBooking.cpp

8.24 RMOL::HistoricalBookingHolder Struct Reference

8.24 RMOL::HistoricalBookingHolder Struct Reference

`#include <rmol/bom/HistoricalBookingHolder.hpp>`

Inheritance diagram for RMOL::HistoricalBookingHolder::



Public Member Functions

- const short getNbOfFlights () const
- const short getNbOfUncensoredData () const
- const stdair::NbOfBookings_T getNbOfUncensoredBookings () const
- const double getUncensoredStandardDeviation (const double &iMeanOfUncensoredBookings, const short iNbOfUncensoredData) const
- const double getDemandMean () const
- const double getStandardDeviation (const double) const

- const std::vector< bool > getListOfToBeUnconstrainedFlags () const
- const stdair::NbOfBookings_T & getHistoricalBooking (const short i) const
- const stdair::NbOfBookings_T & getUnconstrainedDemand (const short i) const
- const stdair::Flag_T & getCensorshipFlag (const short i) const
- const stdair::NbOfBookings_T & getUnconstrainedDemandOnFirstElement () const
- const stdair::NbOfBookings_T calculateExpectedDemand (const double, const double, const short, const stdair::NbOfBookings_T) const
- void setUnconstrainedDemand (const stdair::NbOfBookings_T &iExpectedDemand, const short i)
- void addHistoricalBooking (const HistoricalBooking &iHistoricalBooking)
- void toStream (std::ostream &ioOut) const
- const std::string describe () const
- void display () const
- virtual ~HistoricalBookingHolder ()
- HistoricalBookingHolder ()

8.24.1 Detailed Description

Holder of a HistoricalBookingList object (for memory allocation and recollection purposes).

Definition at line 23 of file HistoricalBookingHolder.hpp.

8.24.2 Constructor & Destructor Documentation

8.24.2.1 RMOL::HistoricalBookingHolder::~HistoricalBookingHolder () [virtual]

Destructor.

Definition at line 23 of file HistoricalBookingHolder.cpp.

8.24.2.2 RMOL::HistoricalBookingHolder::HistoricalBookingHolder ()

Constructor.

Protected to force the use of the Factory.

Definition at line 19 of file HistoricalBookingHolder.cpp.

8.24.3 Member Function Documentation

8.24.3.1 const short RMOL::HistoricalBookingHolder::getNbOfFlights () const

Get number of flights.

Definition at line 28 of file HistoricalBookingHolder.cpp.

Referenced by RMOL::QForecasting::forecast(), RMOL::OldQFF::forecast(), RMOL::HybridForecasting::forecast(), RMOL::BasedForecasting::forecast(), and RMOL::EMDetruncator::unconstrain().

8.24.3.2 const short RMOL::HistoricalBookingHolder::getNbOfUncensoredData () const

Get number of uncensored booking data.

Definition at line 33 of file HistoricalBookingHolder.cpp.

Referenced by RMOL::EMDetruncator::unconstrain().

8.24.3.3 const stdair::NbOfBookings_T RMOL::HistoricalBookingHolder::getNbOfUncensoredBookings () const

Get number of uncensored bookings.

Definition at line 49 of file HistoricalBookingHolder.cpp.

Referenced by RMOL::EMDetruncator::unconstrain().

8.24.3.4 const double RMOL::HistoricalBookingHolder::getUncensoredStandardDeviation (const double & *iMeanOfUncensoredBookings*, const short *iNbOfUncensoredData*) const

Get standard deviation of uncensored bookings.

Definition at line 69 of file HistoricalBookingHolder.cpp.

References RMOL::HistoricalBooking::getNbOfBookings().

Referenced by RMOL::EMDetruncator::unconstrain().

8.24.3.5 const double RMOL::HistoricalBookingHolder::getDemandMean () const

Get mean of historical demand.

Definition at line 95 of file HistoricalBookingHolder.cpp.

Referenced by RMOL::EMDetruncator::unconstrain().

8.24.3.6 const double RMOL::HistoricalBookingHolder::getStandardDeviation (const *double*) const

Get standard deviation of demand.

Definition at line 116 of file HistoricalBookingHolder.cpp.

Referenced by RMOL::EMDetruncator::unconstrain().

8.24.3.7 const std::vector< bool > RMOL::HistoricalBookingHolder::getListOfToBeUnconstrainedFlags () const

Get the list of flags of need to be unconstrained.

Definition at line 140 of file HistoricalBookingHolder.cpp.

Referenced by RMOL::EMDetruncator::unconstrain().

8.24.3.8 const stdair::NbOfBookings_T & RMOL::HistoricalBookingHolder::getHistoricalBooking (const short *i*) const

Get the historical booking of the (i+1)-th flight.

Definition at line 161 of file HistoricalBookingHolder.cpp.

References RMOL::HistoricalBooking::getNbOfBookings().

8.24.3.9 const stdair::NbOfBookings_T & RMOL::HistoricalBookingHolder::getUnconstrainedDemand (const short *i*) const

Get the unconstraining demand of the (i+1)-th flight.

Definition at line 169 of file HistoricalBookingHolder.cpp.

References RMOL::HistoricalBooking::getUnconstrainedDemand().

Referenced by RMOL::QForecasting::forecast(), RMOL::OldQFF::forecast(), RMOL::HybridForecasting::forecast(), RMOL::BasedForecasting::forecast(), getUnconstrainedDemandOnFirstElement(), and RMOL::EMDetruncator::unconstrain().

8.24.3.10 const stdair::Flag_T & RMOL::HistoricalBookingHolder::getCensorshipFlag (const short *i*) const

Get the flag of the (i+1)-th flight.

Definition at line 177 of file HistoricalBookingHolder.cpp.

References RMOL::HistoricalBooking::getFlag().

8.24.3.11 const stdair::NbOfBookings_T& RMOL::HistoricalBookingHolder::getUnconstrainedDemandOnFirstElement () const [inline]

Get the unconstraining demand of the first flight.

Definition at line 60 of file HistoricalBookingHolder.hpp.

References getUnconstrainedDemand().

8.24.3.12 const stdair::NbOfBookings_T RMOL::HistoricalBookingHolder::calculateExpectedDemand (const *double*, const *double*, const *short*, const stdair::NbOfBookings_T) const

Calculate the expected demand.

Definition at line 191 of file HistoricalBookingHolder.cpp.

References RMOL::HistoricalBooking::getNbOfBookings().

8.24.3.13 void RMOL::HistoricalBookingHolder::setUnconstrainedDemand (const stdair::NbOfBookings_T & *iExpectedDemand*, const short *i*)

Set the expected historical demand of the (i+1)-th flight.

Definition at line 185 of file HistoricalBookingHolder.cpp.

Referenced by RMOL::EMDetruncator::unconstrain().

8.24.3.14 void RMOL::HistoricalBookingHolder::addHistoricalBooking (const HistoricalBooking & *iHistoricalBooking*)

Add a HistoricalBooking object to the holder.

Definition at line 236 of file HistoricalBookingHolder.cpp.

Referenced by RMOL::BasedForecasting::prepareHistoricalBooking(), RMOL::QForecasting::preparePriceOrientedHistoricalBooking(), and RMOL::HybridForecasting::prepareProductOrientedHistoricalBooking().

8.24.3.15 void RMOL::HistoricalBookingHolder::toStream (std::ostream & *ioOut*) const

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream

Returns:

ostream& the output stream.

Definition at line 241 of file HistoricalBookingHolder.cpp.

Referenced by display().

8.24.3.16 const std::string RMOL::HistoricalBookingHolder::describe () const

Give a description of the structure (for display purposes).

Definition at line 265 of file HistoricalBookingHolder.cpp.

8.24.3.17 void RMOL::HistoricalBookingHolder::display () const

Display on standard output.

Definition at line 273 of file HistoricalBookingHolder.cpp.

References toStream().

The documentation for this struct was generated from the following files:

- rmol/bom/HistoricalBookingHolder.hpp
- rmol/bom/HistoricalBookingHolder.cpp

8.25 RMOL::HybridForecasting Class Reference

8.25 RMOL::HybridForecasting Class Reference

`#include <rmol/command/HybridForecasting.hpp>`

Static Public Member Functions

- static bool forecast (stdair::SegmentCabin &, const stdair::Date_T &, const stdair::DTD_T &, const stdair::UnconstrainingMethod &, const stdair::NbOfSegments_T &)
- static void prepareProductOrientedHistoricalBooking (const stdair::SegmentCabin &, const stdair::BookingClass &, const stdair::SegmentSnapshotTable &, HistoricalBookingHolder &, const stdair::DCP_T &, const stdair::DCP_T &, const stdair::NbOfSegments_T &, const stdair::NbOf-Segments_T &)

8.25.1 Detailed Description

Class wrapping the forecasting algorithms.

Definition at line 21 of file HybridForecasting.hpp.

8.25.2 Member Function Documentation

8.25.2.1 bool RMOL::HybridForecasting::forecast (stdair::SegmentCabin &, const stdair::Date_T &, const stdair::DTD_T &, const stdair::UnconstrainingMethod &, const stdair::NbOfSegments_T &) [static]

Forecast demand for a segment cabin.

Parameters:

stdair::SegmentCabin& Current Segment Cabin

const stdair::Date_T& Current Date

const stdair::DTD_T& Current DTD

const stdair::UnconstrainingMethod& Method used for the unconstraining

const stdair::NbOfSegments_T& Number of usable historical segments

Definition at line 32 of file HybridForecasting.cpp.

References RMOL::Utilities::computeDistributionParameters(), RMOL::QForecasting::forecast(), RMOL::HistoricalBookingHolder::getNbOfFlights(), RMOL::SegmentSnapshotTableHelper::getNbOfSegmentAlreadyPassedThisDTD(), RMOL::HistoricalBookingHolder::getUnconstrainedDemand(), prepareProductOrientedHistoricalBooking(), and RMOL::Detruncator::unconstrain().

8.25.2.2 void RMOL::HybridForecasting::prepareProductOrientedHistoricalBooking (const stdair::SegmentCabin &, const stdair::BookingClass &, const stdair::SegmentSnapshotTable &, HistoricalBookingHolder &, const stdair::DCP_T &, const stdair::DCP_T &, const stdair::NbOfSegments_T &, const stdair::NbOfSegments_T &) [static]

Prepare the historical product-oriented booking figures for a given cabin

Parameters:

const stdair::DCP_T& DCP range start

const stdair::DCP_T& DCP range end

const stdair::NbOfSegments_T& Segment range start index

const stdair::NbOfSegments_T& Segment range end index

Definition at line 125 of file HybridForecasting.cpp.

References RMOL::HistoricalBookingHolder::addHistoricalBooking().

Referenced by forecast().

The documentation for this class was generated from the following files:

- rmol/command/HybridForecasting.hpp
- rmol/command/HybridForecasting.cpp

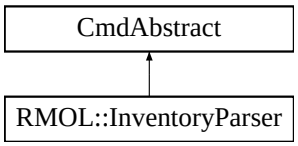
8.26 RMOL::InventoryParser Class Reference

8.26 RMOL::InventoryParser Class Reference

Class filling the virtual class list (representing a list of classes/buckets) from a given input inventory.

`#include <rmol/command/InventoryParser.hpp>`

Inheritance diagram for RMOL::InventoryParser::



Static Public Member Functions

- static bool parseInputFileAndBuildBom (const std::string &iInputFileName, stdair::BomRoot &)

8.26.1 Detailed Description

Class filling the virtual class list (representing a list of classes/buckets) from a given input inventory.

Definition at line 25 of file InventoryParser.hpp.

8.26.2 Member Function Documentation

8.26.2.1 bool RMOL::InventoryParser::parseInputFileAndBuildBom (const std::string & *iInputFileName*, stdair::BomRoot &) `[static]`

Parse the input values from a CSV-formatted inventory file.

Parameters:

const std::string& iInputFileName Inventory file to be parsed.

stdair::BomRoot& The BOM tree.

Returns:

bool Whether or not the parsing was successful.

Definition at line 36 of file InventoryParser.cpp.

Referenced by RMOL::RMOL_Service::parseAndLoad().

The documentation for this class was generated from the following files:

- rmol/command/InventoryParser.hpp
- rmol/command/InventoryParser.cpp

8.27 RMOL::MarginalRevenueTransformation Class Reference

8.27 RMOL::MarginalRevenueTransformation Class Reference

`#include <rmol/command/MarginalRevenueTransformation.hpp>`

Static Public Member Functions

- static bool prepareDemandInput (stdair::SegmentCabin &)

8.27.1 Detailed Description

Class wrapping the pre-optimisation algorithms.

Definition at line 21 of file MarginalRevenueTransformation.hpp.

8.27.2 Member Function Documentation

8.27.2.1 bool RMOL::MarginalRevenueTransformation::prepareDemandInput (stdair::SegmentCabin &) [static]

Prepare the demand input for the optimser.

Definition at line 28 of file MarginalRevenueTransformation.cpp.

The documentation for this class was generated from the following files:

- rmol/command/MarginalRevenueTransformation.hpp
- rmol/command/MarginalRevenueTransformation.cpp

8.28 RMOL::MCOptimiser Class Reference

8.28 RMOL::MCOptimiser Class Reference

`#include <rmol/bom/MCOptimiser.hpp>`

Static Public Member Functions

- static void optimalOptimisationByMCIntegration (stdair::LegCabin &)
- static stdair::GeneratedDemandVector_T generateDemandVector (const stdair::MeanValue_T &, const stdair::StdDevValue_T &, const stdair::NbOfSamples_T &)
- static void optimisationByMCIntegration (stdair::LegCabin &)

8.28.1 Detailed Description

Utility methods for the Monte-Carlo algorithms.

Definition at line 19 of file MCOptimiser.hpp.

8.28.2 Member Function Documentation

8.28.2.1 void RMOL::MCOptimiser::optimalOptimisationByMCIntegration (stdair::LegCabin &) [static]

Calculate the optimal protections for the set of buckets/classes given in input, and update those buckets accordingly.

The Monte Carlo Integration algorithm (see The Theory and Practice of Revenue Management, by Kalyan T. Talluri and Garret J. van Ryzin, Kluwer Academic Publishers, for the details) is used.

Definition at line 28 of file MCOptimiser.cpp.

Referenced by RMOL::Optimiser::optimalOptimisationByMCIntegration().

8.28.2.2 stdair::GeneratedDemandVector_T RMOL::MCOptimiser::generateDemandVector (const stdair::MeanValue_T &, const stdair::StdDevValue_T &, const stdair::NbOfSamples_T &) [static]

Monte-Carlo

Definition at line 154 of file MCOptimiser.cpp.

Referenced by optimisationByMCIntegration().

8.28.2.3 void RMOL::MCOptimiser::optimisationByMCIntegration (stdair::LegCabin &) [static]

Definition at line 175 of file MCOptimiser.cpp.

References RMOL::DEFAULT_NUMBER_OF_DRAWS_FOR_MC_SIMULATION, and generateDemandVector().

Referenced by RMOL::Optimiser::optimiseUsingOnDForecast().

The documentation for this class was generated from the following files:

- rmol/bom/MCOptimiser.hpp
- rmol/bom/MCOptimiser.cpp

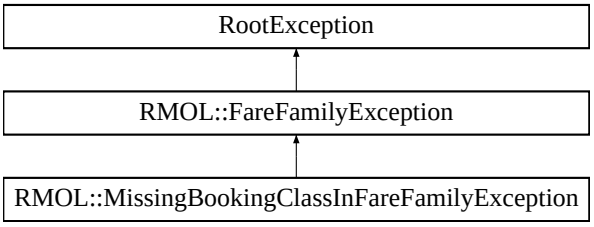
8.29 RMOL::MissingBookingClassInFareFamilyException Class Reference

8.29 RMOL::MissingBookingClassInFareFamilyException Class Reference

Missing Booking Class in Fare Family exception.

`#include <rmol/RMOL_Types.hpp>`

Inheritance diagram for RMOL::MissingBookingClassInFareFamilyException::



Public Member Functions

- MissingBookingClassInFareFamilyException (const std::string &iWhat)

8.29.1 Detailed Description

Missing Booking Class in Fare Family exception.

Definition at line 154 of file RMOL_Types.hpp.

8.29.2 Constructor & Destructor Documentation

8.29.2.1 RMOL::MissingBookingClassInFareFamilyException::MissingBookingClassInFareFamilyException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 157 of file RMOL_Types.hpp.

The documentation for this class was generated from the following file:

- rmol/RMOL_Types.hpp

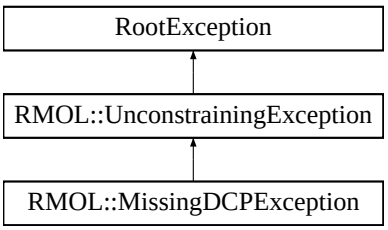
8.30 RMOL::MissingDCPException Class Reference

8.30 RMOL::MissingDCPException Class Reference

Missing a DCP in unconstrainer exception.

`#include <rmol/RMOL_Types.hpp>`

Inheritance diagram for RMOL::MissingDCPException::



Public Member Functions

- MissingDCPException (const std::string &iWhat)

8.30.1 Detailed Description

Missing a DCP in unconstrainer exception.

Definition at line 62 of file RMOL_Types.hpp.

8.30.2 Constructor & Destructor Documentation

8.30.2.1 RMOL::MissingDCPException::MissingDCPException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 65 of file RMOL_Types.hpp.

The documentation for this class was generated from the following file:

- rmol/RMOL_Types.hpp

8.31 RMOL::NewQFF Class Reference

8.31 RMOL::NewQFF Class Reference

`#include <rmol/command/NewQFF.hpp>`

Static Public Member Functions

- static bool forecast (stdair::SegmentCabin &, const stdair::Date_T &, const stdair::DTD_T &, const stdair::UnconstrainingMethod &, const stdair::NbOfSegments_T &)

8.31.1 Detailed Description

Class wrapping the forecasting algorithms.

Definition at line 23 of file NewQFF.hpp.

8.31.2 Member Function Documentation

8.31.2.1 bool RMOL::NewQFF::forecast (stdair::SegmentCabin &, const stdair::Date_T &, const stdair::DTD_T &, const stdair::UnconstrainingMethod &, const stdair::NbOfSegments_T &) [static]

Forecast demand for a segment cabin.

Parameters:

stdair::SegmentCabin& Current Segment Cabin

const stdair::Date_T& Current Date

const stdair::DTD_T& Current DTD

const stdair::UnconstrainingMethod& Method used for the unconstraining

const stdair::NbOfSegments_T& Number of usable historical segments

Definition at line 31 of file NewQFF.cpp.

The documentation for this class was generated from the following files:

- rmol/command/NewQFF.hpp
- rmol/command/NewQFF.cpp

8.32 RMOL::OldQFF Class Reference

8.32 RMOL::OldQFF Class Reference

`#include <rmol/command/OldQFF.hpp>`

Static Public Member Functions

- static bool forecast (stdair::SegmentCabin &, const stdair::Date_T &, const stdair::DTD_T &, const stdair::UnconstrainingMethod &, const stdair::NbOfSegments_T &)

8.32.1 Detailed Description

Class wrapping the forecasting algorithms.

Definition at line 23 of file OldQFF.hpp.

8.32.2 Member Function Documentation

8.32.2.1 bool RMOL::OldQFF::forecast (stdair::SegmentCabin &, const stdair::Date_T &, const stdair::DTD_T &, const stdair::UnconstrainingMethod &, const stdair::NbOfSegments_T &) [static]

Forecast demand for a segment cabin.

Parameters:

stdair::SegmentCabin& Current Segment Cabin

const stdair::Date_T& Current Date

const stdair::DTD_T& Current DTD

const stdair::UnconstrainingMethod& Method used for the unconstraining

const stdair::NbOfSegments_T& Number of usable historical segments

Definition at line 31 of file OldQFF.cpp.

References RMOL::Utilities::computeDistributionParameters(), RMOL::Utilities::computeSellUpFactorCurves(), RMOL::HistoricalBookingHolder::getNbOfFlights(), RMOL::SegmentSnapshotTableHelper::getNbOfSegmentAlreadyPassedThisDTD(), RMOL::HistoricalBookingHolder::getUnconstrainedDemand(), and RMOL::Detruncator::unconstrain().

The documentation for this class was generated from the following files:

- rmol/command/OldQFF.hpp
- rmol/command/OldQFF.cpp

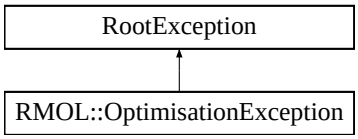
8.33 RMOL::OptimisationException Class Reference

8.33 RMOL::OptimisationException Class Reference

Optimisation-related exception.

`#include <rmol/RMOL_Types.hpp>`

Inheritance diagram for RMOL::OptimisationException::



Public Member Functions

- OptimisationException (const std::string &iWhat)

8.33.1 Detailed Description

Optimisation-related exception.

Definition at line 72 of file RMOL_Types.hpp.

8.33.2 Constructor & Destructor Documentation

8.33.2.1 RMOL::OptimisationException::OptimisationException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 75 of file RMOL_Types.hpp.

The documentation for this class was generated from the following file:

- rmol/RMOL_Types.hpp

8.34 RMOL::Optimiser Class Reference

8.34 RMOL::Optimiser Class Reference

`#include <rmol/command/Optimiser.hpp>`

Static Public Member Functions

- static void optimalOptimisationByMCIntegration (const stdair::NbOfSamples_T &, stdair::LegCabin &)
- static void optimalOptimisationByDP (stdair::LegCabin &)
- static void heuristicOptimisationByEmsr (stdair::LegCabin &)
- static void heuristicOptimisationByEmsrA (stdair::LegCabin &)
- static void heuristicOptimisationByEmsrB (stdair::LegCabin &)
- static bool optimise (stdair::FlightDate &, const stdair::OptimisationMethod &)
- static bool buildVirtualClassListForLegBasedOptimisation (stdair::LegCabin &)
- static double optimiseUsingOnDForecast (stdair::FlightDate &, const bool &iReduceFluctuations=false)

8.34.1 Detailed Description

Class wrapping the optimisation algorithms.

Definition at line 20 of file Optimiser.hpp.

8.34.2 Member Function Documentation

8.34.2.1 void RMOL::Optimiser::optimalOptimisationByMCIntegration (const stdair::NbOfSamples_T &, stdair::LegCabin &) [static]

Monte Carlo Integration algorithm.

Calculate the optimal protections for the set of buckets/classes given in input, and update those buckets accordingly.

The Monte Carlo Integration algorithm (see The Theory and Practice of Revenue Management, by Kalyan T. Talluri and Garret J. van Ryzin, Kluwer Academic Publishers, for the details) is used. Hence, K is the number of random draws to perform. 100 is a minimum for K, as statistics must be drawn from those random generations.

Definition at line 30 of file Optimiser.cpp.

References RMOL::MCOptimiser::optimalOptimisationByMCIntegration().

Referenced by RMOL::RMOL_Service::optimalOptimisationByMCIntegration().

8.34.2.2 void RMOL::Optimiser::optimalOptimisationByDP (stdair::LegCabin &) [static]

Dynamic Programming.

Definition at line 64 of file Optimiser.cpp.

References RMOL::DPOptimiser::optimalOptimisationByDP().

8.34.2.3 void RMOL::Optimiser::heuristicOptimisationByEmsr (stdair::LegCabin &) [static]

EMRS algorithm.

Definition at line 69 of file Optimiser.cpp.

References RMOL::Emsr::heuristicOptimisationByEmsr().

Referenced by RMOL::RMOL_Service::heuristicOptimisationByEmsr().

8.34.2.4 void RMOL::Optimiser::heuristicOptimisationByEmsrA (stdair::LegCabin &) [static]

EMRS-a algorithm.

Definition at line 74 of file Optimiser.cpp.

References RMOL::Emsr::heuristicOptimisationByEmsrA().

Referenced by RMOL::RMOL_Service::heuristicOptimisationByEmsrA().

8.34.2.5 void RMOL::Optimiser::heuristicOptimisationByEmsrB (stdair::LegCabin &) [static]

EMRS-b algorithm.

Definition at line 79 of file Optimiser.cpp.

References RMOL::Emsr::heuristicOptimisationByEmsrB().

Referenced by RMOL::RMOL_Service::heuristicOptimisationByEmsrB().

8.34.2.6 bool RMOL::Optimiser::optimise (stdair::FlightDate &, const stdair::OptimisationMethod &) [static]

Optimise a flight-date using leg-based Monte Carlo Integration.

Definition at line 84 of file Optimiser.cpp.

8.34.2.7 bool RMOL::Optimiser::buildVirtualClassListForLegBasedOptimisation (stdair::LegCabin &) [static]

Build the virtual class list for the given leg-cabin.

Definition at line 164 of file Optimiser.cpp.

8.34.2.8 double RMOL::Optimiser::optimiseUsingOnDForecast (stdair::FlightDate &, const bool & *iReduceFluctuations* = false) [static]

Optimiser

Definition at line 247 of file Optimiser.cpp.

References RMOL::MCOptimiser::optimisationByMCIntegration().

Referenced by RMOL::RMOL_Service::optimiseOnD(), RMOL::RMOL_Service::optimiseOnDUsingAdvancedRMCooperation(), and RMOL::RMOL_Service::optimiseOnDUsingRMCooperation().

The documentation for this class was generated from the following files:

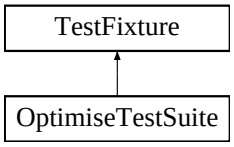
- rmol/command/Optimiser.hpp
- rmol/command/Optimiser.cpp

8.35 OptimiseTestSuite Class Reference

8.35 OptimiseTestSuite Class Reference

`#include <test/rmol/OptimiseTestSuite.hpp>`

Inheritance diagram for OptimiseTestSuite::



Public Member Functions

- void testOptimiseMC ()
- void testOptimiseDP ()
- void testOptimiseEMSR ()
- void testOptimiseEMSRa ()
- void testOptimiseEMSRb ()
- OptimiseTestSuite ()

Protected Attributes

- std::stringstream _describeKey

8.35.1 Detailed Description

Definition at line 6 of file OptimiseTestSuite.hpp.

8.35.2 Constructor & Destructor Documentation

8.35.2.1 OptimiseTestSuite::OptimiseTestSuite ()

Constructor.

8.35.3 Member Function Documentation

8.35.3.1 void OptimiseTestSuite::testOptimiseMC ()

Test the Monte-Carlo (MC) Optimisation functionality.

8.35.3.2 void OptimiseTestSuite::testOptimiseDP ()

Test the Dynamic Programming (DP) Optimisation functionality.

8.35.3.3 void OptimiseTestSuite::testOptimiseEMSR ()

Test the Expected Marginal Seat Revenue (EMSR) Optimisation functionality.

8.35.3.4 void OptimiseTestSuite::testOptimiseEMSRa ()

Test the Expected Marginal Seat Revenue, variant a (EMSR-a), Optimisation functionality.

8.35.3.5 void OptimiseTestSuite::testOptimiseEMSRb ()

Test the Expected Marginal Seat Revenue, variant b (EMSR-b), Optimisation functionality.

8.35.4 Member Data Documentation

8.35.4.1 std::stringstream OptimiseTestSuite::_describeKey [protected]

Definition at line 43 of file OptimiseTestSuite.hpp.

The documentation for this class was generated from the following file:

- test/rmol/OptimiseTestSuite.hpp

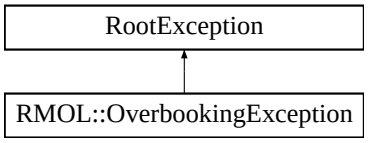
8.36 RMOL::OverbookingException Class Reference

8.36 RMOL::OverbookingException Class Reference

Overbooking-related exception.

`#include <rmol/RMOL_Types.hpp>`

Inheritance diagram for RMOL::OverbookingException::



Public Member Functions

- OverbookingException (const std::string &iWhat)

8.36.1 Detailed Description

Overbooking-related exception.

Definition at line 32 of file RMOL_Types.hpp.

8.36.2 Constructor & Destructor Documentation

8.36.2.1 RMOL::OverbookingException::OverbookingException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 35 of file RMOL_Types.hpp.

The documentation for this class was generated from the following file:

- rmol/RMOL_Types.hpp

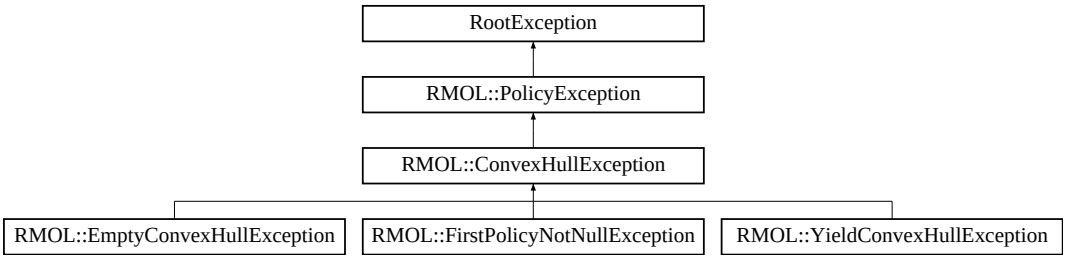
8.37 RMOL::PolicyException Class Reference

8.37 RMOL::PolicyException Class Reference

Policy-related exception.

`#include <rmol/RMOL_Types.hpp>`

Inheritance diagram for RMOL::PolicyException::



Public Member Functions

- PolicyException (const std::string &iWhat)

8.37.1 Detailed Description

Policy-related exception.

Definition at line 82 of file RMOL_Types.hpp.

8.37.2 Constructor & Destructor Documentation

8.37.2.1 RMOL::PolicyException::PolicyException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 85 of file RMOL_Types.hpp.

The documentation for this class was generated from the following file:

- rmol/RMOL_Types.hpp

8.38 RMOL::PolicyHelper Class Reference

8.38 RMOL::PolicyHelper Class Reference

`#include <rmol/bom/PolicyHelper.hpp>`

Static Public Member Functions

- static void diffBetweenTwoPolicies (stdair::NestingNode &, const stdair::Policy &, const stdair::Policy &)
- static void computeLastNode (stdair::NestingNode &, const stdair::Policy &, const stdair::SegmentCabin &)
- static bool isNested (const stdair::Policy &, const stdair::Policy &)

8.38.1 Detailed Description

Class holding helper methods.

Definition at line 28 of file PolicyHelper.hpp.

8.38.2 Member Function Documentation

8.38.2.1 void RMOL::PolicyHelper::diffBetweenTwoPolicies (stdair::NestingNode &, const stdair::Policy &, const stdair::Policy &) [static]

Find the booking class list representing the difference between two Policies (first minus second)

Definition at line 24 of file PolicyHelper.cpp.

8.38.2.2 void RMOL::PolicyHelper::computeLastNode (stdair::NestingNode &, const stdair::Policy &, const stdair::SegmentCabin &) [static]

Compute the list of the booking class which is not in the node.

Definition at line 164 of file PolicyHelper.cpp.

8.38.2.3 bool RMOL::PolicyHelper::isNested (const stdair::Policy &, const stdair::Policy &) [static]

Check if the first policy is nested under the second policy.

Definition at line 220 of file PolicyHelper.cpp.

The documentation for this class was generated from the following files:

- rmol/bom/PolicyHelper.hpp
- rmol/bom/PolicyHelper.cpp

8.39 RMOL::PreOptimiser Class Reference

8.39 RMOL::PreOptimiser Class Reference

`#include <rmol/command/PreOptimiser.hpp>`

Static Public Member Functions

- static bool preOptimise (stdair::FlightDate &, const stdair::PreOptimisationMethod &)

8.39.1 Detailed Description

Class wrapping the pre-optimisation algorithms.

Definition at line 22 of file PreOptimiser.hpp.

8.39.2 Member Function Documentation

8.39.2.1 bool RMOL::PreOptimiser::preOptimise (stdair::FlightDate &, const stdair::PreOptimisationMethod &) [static]

Prepare the demand input for the optimser.

Definition at line 30 of file PreOptimiser.cpp.

Referenced by RMOL::RMOL_Service::optimise().

The documentation for this class was generated from the following files:

- rmol/command/PreOptimiser.hpp
- rmol/command/PreOptimiser.cpp

8.40 RMOL::QForecasting Class Reference

8.40 RMOL::QForecasting Class Reference

`#include <rmol/command/QForecasting.hpp>`

Static Public Member Functions

- static bool forecast (stdair::SegmentCabin &, const stdair::Date_T &, const stdair::DTD_T &, const stdair::UnconstrainingMethod &, const stdair::NbOfSegments_T &)
- static void preparePriceOrientedHistoricalBooking (const stdair::SegmentCabin &, const stdair::SegmentSnapshotTable &, HistoricalBookingHolder &, const stdair::DCP_T &, const stdair::DCP_T &, const stdair::NbOfSegments_T &, const stdair::NbOfSegments_T &, const stdair::BookingClassSellUpCurveMap_T &)

8.40.1 Detailed Description

Class wrapping the optimisation algorithms.

Definition at line 23 of file QForecasting.hpp.

8.40.2 Member Function Documentation

8.40.2.1 bool RMOL::QForecasting::forecast (stdair::SegmentCabin &, const stdair::Date_T &, const stdair::DTD_T &, const stdair::UnconstrainingMethod &, const stdair::NbOfSegments_T &) [static]

Forecast demand for a flight-date.

Parameters:

const stdair::Date_T& Current Date

const stdair::NbOfSegments_T& Number of usable historical segments

Definition at line 31 of file QForecasting.cpp.

References RMOL::Utilities::computeDispatchingFactorCurves(), RMOL::Utilities::computeDistributionParameters(), RMOL::Utilities::computeSellUpFactorCurves(), RMOL::Utilities::dispatchDemandForecast(), RMOL::Utilities::dispatchDemandForecastForFA(), RMOL::HistoricalBookingHolder::getNbOfFlights(), RMOL::SegmentSnapshotTableHelper::getNbOfSegmentAlreadyPassedThisDTD(), RMOL::HistoricalBookingHolder::getUnconstrainedDemand(), preparePriceOrientedHistoricalBooking(), and RMOL::Detruncator::unconstrain().

Referenced by RMOL::HybridForecasting::forecast().

8.40.2.2 void RMOL::QForecasting::preparePriceOrientedHistoricalBooking (const stdair::SegmentCabin &, const stdair::SegmentSnapshotTable &, HistoricalBookingHolder &, const stdair::DCP_T &, const stdair::DCP_T &, const stdair::NbOfSegments_T &, const stdair::NbOfSegments_T &, const stdair::BookingClassSellUpCurveMap_T &) [static]

Prepare the historical price-oriented booking figures for a given cabin

Parameters:

const stdair::DCP_T& DCP range start

const stdair::DCP_T& DCP range end

const stdair::NbOfSegments_T& Segment range start index

const stdair::NbOfSegments_T& Segment range end index

Definition at line 136 of file QForecasting.cpp.

References RMOL::HistoricalBookingHolder::addHistoricalBooking().

Referenced by forecast().

The documentation for this class was generated from the following files:

- rmol/command/QForecasting.hpp
- rmol/command/QForecasting.cpp

8.41 RMOL::RMOL_Service Class Reference

8.41 RMOL::RMOL_Service Class Reference

Interface for the RMOL Services.

`#include <rmol/RMOL_Service.hpp>`

Public Member Functions

- RMOL_Service (const stdair::BasLogParams &, const stdair::BasDBParams &)
- RMOL_Service (const stdair::BasLogParams &)
- RMOL_Service (stdair::STDAIR_ServicePtr_T)
- void parseAndLoad (const stdair::CabinCapacity_T &iCabinCapacity, const stdair::Filename_T &iDemandAndClassDataFile)
- void setUpStudyStatManager ()
- ~RMOL_Service ()
- void buildSampleBom ()
- void clonePersistentBom ()
- void buildComplementaryLinks (stdair::BomRoot &)
- void optimalOptimisationByMCIntegration (const int K)
- void optimalOptimisationByDP ()
- void heuristicOptimisationByEmsr ()
- void heuristicOptimisationByEmsrA ()
- void heuristicOptimisationByEmsrB ()
- void heuristicOptimisationByMCIntegrationForQFF ()
- void heuristicOptimisationByEmsrBForQFF ()
- void MRTForNewQFF ()
- const stdair::SegmentCabin & retrieveDummySegmentCabin (const bool isForFareFamilies=false)
- bool optimise (stdair::FlightDate &, const stdair::DateTime_T &, const stdair::UnconstrainingMethod &, const stdair::ForecastingMethod &, const stdair::PreOptimisationMethod &, const stdair::OptimisationMethod &, const stdair::PartnershipTechnique &)
- void forecastOnD (const stdair::DateTime_T &)
- stdair::YieldFeatures * getYieldFeatures (const stdair::OnDDate &, const stdair::CabinCode_T &, stdair::BomRoot &)
- void forecastOnD (const stdair::YieldFeatures &, stdair::OnDDate &, const stdair::CabinCode_T &, const stdair::DTD_T &, stdair::BomRoot &)
- void setOnDForecast (const stdair::AirlineClassList &, const stdair::MeanValue_T &, const stdair::StdDevValue_T &, stdair::OnDDate &, const stdair::CabinCode_T &, stdair::BomRoot &)
- void setOnDForecast (const stdair::AirlineCode_T &, const stdair::Date_T &, const stdair::AirportCode_T &, const stdair::AirportCode_T &, const stdair::CabinCode_T &, const stdair::ClassCode_T &, const stdair::MeanValue_T &, const stdair::StdDevValue_T &, const stdair::Yield_T &, stdair::BomRoot &)
- void setOnDForecast (const stdair::AirlineCodeList_T &, const stdair::AirlineCode_T &, const stdair::Date_T &, const stdair::AirportCode_T &, const stdair::AirportCode_T &, const stdair::CabinCode_T &, const stdair::ClassCodeList_T &, const stdair::MeanValue_T &, const stdair::StdDevValue_T &, const stdair::Yield_T &, stdair::BomRoot &)
- void resetDemandInformation (const stdair::DateTime_T &)
- void resetDemandInformation (const stdair::DateTime_T &, const stdair::Inventory &)
- void projectAggregatedDemandOnLegCabins (const stdair::DateTime_T &)
- void projectOnDDemandOnLegCabinsUsingYP (const stdair::DateTime_T &)
- void projectOnDDemandOnLegCabinsUsingDA (const stdair::DateTime_T &)
- void projectOnDDemandOnLegCabinsUsingDYP (const stdair::DateTime_T &)
- void projectOnDDemandOnLegCabinsUsingDYP (const stdair::DateTime_T &, const stdair::Inventory &)
- void optimiseOnD (const stdair::DateTime_T &)
- void optimiseOnDUsingRMCooperation (const stdair::DateTime_T &)
- void optimiseOnDUsingAdvancedRMCooperation (const stdair::DateTime_T &)
- void updateBidPrice (const stdair::DateTime_T &)

- void updateBidPrice (const stdair::FlightDate &, stdair::BomRoot &)
- std::string jsonExport (const stdair::AirlineCode_T &, const stdair::FlightNumber_T &, const stdair::Date_T &iDepartureDate) const
- std::string csvDisplay () const

8.41.1 Detailed Description

Interface for the RMOL Services.

Definition at line 43 of file RMOL_Service.hpp.

8.41.2 Constructor & Destructor Documentation

8.41.2.1 RMOL::RMOL_Service::RMOL_Service (const stdair::BasLogParams &, const stdair::BasDBParams &)

Constructor.

The initRmolService() 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 a session can be created on the corresponding database.

Parameters:

const stdair::BasLogParams& Parameters for the output log stream.

const stdair::BasDBParams& Parameters for the database access.

Definition at line 85 of file RMOL_Service.cpp.

8.41.2.2 RMOL::RMOL_Service::RMOL_Service (const stdair::BasLogParams &)

Constructor.

The initRmolService() 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:

const stdair::BasLogParams& Parameters for the output log stream.

Definition at line 64 of file RMOL_Service.cpp.

8.41.2.3 RMOL::RMOL_Service::RMOL_Service (stdair::STDAIR_ServicePtr_T)

Constructor.

The initRmolService() method is called; see the corresponding documentation for more details.

Moreover, as no reference on any output stream is given, it is assumed that the StdAir log service has already been initialised with the proper log output stream by some other methods in the calling chain (for instance, when the RMOL_Service is itself being initialised by another library service such as AIRINV_-Service).

Parameters:

STDAIR_ServicePtr_T the shared pointer of stdair service.

Definition at line 107 of file RMOL_Service.cpp.

8.41.2.4 RMOL::RMOL_Service::~RMOL_Service ()

Destructor.

Definition at line 124 of file RMOL_Service.cpp.

8.41.3 Member Function Documentation

8.41.3.1 void RMOL::RMOL_Service::parseAndLoad (const stdair::CabinCapacity_T & *iCabinCapacity*, const stdair::Filename_T & *iDemandAndClassDataFile*)

Parse the optimisation-related data and load them into memory.

First, the STDAIR_Service::buildDummyInventory() method is called, for RMOL and with the given cabin capacity, in order to build the miminum required flight-date structure in order to perform an optimisation on a leg-cabin.

The CSV input file describes the problem to be optimised, i.e.:

- the demand specifications for all the booking classes (mean and standard deviations for the demand distribution); the yields corresponding to those booking classes.

That CSV file is parsed and instantiated in memory accordingly. The leg-cabin capacity has been set at the initialisation of the (RMOL) service.

Parameters:

const stdair::CabinCapacity& Capacity of the leg-cabin to be optimised.

const stdair::Filename_T& (CSV) input file.

Definition at line 201 of file RMOL_Service.cpp.

References buildComplementaryLinks(), clonePersistentBom(), RMOL::RMOL_ServiceContext::getOwnStdairServiceFlag(), RMOL::RMOL_ServiceContext::getSTDAIR_Service(), and RMOL::InventoryParser::parseInputFileAndBuildBom().

Referenced by main().

8.41.3.2 void RMOL::RMOL_Service::setUpStudyStatManager ()

Set up the StudyStatManager.

8.41.3.3 void RMOL::RMOL_Service::buildSampleBom ()

Build a sample BOM tree, and attach it to the BomRoot instance.

See also:

stdair::CmdBomManager::buildSampleBom() for more details.

Definition at line 260 of file RMOL_Service.cpp.

References buildComplementaryLinks(), clonePersistentBom(), RMOL::RMOL_ServiceContext::getOwnStdairServiceFlag(), and RMOL::RMOL_ServiceContext::getSTDAIR_Service().

Referenced by main().

8.41.3.4 void RMOL::RMOL_Service::clonePersistentBom ()

Clone the persistent BOM object.

Definition at line 319 of file RMOL_Service.cpp.

References buildComplementaryLinks(), RMOL::RMOL_ServiceContext::getOwnStdairServiceFlag(), and RMOL::RMOL_ServiceContext::getSTDAIR_Service().

Referenced by buildSampleBom(), and parseAndLoad().

8.41.3.5 void RMOL::RMOL_Service::buildComplementaryLinks (stdair::BomRoot &)

Build all the complementary links in the given bom root object. Build the links between dummy leg cabin and dummy segment cabin.

Definition at line 357 of file RMOL_Service.cpp.

References RMOL::RMOL_ServiceContext::getSTDAIR_Service().

Referenced by buildSampleBom(), clonePersistentBom(), and parseAndLoad().

8.41.3.6 void RMOL::RMOL_Service::optimalOptimisationByMCIntegration (const int *K*)

Single resource optimization using the Monte Carlo algorithm.

Definition at line 382 of file RMOL_Service.cpp.

References RMOL::RMOL_ServiceContext::getSTDAIR_Service(), and RMOL::Optimiser::optimalOptimisationByMCIntegration().

Referenced by optimise().

8.41.3.7 void RMOL::RMOL_Service::optimalOptimisationByDP ()

Single resource optimization using dynamic programming.

Definition at line 426 of file RMOL_Service.cpp.

Referenced by optimise().

8.41.3.8 void RMOL::RMOL_Service::heuristicOptimisationByEmsr ()

Single resource optimization using EMSR heuristic.

Definition at line 430 of file RMOL_Service.cpp.

References RMOL::RMOL_ServiceContext::getSTDAIR_Service(), and RMOL::Optimiser::heuristicOptimisationByEmsr().

Referenced by optimise().

8.41.3.9 void RMOL::RMOL_Service::heuristicOptimisationByEmsrA ()

Single resource optimization using EMSR-a heuristic.

Definition at line 475 of file RMOL_Service.cpp.

References RMOL::RMOL_ServiceContext::getSTDAIR_Service(), and RMOL::Optimiser::heuristicOptimisationByEmsrA().

Referenced by optimise().

8.41.3.10 void RMOL::RMOL_Service::heuristicOptimisationByEmsrB ()

Single resource optimization using EMSR-b heuristic.

Definition at line 500 of file RMOL_Service.cpp.

References RMOL::RMOL_ServiceContext::getSTDAIR_Service(), and RMOL::Optimiser::heuristicOptimisationByEmsrB().

Referenced by optimise().

8.41.3.11 void RMOL::RMOL_Service::heuristicOptimisationByMCIntegrationForQFF ()

Single resource optimization using the Monte Carlo algorithm for QFF method.

8.41.3.12 void RMOL::RMOL_Service::heuristicOptimisationByEmsrBForQFF ()

Single resource optimization using EMSR-b heuristic for QFF method.

8.41.3.13 void RMOL::RMOL_Service::MRTForNewQFF ()

Single resource pre-optimization using Marginal Revenue Transformation for QFF method.

8.41.3.14 const stdair::SegmentCabin & RMOL::RMOL_Service::retrieveDummySegmentCabin (const bool *isForFareFamilies* = `false`)

Retrieve one sample segment-cabin of the dummy inventory of "XX".

Parameters:

const bool Boolean to choose the sample segment-cabin. True: the dummy segment-cabin with fare families. False: the dummy segment-cabin without fare families. By default the value is false.

Definition at line 525 of file RMOL_Service.cpp.

References RMOL::RMOL_ServiceContext::getSTDAIR_Service().

8.41.3.15 bool RMOL::RMOL_Service::optimise (stdair::FlightDate &, const stdair::DateTime_T &, const stdair::UnconstrainingMethod &, const stdair::ForecastingMethod &, const stdair::PreOptimisationMethod &, const stdair::OptimisationMethod &, const stdair::PartnershipTechnique &)

Optimise (revenue management) an flight-date/network-date

Definition at line 547 of file RMOL_Service.cpp.

References RMOL::Forecaster::forecast(), forecastOnD(), optimise(), optimiseOnD(), optimiseOnDUsingAdvancedRMCooperation(), optimiseOnDUsingRMCooperation(), RMOL::PreOptimiser::preOptimise(), projectAggregatedDemandOnLegCabins(), projectOnDDemandOnLegCabinsUsingDYP(), projectOnDDemandOnLegCabinsUsingYP(), resetDemandInformation(), and updateBidPrice().

8.41.3.16 void RMOL::RMOL_Service::forecastOnD (const stdair::DateTime_T &)

Forecaster

Definition at line 648 of file RMOL_Service.cpp.

References RMOL::RMOL_ServiceContext::getSTDAIR_Service(), and getYieldFeatures().

Referenced by optimise().

8.41.3.17 stdair::YieldFeatures * RMOL::RMOL_Service::getYieldFeatures (const stdair::OnDDate &, const stdair::CabinCode_T &, stdair::BomRoot &)

Definition at line 723 of file RMOL_Service.cpp.

Referenced by forecastOnD().

8.41.3.18 void RMOL::RMOL_Service::forecastOnD (const stdair::YieldFeatures &, stdair::OnDDate &, const stdair::CabinCode_T &, const stdair::DTD_T &, stdair::BomRoot &)

Definition at line 796 of file RMOL_Service.cpp.

References setOnDForecast().

8.41.3.19 void RMOL::RMOL_Service::setOnDForecast (const stdair::AirlineClassList &, const stdair::MeanValue_T &, const stdair::StdDevValue_T &, stdair::OnDDate &, const stdair::CabinCode_T &, stdair::BomRoot &)

Definition at line 911 of file RMOL_Service.cpp.

Referenced by forecastOnD().

8.41.3.20 void RMOL::RMOL_Service::setOnDForecast (const stdair::AirlineCode_T &, const stdair::Date_T &, const stdair::AirportCode_T &, const stdair::AirportCode_T &, const stdair::CabinCode_T &, const stdair::ClassCode_T &, const stdair::MeanValue_T &, const stdair::StdDevValue_T &, const stdair::Yield_T &, stdair::BomRoot &)

Definition at line 970 of file RMOL_Service.cpp.

8.41.3.21 void RMOL::RMOL_Service::setOnDForecast (const stdair::AirlineCodeList_T &, const stdair::AirlineCode_T &, const stdair::Date_T &, const stdair::AirportCode_T &, const stdair::AirportCode_T &, const stdair::CabinCode_T &, const stdair::ClassCodeList_T &, const stdair::MeanValue_T &, const stdair::StdDevValue_T &, const stdair::Yield_T &, stdair::BomRoot &)

Definition at line 1034 of file RMOL_Service.cpp.

8.41.3.22 void RMOL::RMOL_Service::resetDemandInformation (const stdair::DateTime_T &)

Definition at line 1151 of file RMOL_Service.cpp.

References RMOL::RMOL_ServiceContext::getSTDAIR_Service().

Referenced by optimise(), optimiseOnDUsingAdvancedRMCooperation(), and optimiseOnDUsing-RMCooperation().

8.41.3.23 void RMOL::RMOL_Service::resetDemandInformation (const stdair::DateTime_T &, const stdair::Inventory &)

Definition at line 1177 of file RMOL_Service.cpp.

8.41.3.24 void RMOL::RMOL_Service::projectAggregatedDemandOnLegCabins (const stdair::DateTime_T &)

Definition at line 1227 of file RMOL_Service.cpp.

References RMOL::RMOL_ServiceContext::getSTDAIR_Service().

Referenced by optimise().

8.41.3.25 void RMOL::RMOL_Service::projectOnDDemandOnLegCabinsUsingYP (const stdair::DateTime_T &)

Definition at line 1332 of file RMOL_Service.cpp.

References RMOL::RMOL_ServiceContext::getSTDAIR_Service().

Referenced by optimise().

8.41.3.26 void RMOL::RMOL_Service::projectOnDDemandOnLegCabinsUsingDA (const stdair::DateTime_T &)

Definition at line 1609 of file RMOL_Service.cpp.

References RMOL::RMOL_ServiceContext::getSTDAIR_Service().

8.41.3.27 void RMOL::RMOL_Service::projectOnDDemandOnLegCabinsUsingDYP (const stdair::DateTime_T &)

Definition at line 1765 of file RMOL_Service.cpp.

References RMOL::RMOL_ServiceContext::getSTDAIR_Service().

Referenced by optimise(), optimiseOnDUsingAdvancedRMCooperation(), and optimiseOnDUsing-RMCooperation().

8.41.3.28 void RMOL::RMOL_Service::projectOnDDemandOnLegCabinsUsingDYP (const stdair::DateTime_T &, const stdair::Inventory &)

Definition at line 1791 of file RMOL_Service.cpp.

8.41.3.29 void RMOL::RMOL_Service::optimiseOnD (const stdair::DateTime_T &)

Optimiser

Definition at line 1431 of file RMOL_Service.cpp.

References RMOL::RMOL_ServiceContext::getSTDAIR_Service(), and RMOL::Optimiser::optimise-UsingOnDForecast().

Referenced by optimise().

8.41.3.30 void RMOL::RMOL_Service::optimiseOnDUsingRMCooperation (const stdair::Date-Time_T &)

Definition at line 1907 of file RMOL_Service.cpp.

References RMOL::RMOL_ServiceContext::getSTDAIR_Service(), RMOL::Optimiser::optimiseUsingOnDForecast(), projectOnDDemandOnLegCabinsUsingDYP(), and resetDemandInformation().

Referenced by optimise().

8.41.3.31 void RMOL::RMOL_Service::optimiseOnDUsingAdvancedRMCooperation (const stdair::DateTime_T &)

Definition at line 1967 of file RMOL_Service.cpp.

References RMOL::RMOL_ServiceContext::getSTDAIR_Service(), RMOL::Optimiser::optimiseUsingOnDForecast(), projectOnDDemandOnLegCabinsUsingDYP(), resetDemandInformation(), and updateBidPrice().

Referenced by optimise().

8.41.3.32 void RMOL::RMOL_Service::updateBidPrice (const stdair::DateTime_T &)

Definition at line 1480 of file RMOL_Service.cpp.

References RMOL::RMOL_ServiceContext::getSTDAIR_Service().

Referenced by optimise(), and optimiseOnDUsingAdvancedRMCooperation().

8.41.3.33 void RMOL::RMOL_Service::updateBidPrice (const stdair::FlightDate &, stdair::BomRoot &)

Definition at line 1528 of file RMOL_Service.cpp.

8.41.3.34 std::string RMOL::RMOL_Service::jsonExport (const stdair::AirlineCode_T &, const stdair::FlightNumber_T &, const stdair::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:

const stdair::AirlineCode_T& Airline code of the flight to dump.

const stdair::FlightNumber_T& Flight number of the flight to dump.

const stdair::Date_T& Departure date of a flight to dump.

Returns:

std::string Output string in which the BOM tree is JSON-ified.

8.41.3.35 std::string RMOL::RMOL_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.

The documentation for this class was generated from the following files:

- rmol/RMOL_Service.hpp
- rmol/service/RMOL_Service.cpp

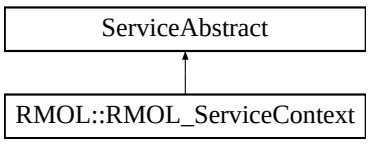
8.42 RMOL::RMOL_ServiceContext Class Reference

8.42 RMOL::RMOL_ServiceContext Class Reference

Inner class holding the context for the RMOL Service object.

`#include <rmol/service/RMOL_ServiceContext.hpp>`

Inheritance diagram for RMOL::RMOL_ServiceContext::



Friends

- class RMOL_Service
- class FacRmolServiceContext

8.42.1 Detailed Description

Inner class holding the context for the RMOL Service object.

Definition at line 29 of file RMOL_ServiceContext.hpp.

8.42.2 Friends And Related Function Documentation

8.42.2.1 friend class RMOL_Service [friend]

The RMOL_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 35 of file RMOL_ServiceContext.hpp.

8.42.2.2 friend class FacRmolServiceContext [friend]

Definition at line 36 of file RMOL_ServiceContext.hpp.

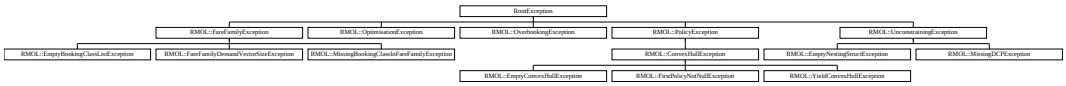
The documentation for this class was generated from the following files:

- rmol/service/RMOL_ServiceContext.hpp
- rmol/service/RMOL_ServiceContext.cpp

8.43 RootException Class Reference

8.43 RootException Class Reference

Inheritance diagram for RootException::



The documentation for this class was generated from the following file:

- rmol/RMOL_Types.hpp

8.44 RMOL::SegmentSnapshotTableHelper Class Reference

8.44 RMOL::SegmentSnapshotTableHelper Class Reference

`#include <rmol/bom/SegmentSnapshotTableHelper.hpp>`

Static Public Member Functions

- static stdair::NbOfSegments_T getNbOfSegmentAlreadyPassedThisDTD (const stdair::SegmentSnapshotTable &, const stdair::DTD_T &, const stdair::Date_T &)
- static bool hasPassedThisDTD (const stdair::SegmentCabin &, const stdair::DTD_T &, const stdair::Date_T &)

8.44.1 Detailed Description

Class representing the actual business functions for an airline guillotine block.

Definition at line 23 of file SegmentSnapshotTableHelper.hpp.

8.44.2 Member Function Documentation

8.44.2.1 stdair::NbOfSegments_T RMOL::SegmentSnapshotTableHelper::getNbOfSegmentAlreadyPassedThisDTD (const stdair::SegmentSnapshotTable &, const stdair::DTD_T &, const stdair::Date_T &) [static]

Retrieve the number of similar segments which already passed the given DTD.

Definition at line 20 of file SegmentSnapshotTableHelper.cpp.

References hasPassedThisDTD().

Referenced by RMOL::QForecasting::forecast(), RMOL::OldQFF::forecast(), RMOL::HybridForecasting::forecast(), RMOL::BasedForecasting::forecast(), and RMOL::Utilities::getNbOfDepartedSimilarSegments().

8.44.2.2 bool RMOL::SegmentSnapshotTableHelper::hasPassedThisDTD (const stdair::SegmentCabin &, const stdair::DTD_T &, const stdair::Date_T &) [static]

Check if the given segment has passed the given DTD.

Definition at line 42 of file SegmentSnapshotTableHelper.cpp.

Referenced by getNbOfSegmentAlreadyPassedThisDTD().

The documentation for this class was generated from the following files:

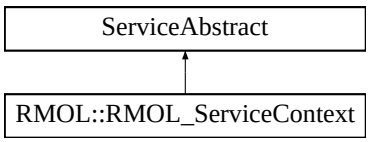
- rmol/bom/SegmentSnapshotTableHelper.hpp

- rmol/bom/SegmentSnapshotTableHelper.cpp

8.45 ServiceAbstract Class Reference

8.45 ServiceAbstract Class Reference

Inheritance diagram for ServiceAbstract::



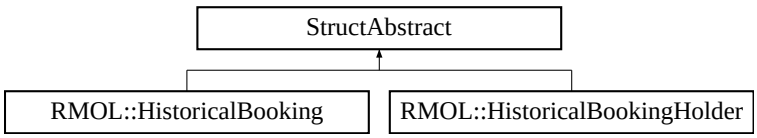
The documentation for this class was generated from the following file:

- rmol/service/RMOL_ServiceContext.hpp

8.46 StructAbstract Class Reference

8.46 StructAbstract Class Reference

Inheritance diagram for StructAbstract::



The documentation for this class was generated from the following files:

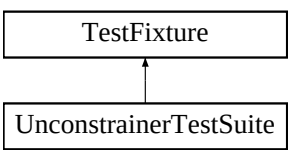
- rmol/bom/HistoricalBooking.hpp
- rmol/bom/HistoricalBookingHolder.hpp

8.47 UnconstrainerTestSuite Class Reference

8.47 UnconstrainerTestSuite Class Reference

`#include <test/rmol/UnconstrainerTestSuite.hpp>`

Inheritance diagram for UnconstrainerTestSuite::



Public Member Functions

- void testUnconstrainingByEM ()
- UnconstrainerTestSuite ()

Protected Attributes

- std::stringstream _describeKey

8.47.1 Detailed Description

Definition at line 6 of file UnconstrainerTestSuite.hpp.

8.47.2 Constructor & Destructor Documentation

8.47.2.1 UnconstrainerTestSuite::UnconstrainerTestSuite ()

Constructor.

8.47.3 Member Function Documentation

8.47.3.1 void UnconstrainerTestSuite::testUnconstrainingByEM ()

Test data unconstraining by Expectation Maximization.

8.47.4 Member Data Documentation

8.47.4.1 std::stringstream UnconstrainerTestSuite::_describeKey [protected]

Definition at line 19 of file UnconstrainerTestSuite.hpp.

The documentation for this class was generated from the following file:

- test/rmol/UnconstrainerTestSuite.hpp

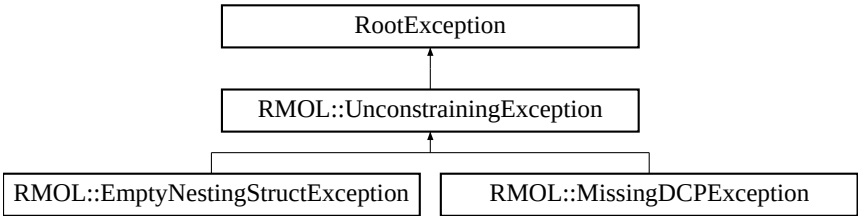
8.48 RMOL::UnconstrainingException Class Reference

8.48 RMOL::UnconstrainingException Class Reference

Unconstraining-related exception.

`#include <rmol/RMOL_Types.hpp>`

Inheritance diagram for RMOL::UnconstrainingException::



Public Member Functions

- UnconstrainingException (const std::string &iWhat)

8.48.1 Detailed Description

Unconstraining-related exception.

Definition at line 42 of file RMOL_Types.hpp.

8.48.2 Constructor & Destructor Documentation

8.48.2.1 RMOL::UnconstrainingException::UnconstrainingException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 45 of file RMOL_Types.hpp.

The documentation for this class was generated from the following file:

- rmol/RMOL_Types.hpp

8.49 RMOL::Utilities Class Reference

8.49 RMOL::Utilities Class Reference

`#include <rmol/bom/Utilities.hpp>`

Static Public Member Functions

- static void computeDistributionParameters (const stdair::UncDemVector_T &, stdair::MeanValue_T &, stdair::StdDevValue_T &)
- static stdair::DCPList_T buildRemainingDCPList (const stdair::DTD_T &)
- static stdair::DCPList_T buildPastDCPList (const stdair::DTD_T &)
- static stdair::NbOfSegments_T getNbOfDepartedSimilarSegments (const stdair::SegmentCabin &, const stdair::Date_T &)
- static stdair::BookingClassSellUpCurveMap_T computeSellUpFactorCurves (const stdair::FRAT5Curve_T &, const stdair::BookingClassList_T &)
- static stdair::BookingClassDispatchingCurveMap_T computeDispatchingFactorCurves (const stdair::FRAT5Curve_T &, const stdair::BookingClassList_T &)
- static void dispatchDemandForecast (const stdair::BookingClassDispatchingCurveMap_T &, const stdair::MeanValue_T &, const stdair::StdDevValue_T &, const stdair::DTD_T &)
- static void dispatchDemandForecastForFA (const stdair::BookingClassSellUpCurveMap_T &, const stdair::MeanValue_T &, const stdair::StdDevValue_T &, const stdair::DTD_T &)

8.49.1 Detailed Description

Class holding helper methods.

Definition at line 20 of file Utilities.hpp.

8.49.2 Member Function Documentation

8.49.2.1 void RMOL::Utilities::computeDistributionParameters (const stdair::UncDemVector_T &, stdair::MeanValue_T &, stdair::StdDevValue_T &) [static]

Compute the mean and the standard deviation from a set of samples.

Definition at line 27 of file Utilities.cpp.

Referenced by RMOL::QForecasting::forecast(), RMOL::OldQFF::forecast(), RMOL::HybridForecasting::forecast(), and RMOL::BasedForecasting::forecast().

8.49.2.2 stdair::DCPList_T RMOL::Utilities::buildRemainingDCPList (const stdair::DTD_T &) [static]

Build the list of remaining DCP's for the segment-date.

Definition at line 59 of file Utilities.cpp.

8.49.2.3 stdair::DCPList_T RMOL::Utilities::buildPastDCPList (const stdair::DTD_T &) [static]

Build the list of past DCP's for the segment-date.

Definition at line 84 of file Utilities.cpp.

8.49.2.4 stdair::NbOfSegments_T RMOL::Utilities::getNbOfDepartedSimilarSegments (const stdair::SegmentCabin &, const stdair::Date_T &) [static]

Retrieve the number of departed similar segments.

Definition at line 104 of file Utilities.cpp.

References RMOL::SegmentSnapshotTableHelper::getNbOfSegmentAlreadyPassedThisDTD().

8.49.2.5 stdair::BookingClassSellUpCurveMap_T RMOL::Utilities::computeSellUpFactorCurves (const stdair::FRAT5Curve_T &, const stdair::BookingClassList_T &) [static]

Precompute the sell-up factors for each class and each DCP.

Definition at line 116 of file Utilities.cpp.

Referenced by RMOL::QForecasting::forecast(), and RMOL::OldQFF::forecast().

8.49.2.6 stdair::BookingClassDispatchingCurveMap_T RMOL::Utilities::computeDispatchingFactorCurves (const stdair::FRAT5Curve_T &, const stdair::BookingClassList_T &) [static]

Precompute the dispatching factors for each class and each DCP.

Definition at line 177 of file Utilities.cpp.

Referenced by RMOL::QForecasting::forecast().

8.49.2.7 void RMOL::Utilities::dispatchDemandForecast (const stdair::BookingClassDispatchingCurveMap_T &, const stdair::MeanValue_T &, const stdair::StdDevValue_T &, const stdair::DTD_T &) [static]

Dispatching the demand forecast to all classes.

Definition at line 253 of file Utilities.cpp.

Referenced by RMOL::QForecasting::forecast().

8.49.2.8 void RMOL::Utilities::dispatchDemandForecastForFA (const stdair::BookingClassSellUpCurveMap_T &, const stdair::MeanValue_T &, const stdair::StdDevValue_T &, const stdair::DTD_T &) [static]

Dispatching the demand forecast to all classes for FA.

Definition at line 286 of file Utilities.cpp.

Referenced by RMOL::QForecasting::forecast().

The documentation for this class was generated from the following files:

- rmol/bom/Utilities.hpp
- rmol/bom/Utilities.cpp

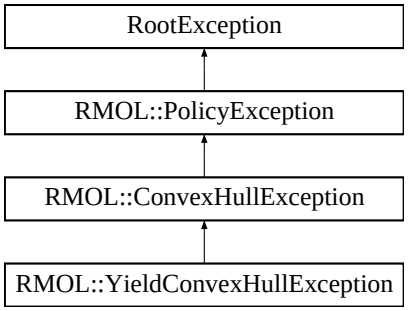
8.50 RMOL::YieldConvexHullException Class Reference

8.50 RMOL::YieldConvexHullException Class Reference

Yield convex hull exception.

`#include <rmol/RMOL_Types.hpp>`

Inheritance diagram for RMOL::YieldConvexHullException::



Public Member Functions

- YieldConvexHullException (const std::string &iWhat)

8.50.1 Detailed Description

Yield convex hull exception.

Definition at line 123 of file RMOL_Types.hpp.

8.50.2 Constructor & Destructor Documentation

8.50.2.1 RMOL::YieldConvexHullException::YieldConvexHullException (const std::string & *i-What*) [inline]

Constructor.

Definition at line 126 of file RMOL_Types.hpp.

The documentation for this class was generated from the following file:

- rmol/RMOL_Types.hpp

9 RMOL File Documentation

9 RMOL File Documentation

9.1 doc/local/authors.doc File Reference

9.1 doc/local/authors.doc File Reference

9.2 doc/local/codingrules.doc File Reference

9.2 doc/local/codingrules.doc File Reference

9.3 doc/local/copyright.doc File Reference

9.3 doc/local/copyright.doc File Reference

9.4 doc/local/documentation.doc File Reference

9.4 doc/local/documentation.doc File Reference

9.5 doc/local/features.doc File Reference

9.5 doc/local/features.doc File Reference

9.6 doc/local/help_wanted.doc File Reference

9.6 doc/local/help_wanted.doc File Reference

9.7 doc/local/howto_release.doc File Reference

9.7 doc/local/howto_release.doc File Reference

9.8 doc/local/index.doc File Reference

9.8 doc/local/index.doc File Reference

9.9 doc/local/installation.doc File Reference

9.9 doc/local/installation.doc File Reference

9.10 doc/local/linking.doc File Reference

9.10 doc/local/linking.doc File Reference

9.11 doc/local/test.doc File Reference

9.11 doc/local/test.doc File Reference

9.12 doc/local/users_guide.doc File Reference

9.12 doc/local/users_guide.doc File Reference

9.13 doc/local/verification.doc File Reference

9.13 doc/local/verification.doc File Reference

9.14 doc/tutorial/tutorial.doc File Reference

9.14 doc/tutorial/tutorial.doc File Reference

9.15 rmol/basic/BasConst.cpp File Reference

9.15 rmol/basic/BasConst.cpp File Reference

```
#include <rmol/basic/BasConst_General.hpp>
#include <rmol/basic/BasConst_RMOL_Service.hpp>
```

Namespaces

- namespace RMOL

Variables

- const stdair::AirlineCode_T RMOL::DEFAULT_RMOL_SERVICE_AIRLINE_CODE = "BA"
- const double RMOL::DEFAULT_RMOL_SERVICE_CAPACITY = 1.0
- const int RMOL::DEFAULT_NUMBER_OF_DRAWS_FOR_MC_SIMULATION = 10000
- const int RMOL::DEFAULT_PRECISION = 10
- const double RMOL::DEFAULT_EPSILON = 0.0001
- const double RMOL::DEFAULT_STOPPING_CRITERION = 0.01
- const double RMOL::DEFAULT_INITIALIZER_DOUBLE_NEGATIVE = -10.0

9.16 rmol/basic/BasConst_General.hpp File Reference

9.16 rmol/basic/BasConst_General.hpp File Reference

```
#include <stdair/stdair_types.hpp>
```

Namespaces

- namespace RMOL

Variables

- const int RMOL::DEFAULT_NUMBER_OF_DRAWS_FOR_MC_SIMULATION
- const int RMOL::DEFAULT_PRECISION
- const double RMOL::DEFAULT_EPSILON
- const double RMOL::DEFAULT_STOPPING_CRITERION
- const double RMOL::DEFAULT_INITIALIZER_DOUBLE_NEGATIVE

9.17 rmol/basic/BasConst_RMOL_Service.hpp File Reference

9.17 rmol/basic/BasConst_RMOL_Service.hpp File Reference

```
#include <vector>
#include <stdair/stdair_basic_types.hpp>
#include <rmol/RMOL_Types.hpp>
```

Namespaces

- namespace RMOL

Variables

- const stdair::AirlineCode_T RMOL::DEFAULT_RMOL_SERVICE_AIRLINE_CODE
- const double RMOL::DEFAULT_RMOL_SERVICE_CAPACITY

9.18 rmol/batches/rmol.cpp File Reference

9.18 rmol/batches/rmol.cpp File Reference

```
#include <cassert>
#include <iostream>
#include <sstream>
#include <fstream>
#include <string>
#include <boost/date_time/posix_time/posix_time.hpp>
#include <boost/date_time/gregorian/gregorian.hpp>
#include <boost/program_options.hpp>
#include <stdair/service/Logger.hpp>
#include <rmol/basic/BasConst_General.hpp>
#include <rmol/RMOL_Service.hpp>
#include <rmol/config/rmol-paths.hpp>
```

Functions

- const std::string K_RMOL_DEFAULT_LOG_FILENAME ("rmol.log")

- const std::string K_RMOL_DEFAULT_INPUT_FILENAME (STDAIR_SAMPLE_-DIR"/rm01.csv")
- template<class T> std::ostream & operator<< (std::ostream &os, const std::vector< T > &v)
- int readConfiguration (int argc, char *argv[ ], int &ioRandomDraws, double &ioCapacity, short &io-Method, bool &ioIsBuiltin, std::string &ioInputFilename, std::string &ioLogFilename)
- void optimise (RMOL::RMOL_Service &rmolService, const short &iMethod, const int &iRandom-Draws)
- int main (int argc, char *argv[ ])

Variables

- const bool K_RMOL_DEFAULT_BUILT_IN_INPUT = false
- const int K_RMOL_DEFAULT_RANDOM_DRAWS = RMOL::DEFAULT_NUMBER_OF_-DRAWS_FOR_MC_SIMULATION
- const double K_RMOL_DEFAULT_CAPACITY = 500.0
- const short K_RMOL_DEFAULT_METHOD = 0
- const int K_RMOL_EARLY_RETURN_STATUS = 99

9.18.1 Function Documentation

9.18.1.1 const std::string K_RMOL_DEFAULT_LOG_FILENAME ("rmol.log")

Default name and location for the log file.

Referenced by readConfiguration().

9.18.1.2 const std::string K_RMOL_DEFAULT_INPUT_FILENAME (STDAIR_SAMPLE_-DIR"/rm01.csv")

Default name and location for the (CSV) input file.

Referenced by readConfiguration().

9.18.1.3 template<class T> std::ostream& operator<< (std::ostream & *os*, const std::vector< T > & *v*)

Definition at line 48 of file rmol.cpp.

9.18.1.4 int readConfiguration (int *argc*, char * *argv*[ ], int & *ioRandomDraws*, double & *ioCapacity*, short & *ioMethod*, bool & *ioIsBuiltin*, std::string & *ioInputFilename*, std::string & *ioLogFilename*)

Read and parse the command line options.

Definition at line 58 of file rmol.cpp.

References K_RMOL_DEFAULT_BUILT_IN_INPUT, K_RMOL_DEFAULT_CAPACITY, K_RMOL_-DEFAULT_INPUT_FILENAME(), K_RMOL_DEFAULT_LOG_FILENAME(), K_RMOL_DEFAULT_-METHOD, K_RMOL_DEFAULT_RANDOM_DRAWS, and K_RMOL_EARLY_RETURN_STATUS.

Referenced by main().

9.18.1.5 void optimise (RMOL::RMOL_Service & *rmolService*, const short & *iMethod*, const int & *iRandomDraws*)

Definition at line 168 of file rmol.cpp.

References RMOL::RMOL_Service::heuristicOptimisationByEmsr(), RMOL::RMOL_Service::heuristicOptimisationByEmsrA(), RMOL::RMOL_Service::heuristicOptimisationByEmsrB(), RMOL::RMOL_Service::optimalOptimisationByDP(), and RMOL::RMOL_Service::optimalOptimisationByMCIntegration().

Referenced by main(), and RMOL::RMOL_Service::optimise().

9.18.1.6 int main (int *argc*, char * *argv*[ ])

Definition at line 205 of file rmol.cpp.

References RMOL::RMOL_Service::buildSampleBom(), K_RMOL_EARLY_RETURN_STATUS, optimise(), RMOL::RMOL_Service::parseAndLoad(), and readConfiguration().

9.18.2 Variable Documentation

9.18.2.1 const bool K_RMOL_DEFAULT_BUILT_IN_INPUT = false

Default for the input type. It can be either built-in or provided by an input file. That latter must then be given with the -i/–input option.

Definition at line 24 of file rmol.cpp.

Referenced by readConfiguration().

9.18.2.2 const int K_RMOL_DEFAULT_RANDOM_DRAWS = RMOL::DEFAULT_NUMBER_OF_DRAWS_FOR_MC_SIMULATION

Default number of random draws to be generated (best if over 100).

Definition at line 30 of file rmol.cpp.

Referenced by readConfiguration().

9.18.2.3 const double K_RMOL_DEFAULT_CAPACITY = 500.0

Default value for the capacity of the resource (e.g., a flight cabin).

Definition at line 33 of file rmol.cpp.

Referenced by readConfiguration().

9.18.2.4 const short K_RMOL_DEFAULT_METHOD = 0

Default name and location for the Revenue Management method to be used.

- 0 = Monte-Carlo
- 1 = Dynamic Programming
- 2 = EMSR
- 3 = EMSR-a

- 4 = EMSR-b

Definition at line 44 of file rmol.cpp.

Referenced by readConfiguration().

9.18.2.5 const int K_RMOL_EARLY_RETURN_STATUS = 99

Early return status (so that it can be differentiated from an error).

Definition at line 55 of file rmol.cpp.

Referenced by main(), and readConfiguration().

9.19 rmol/bom/BucketHolderTypes.hpp File Reference

9.19 rmol/bom/BucketHolderTypes.hpp File Reference

```
#include <list>
#include <map>
#include <stdair/stdair_basic_types.hpp>
```

Namespaces

- namespace RMOL

Typedefs

- typedef std::list< BucketHolder * > RMOL::BucketHolderList_T

9.20 rmol/bom/DistributionParameterList.hpp File Reference

9.20 rmol/bom/DistributionParameterList.hpp File Reference

```
#include <list>
#include <rmol/field/FldDistributionParameters.hpp>
```

Namespaces

- namespace RMOL

Typedefs

- typedef std::list< FldDistributionParameters > RMOL::DistributionParameterList_T

9.21 rmol/bom/DPOptimiser.cpp File Reference

9.21 rmol/bom/DPOptimiser.cpp File Reference

```
#include <cassert>
```

```
#include <sstream>
#include <vector>
#include <cmath>
#include <boost/math/distributions/normal.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/VirtualClassStruct.hpp>
#include <stdair/service/Logger.hpp>
#include <rmol/basic/BasConst_General.hpp>
#include <rmol/bom/DPOptimiser.hpp>
```

Namespaces

- namespace RMOL

9.22 rmol/bom/DPOptimiser.hpp File Reference

9.22 rmol/bom/DPOptimiser.hpp File Reference

```
#include <rmol/RMOL_Types.hpp>
```

Namespaces

- namespace stdair
- namespace RMOL

Classes

- class RMOL::DPOptimiser

9.23 rmol/bom/EMDetruncator.cpp File Reference

9.23 rmol/bom/EMDetruncator.cpp File Reference

```
#include <iostream>
#include <cmath>
#include <vector>
#include <cassert>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/service/Logger.hpp>
#include <rmol/bom/HistoricalBookingHolder.hpp>
#include <rmol/bom/EMDetruncator.hpp>
```

Namespaces

- namespace RMOL

9.24 rmol/bom/EMDetruncator.hpp File Reference

9.24 rmol/bom/EMDetruncator.hpp File Reference

Namespaces

- namespace RMOL

Classes

- class RMOL::EMDetruncator

9.25 rmol/bom/Emsr.cpp File Reference

9.25 rmol/bom/Emsr.cpp File Reference

```
#include <assert.h>
#include <iostream>
#include <cmath>
#include <list>
#include <algorithm>
#include <stdair/stdair_rm_types.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/VirtualClassStruct.hpp>
#include <rmol/bom/Emsr.hpp>
#include <rmol/bom/EmsrUtils.hpp>
```

Namespaces

- namespace RMOL

9.26 rmol/bom/Emsr.hpp File Reference

9.26 rmol/bom/Emsr.hpp File Reference

```
#include <rmol/RMOL_Types.hpp>
```

Namespaces

- namespace stdair
- namespace RMOL

Classes

- class RMOL::Emsr

9.27 rmol/bom/EmsrUtils.cpp File Reference

9.27 rmol/bom/EmsrUtils.cpp File Reference

```
#include <cassert>
#include <cmath>
#include <boost/math/distributions/normal.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/bom/VirtualClassStruct.hpp>
#include <rmol/bom/EmsrUtils.hpp>
#include <rmol/basic/BasConst_General.hpp>
```

Namespaces

- namespace RMOL

9.28 rmol/bom/EmsrUtils.hpp File Reference

9.28 rmol/bom/EmsrUtils.hpp File Reference

```
#include <stdair/stdair_inventory_types.hpp>
```

Namespaces

- namespace stdair
- namespace RMOL

Classes

- class RMOL::EmsrUtils

9.29 rmol/bom/HistoricalBooking.cpp File Reference

9.29 rmol/bom/HistoricalBooking.cpp File Reference

```
#include <sstream>
#include <cassert>
#include <iomanip>
#include <iostream>
#include <rmol/bom/HistoricalBooking.hpp>
```

Namespaces

- namespace RMOL

9.30 rmol/bom/HistoricalBooking.hpp File Reference

9.30 rmol/bom/HistoricalBooking.hpp File Reference

```
#include <stdair/stdair_basic_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
```

Namespaces

- namespace RMOL

Classes

- struct RMOL::HistoricalBooking

  Structure keeping track, for a given class, of the number of historical bookings and of the censorship flag.

9.31 rmol/bom/HistoricalBookingHolder.cpp File Reference

9.31 rmol/bom/HistoricalBookingHolder.cpp File Reference

```
#include <sstream>
#include <iostream>
#include <iomanip>
#include <cmath>
#include <cassert>
#include <stdair/service/Logger.hpp>
#include <rmol/bom/HistoricalBooking.hpp>
#include <rmol/bom/HistoricalBookingHolder.hpp>
```

Namespaces

- namespace RMOL

9.32 rmol/bom/HistoricalBookingHolder.hpp File Reference

9.32 rmol/bom/HistoricalBookingHolder.hpp File Reference

```
#include <iostream>
#include <vector>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
```

Namespaces

- namespace RMOL

Classes

- struct RMOL::HistoricalBookingHolder

Typedefs

- typedef std::vector< HistoricalBooking > RMOL::HistoricalBookingVector_T

9.33 rmol/bom/MCOptimiser.cpp File Reference

9.33 rmol/bom/MCOptimiser.cpp File Reference

```
#include <cassert>
#include <string>
#include <sstream>
#include <algorithm>
#include <cmath>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/VirtualClassStruct.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/basic/RandomGeneration.hpp>
#include <stdair/basic/BasConst_General.hpp>
#include <rmol/basic/BasConst_General.hpp>
#include <rmol/bom/MCOptimiser.hpp>
```

Namespaces

- namespace RMOL

9.34 rmol/bom/MCOptimiser.hpp File Reference

9.34 rmol/bom/MCOptimiser.hpp File Reference

```
#include <rmol/RMOL_Types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/stdair_rm_types.hpp>
```

Namespaces

- namespace stdair
- namespace RMOL

Classes

- class RMOL::MCOptimiser

9.35 rmol/bom/old/DemandGeneratorList.cpp File Reference

9.35 rmol/bom/old/DemandGeneratorList.cpp File Reference

```
#include <rmol/bom/DemandGeneratorList.hpp>
```

Namespaces

- namespace RMOL

9.36 rmol/bom/old/DemandGeneratorList.hpp File Reference

9.36 rmol/bom/old/DemandGeneratorList.hpp File Reference

```
#include <list>
#include <rmol/bom/VariateList.hpp>
#include <rmol/bom/DistributionParameterList.hpp>
#include <rmol/bom/Gaussian.hpp>
```

Namespaces

- namespace RMOL

Classes

- class RMOL::DemandGeneratorList

9.37 rmol/bom/PolicyHelper.cpp File Reference

9.37 rmol/bom/PolicyHelper.cpp File Reference

```
#include <cassert>
#include <cmath>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_Yield.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/Policy.hpp>
```

```
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <stdair/bom/NestingNode.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/factory/FacBomManager.hpp>
#include <rmol/bom/PolicyHelper.hpp>
```

Namespaces

- namespace RMOL

9.38 rmol/bom/PolicyHelper.hpp File Reference

9.38 rmol/bom/PolicyHelper.hpp File Reference

```
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/PolicyTypes.hpp>
#include <stdair/bom/BookingClassTypes.hpp>
#include <stdair/bom/FareFamilyTypes.hpp>
#include <rmol/RMOL_Types.hpp>
```

Namespaces

- namespace stdair
- namespace RMOL

Classes

- class RMOL::PolicyHelper

9.39 rmol/bom/SegmentSnapshotTableHelper.cpp File Reference

9.39 rmol/bom/SegmentSnapshotTableHelper.cpp File Reference

```
#include <cassert>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/SegmentSnapshotTable.hpp>
#include <stdair/service/Logger.hpp>
#include <rmol/bom/SegmentSnapshotTableHelper.hpp>
```

Namespaces

- namespace RMOL

9.40 rmol/bom/SegmentSnapshotTableHelper.hpp File Reference

9.40 rmol/bom/SegmentSnapshotTableHelper.hpp File Reference

```
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
```

Namespaces

- namespace stdair
- namespace RMOL

Classes

- class RMOL::SegmentSnapshotTableHelper

9.41 rmol/bom/Utilities.cpp File Reference

9.41 rmol/bom/Utilities.cpp File Reference

```
#include <cassert>
#include <string>
#include <numeric>
#include <algorithm>
#include <cmath>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/BookingClassTypes.hpp>
#include <stdair/service/Logger.hpp>
#include <rmol/basic/BasConst_General.hpp>
#include <rmol/bom/Utilities.hpp>
#include <rmol/bom/SegmentSnapshotTableHelper.hpp>
```

Namespaces

- namespace RMOL

9.42 rmol/bom/Utilities.hpp File Reference

9.42 rmol/bom/Utilities.hpp File Reference

```
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/FareFamilyTypes.hpp>
#include <rmol/RMOL_Types.hpp>
```

Namespaces

- namespace stdair
- namespace RMOL

Classes

- class RMOL::Utilities

9.43 rmol/command/BasedForecasting.cpp File Reference

9.43 rmol/command/BasedForecasting.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <cmath>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/RandomGeneration.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/SegmentSnapshotTable.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/service/Logger.hpp>
#include <rmol/bom/Utilities.hpp>
#include <rmol/bom/SegmentSnapshotTableHelper.hpp>
#include <rmol/bom/HistoricalBookingHolder.hpp>
#include <rmol/bom/HistoricalBooking.hpp>
#include <rmol/command/BasedForecasting.hpp>
#include <rmol/command/Detruncator.hpp>
```

Namespaces

- namespace RMOL

9.44 rmol/command/BasedForecasting.hpp File Reference

9.44 rmol/command/BasedForecasting.hpp File Reference

```
#include <stdair/stdair_inventory_types.hpp>
#include <rmol/RMOL_Types.hpp>
```

Namespaces

- namespace stdair
- namespace RMOL

Classes

- class RMOL::BasedForecasting

9.45 rmol/command/DemandInputPreparation.cpp File Reference

9.45 rmol/command/DemandInputPreparation.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <cmath>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/service/Logger.hpp>
#include <rmol/bom/Utilities.hpp>
#include <rmol/command/DemandInputPreparation.hpp>
```

Namespaces

- namespace RMOL

9.46 rmol/command/DemandInputPreparation.hpp File Reference

9.46 rmol/command/DemandInputPreparation.hpp File Reference

```
#include <map>
```

```
#include <stdair/stdair_inventory_types.hpp>
#include <rmol/RMOL_Types.hpp>
```

Namespaces

- namespace stdair
- namespace RMOL

Classes

- class RMOL::DemandInputPreparation

9.47 rmol/command/Detruncator.cpp File Reference

9.47 rmol/command/Detruncator.cpp File Reference

```
#include <cassert>
#include <stdair/basic/UnconstrainingMethod.hpp>
#include <stdair/service/Logger.hpp>
#include <rmol/bom/HistoricalBookingHolder.hpp>
#include <rmol/bom/EMDetruncator.hpp>
#include <rmol/command/Detruncator.hpp>
```

Namespaces

- namespace RMOL

9.48 rmol/command/Detruncator.hpp File Reference

9.48 rmol/command/Detruncator.hpp File Reference

```
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/UnconstrainingMethod.hpp>
#include <rmol/RMOL_Types.hpp>
```

Namespaces

- namespace RMOL

Classes

- class RMOL::Detruncator

9.49 rmol/command/FareAdjustment.cpp File Reference

9.49 rmol/command/FareAdjustment.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <cmath>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/service/Logger.hpp>
#include <rmol/bom/Utilities.hpp>
#include <rmol/command/FareAdjustment.hpp>
```

Namespaces

- namespace RMOL

9.50 rmol/command/FareAdjustment.hpp File Reference

9.50 rmol/command/FareAdjustment.hpp File Reference

```
#include <map>
#include <stdair/stdair_inventory_types.hpp>
#include <rmol/RMOL_Types.hpp>
```

Namespaces

- namespace stdair
- namespace RMOL

Classes

- class RMOL::FareAdjustment

9.51 rmol/command/Forecaster.cpp File Reference

9.51 rmol/command/Forecaster.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <cmath>
#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/SegmentDate.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/SegmentSnapshotTable.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/service/Logger.hpp>
#include <rmol/bom/Utilities.hpp>
#include <rmol/bom/SegmentSnapshotTableHelper.hpp>
#include <rmol/bom/HistoricalBookingHolder.hpp>
#include <rmol/bom/HistoricalBooking.hpp>
#include <rmol/command/BasedForecasting.hpp>
#include <rmol/command/Forecaster.hpp>
#include <rmol/command/QForecasting.hpp>
#include <rmol/command/HybridForecasting.hpp>
#include <rmol/command/OldQFF.hpp>
#include <rmol/command/NewQFF.hpp>
```

Namespaces

- namespace RMOL

9.52 rmol/command/Forecaster.hpp File Reference

9.52 rmol/command/Forecaster.hpp File Reference

```
#include <map>
#include <stdair/stdair_inventory_types.hpp>
#include <rmol/RMOL_Types.hpp>
```

Namespaces

- namespace stdair
- namespace RMOL

Classes

- class RMOL::Forecaster

9.53 rmol/command/HybridForecasting.cpp File Reference

9.53 rmol/command/HybridForecasting.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <cmath>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/RandomGeneration.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/SegmentSnapshotTable.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/service/Logger.hpp>
#include <rmol/bom/Utilities.hpp>
#include <rmol/bom/SegmentSnapshotTableHelper.hpp>
#include <rmol/bom/HistoricalBookingHolder.hpp>
#include <rmol/bom/HistoricalBooking.hpp>
#include <rmol/command/QForecasting.hpp>
#include <rmol/command/HybridForecasting.hpp>
#include <rmol/command/Detruncator.hpp>
```

Namespaces

- namespace RMOL

9.54 rmol/command/HybridForecasting.hpp File Reference

9.54 rmol/command/HybridForecasting.hpp File Reference

```
#include <stdair/stdair_inventory_types.hpp>
#include <rmol/RMOL_Types.hpp>
```

Namespaces

- namespace stdair
- namespace RMOL

Classes

- class RMOL::HybridForecasting

9.55 rmol/command/InventoryParser.cpp File Reference

9.55 rmol/command/InventoryParser.cpp File Reference

```
#include <sstream>
#include <fstream>
#include <cassert>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/BasConst_DefaultObject.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/bom/BomRetriever.hpp>
#include <stdair/bom/BomManager.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/LegDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/VirtualClassStruct.hpp>
#include <stdair/factory/FacBom.hpp>
#include <stdair/factory/FacBomManager.hpp>
#include <stdair/service/Logger.hpp>
#include <rmol/command/InventoryParser.hpp>
```

Namespaces

- namespace RMOL

9.56 rmol/command/InventoryParser.hpp File Reference

9.56 rmol/command/InventoryParser.hpp File Reference

```
#include <string>
```

```
#include <stdair/command/CmdAbstract.hpp>
```

Namespaces

- namespace stdair
- namespace RMOL

Classes

- class RMOL::InventoryParser

  Class filling the virtual class list (representing a list of classes/buckets) from a given input inventory.

9.57 rmol/command/MarginalRevenueTransformation.cpp File Reference

9.57 rmol/command/MarginalRevenueTransformation.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <cmath>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/SimpleNestingStructure.hpp>
#include <stdair/bom/NestingNode.hpp>
#include <stdair/bom/Policy.hpp>
#include <stdair/factory/FacBomManager.hpp>
#include <stdair/service/Logger.hpp>
#include <rmol/bom/PolicyHelper.hpp>
#include <rmol/bom/Utilities.hpp>
#include <rmol/command/MarginalRevenueTransformation.hpp>
```

Namespaces

- namespace RMOL

9.58 rmol/command/MarginalRevenueTransformation.hpp File Reference

9.58 rmol/command/MarginalRevenueTransformation.hpp File Reference

```
#include <map>
```

```
#include <stdair/stdair_inventory_types.hpp>
#include <rmol/RMOL_Types.hpp>
```

Namespaces

- namespace stdair
- namespace RMOL

Classes

- class RMOL::MarginalRevenueTransformation

9.59 rmol/command/NewQFF.cpp File Reference

9.59 rmol/command/NewQFF.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <cmath>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/SegmentSnapshotTable.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/Policy.hpp>
#include <stdair/service/Logger.hpp>
#include <rmol/bom/Utilities.hpp>
#include <rmol/bom/SegmentSnapshotTableHelper.hpp>
#include <rmol/bom/HistoricalBookingHolder.hpp>
#include <rmol/bom/HistoricalBooking.hpp>
#include <rmol/bom/EMDetruncator.hpp>
#include <rmol/command/NewQFF.hpp>
#include <rmol/command/Detruncator.hpp>
```

Namespaces

- namespace RMOL

9.60 rmol/command/NewQFF.hpp File Reference

9.60 rmol/command/NewQFF.hpp File Reference

```
#include <map>
#include <stdair/stdair_inventory_types.hpp>
#include <rmol/RMOL_Types.hpp>
```

Namespaces

- namespace stdair
- namespace RMOL

Classes

- class RMOL::NewQFF

9.61 rmol/command/OldQFF.cpp File Reference

9.61 rmol/command/OldQFF.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <cmath>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/SegmentSnapshotTable.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/Policy.hpp>
#include <stdair/service/Logger.hpp>
#include <rmol/bom/Utilities.hpp>
#include <rmol/bom/SegmentSnapshotTableHelper.hpp>
#include <rmol/bom/HistoricalBookingHolder.hpp>
#include <rmol/bom/HistoricalBooking.hpp>
#include <rmol/bom/EMDetruncator.hpp>
#include <rmol/command/OldQFF.hpp>
#include <rmol/command/Detruncator.hpp>
```

Namespaces

- namespace RMOL

9.62 rmol/command/OldQFF.hpp File Reference

9.62 rmol/command/OldQFF.hpp File Reference

```
#include <map>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/PolicyTypes.hpp>
#include <rmol/RMOL_Types.hpp>
```

Namespaces

- namespace stdair
- namespace RMOL

Classes

- class RMOL::OldQFF

9.63 rmol/command/Optimiser.cpp File Reference

9.63 rmol/command/Optimiser.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/RandomGeneration.hpp>
#include <stdair/bom/BomManager.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/service/Logger.hpp>
#include <rmol/basic/BasConst_General.hpp>
#include <rmol/bom/MCOptimiser.hpp>
#include <rmol/bom/Emsr.hpp>
#include <rmol/bom/DPOptimiser.hpp>
```

```
#include <rmol/command/Optimiser.hpp>
```

Namespaces

- namespace RMOL

9.64 rmol/command/Optimiser.hpp File Reference

9.64 rmol/command/Optimiser.hpp File Reference

```
#include <stdair/basic/OptimisationMethod.hpp>
#include <rmol/RMOL_Types.hpp>
```

Namespaces

- namespace stdair
- namespace RMOL

Classes

- class RMOL::Optimiser

9.65 rmol/command/PreOptimiser.cpp File Reference

9.65 rmol/command/PreOptimiser.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <cmath>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/RandomGeneration.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/SegmentSnapshotTable.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/service/Logger.hpp>
#include <rmol/bom/Utilities.hpp>
#include <rmol/command/PreOptimiser.hpp>
#include <rmol/command/DemandInputPreparation.hpp>
#include <rmol/command/FareAdjustment.hpp>
```

```
#include <rmol/command/MarginalRevenueTransformation.hpp>
```

Namespaces

- namespace RMOL

9.66 rmol/command/PreOptimiser.hpp File Reference

9.66 rmol/command/PreOptimiser.hpp File Reference

```
#include <map>
#include <stdair/stdair_inventory_types.hpp>
#include <rmol/RMOL_Types.hpp>
```

Namespaces

- namespace stdair
- namespace RMOL

Classes

- class RMOL::PreOptimiser

9.67 rmol/command/QForecasting.cpp File Reference

9.67 rmol/command/QForecasting.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <cmath>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/SegmentSnapshotTable.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/service/Logger.hpp>
#include <rmol/bom/Utilities.hpp>
#include <rmol/bom/SegmentSnapshotTableHelper.hpp>
```

```
#include <rmol/bom/HistoricalBookingHolder.hpp>
#include <rmol/bom/HistoricalBooking.hpp>
#include <rmol/command/QForecasting.hpp>
#include <rmol/command/Detruncator.hpp>
```

Namespaces

- namespace RMOL

9.68 rmol/command/QForecasting.hpp File Reference

9.68 rmol/command/QForecasting.hpp File Reference

```
#include <stdair/stdair_inventory_types.hpp>
#include <rmol/RMOL_Types.hpp>
```

Namespaces

- namespace stdair
- namespace RMOL

Classes

- class RMOL::QForecasting

9.69 rmol/factory/FacRmolServiceContext.cpp File Reference

9.69 rmol/factory/FacRmolServiceContext.cpp File Reference

```
#include <cassert>
#include <stdair/service/FacSupervisor.hpp>
#include <rmol/factory/FacRmolServiceContext.hpp>
#include <rmol/service/RMOL_ServiceContext.hpp>
```

Namespaces

- namespace RMOL

9.70 rmol/factory/FacRmolServiceContext.hpp File Reference

9.70 rmol/factory/FacRmolServiceContext.hpp File Reference

```
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/service/FacServiceAbstract.hpp>
```

Namespaces

- namespace RMOL

Classes

- class RMOL::FacRmolServiceContext

  Factory for the service context.

9.71 rmol/RMOL_Service.hpp File Reference

9.71 rmol/RMOL_Service.hpp File Reference

```
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_service_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/basic/UnconstrainingMethod.hpp>
#include <stdair/basic/ForecastingMethod.hpp>
#include <stdair/basic/PreOptimisationMethod.hpp>
#include <stdair/basic/OptimisationMethod.hpp>
#include <stdair/basic/PartnershipTechnique.hpp>
#include <rmol/RMOL_Types.hpp>
```

Namespaces

- namespace stdair
- namespace RMOL

Classes

- class RMOL::RMOL_Service

  Interface for the RMOL Services.

9.72 rmol/RMOL_Types.hpp File Reference

9.72 rmol/RMOL_Types.hpp File Reference

```
#include <map>
#include <vector>
#include <boost/shared_ptr.hpp>
#include <stdair/stdair_inventory_types.hpp>
```

```
#include <stdair/stdair_rm_types.hpp>
#include <stdair/stdair_exceptions.hpp>
```

Namespaces

- namespace stdair
- namespace RMOL

Classes

- class RMOL::OverbookingException
  Overbooking-related exception.
- class RMOL::UnconstrainingException
  Unconstraining-related exception.
- class RMOL::EmptyNestingStructException
  Empty nesting structure in unconstrainer exception.
- class RMOL::MissingDCPException
  Missing a DCP in unconstrainer exception.
- class RMOL::OptimisationException
  Optimisation-related exception.
- class RMOL::PolicyException
  Policy-related exception.
- class RMOL::ConvexHullException
  Convex Hull-related exception.
- class RMOL::EmptyConvexHullException
  Empty convex hull exception.
- class RMOL::FirstPolicyNotNullException
  Missing policy NULL in convex hull exception.
- class RMOL::YieldConvexHullException
  Yield convex hull exception.
- class RMOL::FareFamilyException
  Fare Family-related exception.
- class RMOL::EmptyBookingClassListException
  Empty Booking Class List of Fare Family exception.
- class RMOL::MissingBookingClassInFareFamilyException
  Missing Booking Class in Fare Family exception.

- class RMOL::FareFamilyDemandVectorSizeException

  Fare Family demand exception.

Typedefs

- typedef boost::shared_ptr< RMOL_Service > RMOL::RMOL_ServicePtr_T
- typedef std::vector< stdair::Flag_T > RMOL::FlagVector_T
- typedef std::map< stdair::BookingClass *, stdair::MeanStdDevPair_T > RMOL::BookingClass-MeanStdDevPairMap_T

9.73 rmol/service/RMOL_Service.cpp File Reference

9.73 rmol/service/RMOL_Service.cpp File Reference

```
#include <cassert>
#include <boost/make_shared.hpp>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/BasChronometer.hpp>
#include <stdair/basic/ContinuousAttributeLite.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomRetriever.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/YieldFeatures.hpp>
#include <stdair/bom/AirportPair.hpp>
#include <stdair/bom/PosChannel.hpp>
#include <stdair/bom/DatePeriod.hpp>
#include <stdair/bom/TimePeriod.hpp>
#include <stdair/bom/AirlineClassList.hpp>
#include <stdair/basic/BasConst_Request.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/OnDDate.hpp>
#include <stdair/bom/OnDDateTypes.hpp>
#include <stdair/command/CmdBomManager.hpp>
```

```
#include <stdair/service/Logger.hpp>
#include <stdair/STDAIR_Service.hpp>
#include <rmol/basic/BasConst_RMOL_Service.hpp>
#include <rmol/factory/FacRmolServiceContext.hpp>
#include <rmol/command/InventoryParser.hpp>
#include <rmol/command/Optimiser.hpp>
#include <rmol/command/PreOptimiser.hpp>
#include <rmol/command/Forecaster.hpp>
#include <rmol/service/RMOL_ServiceContext.hpp>
#include <rmol/RMOL_Service.hpp>
```

Namespaces

- namespace RMOL

9.74 rmol/service/RMOL_ServiceContext.cpp File Reference

9.74 rmol/service/RMOL_ServiceContext.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/STDAIR_Service.hpp>
#include <rmol/basic/BasConst_RMOL_Service.hpp>
#include <rmol/service/RMOL_ServiceContext.hpp>
```

Namespaces

- namespace RMOL

9.75 rmol/service/RMOL_ServiceContext.hpp File Reference

9.75 rmol/service/RMOL_ServiceContext.hpp File Reference

```
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/stdair_service_types.hpp>
#include <stdair/service/ServiceAbstract.hpp>
#include <rmol/RMOL_Types.hpp>
```

Namespaces

- namespace stdair
- namespace RMOL

Classes

- class RMOL::RMOL_ServiceContext

  Inner class holding the context for the RMOL Service object.

9.76 test/rmol/bomsforforecaster.cpp File Reference

9.76 test/rmol/bomsforforecaster.cpp File Reference

9.77 test/rmol/ForecasterTestSuite.cpp File Reference

9.77 test/rmol/ForecasterTestSuite.cpp File Reference

9.78 test/rmol/ForecasterTestSuite.hpp File Reference

9.78 test/rmol/ForecasterTestSuite.hpp File Reference

```
#include <sstream>
#include <cppunit/extensions/HelperMacros.h>
```

Classes

- class ForecasterTestSuite

Functions

- CPPUNIT_TEST_SUITE_REGISTRATION (ForecasterTestSuite)

9.78.1 Function Documentation

9.78.1.1 CPPUNIT_TEST_SUITE_REGISTRATION (ForecasterTestSuite)

9.79 test/rmol/OptimiseTestSuite.cpp File Reference

9.79 test/rmol/OptimiseTestSuite.cpp File Reference

9.80 test/rmol/OptimiseTestSuite.hpp File Reference

9.80 test/rmol/OptimiseTestSuite.hpp File Reference

```
#include <sstream>
#include <cppunit/extensions/HelperMacros.h>
```

Classes

- class OptimiseTestSuite

Functions

- CPPUNIT_TEST_SUITE_REGISTRATION (OptimiseTestSuite)

9.80.1 Function Documentation

9.80.1.1 CPPUNIT_TEST_SUITE_REGISTRATION (OptimiseTestSuite)

9.81 test/rmol/UnconstrainerTestSuite.cpp File Reference

9.81 test/rmol/UnconstrainerTestSuite.cpp File Reference

9.82 test/rmol/UnconstrainerTestSuite.hpp File Reference

9.82 test/rmol/UnconstrainerTestSuite.hpp File Reference

```
#include <sstream>
#include <cppunit/extensions/HelperMacros.h>
```

Classes

- class UnconstrainerTestSuite

Functions

- CPPUNIT_TEST_SUITE_REGISTRATION (UnconstrainerTestSuite)

9.82.1 Function Documentation

9.82.1.1 CPPUNIT_TEST_SUITE_REGISTRATION (UnconstrainerTestSuite)

10 RMOL Page Documentation

10 RMOL Page Documentation

10.1 People

10.1 People

10.1.1 Project Admins

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

10.1.2 Developers

- Anh Quan Nguyen <quannaus@users.sourceforge.net> (N)
- Denis Arnaud <denis_arnaud@users.sourceforge.net> (N)
- Nicolas Bondoux <nbondoux@users.sourceforge.net> (N)

10.1.3 Retired Developers

- Patrick Grandjean <pgrandjean@users.sourceforge.net> (N)
- Benoit Lardeux <benlardeux@users.sourceforge.net> (N)
- Karim Duval <duvalkarim@users.sourceforge.net> (N)
- Ngoc-Thach Hoang <hoangngocthach@users.sourceforge.net> (N)
- Son Nguyen Kim <snguyenkim@users.sourceforge.net> (N)

10.1.4 Contributors

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

10.1.5 Distribution Maintainers

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

Note:

(N) - `Amadeus` employees.

10.2 Coding Rules

10.2 Coding Rules

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

10.2.1 Default Naming Rules for Variables

Variables names follow Java naming conventions. Examples:

- `lNumberOfPassengers`
- `lSeatAvailability`

10.2.2 Default Naming Rules for Functions

Function names follow Java naming conventions. Example:

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

10.2.3 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`

10.2.4 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`

10.2.5 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.

10.3 Copyright and License

10.3 Copyright and License

10.3.1 GNU LESSER GENERAL PUBLIC LICENSE

10.3.1.1 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.]
```

10.3.2 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.

10.3.3 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.

10.3.3.1 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 LIABLE 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.

10.3.3.2 END OF TERMS AND CONDITIONS

10.3.4 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

10.4 Documentation Rules

10.4 Documentation Rules

10.4.1 General Rules

All classes in RMOL 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 RMOL 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
};
```

10.4.2 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.
 *
 * ---------------------------------------------------------------------
 *
 * RMOL - C++ Revenue Management Object Library
 *
 * Copyright (C) 2007-2010  (\see authors file for a list of contributors)
 *
 * \see copyright file for license information
 *
 * ---------------------------------------------------------------------
 */
```

10.4.3 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);
```

10.5 Main features

A short list of the main features of RMOL is given below sorted in different categories. Many more features and functions exist and for these we refer to the reference documentation.

10.5.1 Optimisation features

- `Dynamic Programming (DP)`
- `EMSRa` and `EMSRb`
- Network optimisation with `Linear Programming (LP)`

10.5.2 Unconstraining

- Inventory censorflag and guillotine
- E-M (Expectation Maximisation)

10.5.3 Forecasting features

- `Exponential Smoothing`
- `Moving Average`

10.5.4 Overbooking features

- Cancellations and No-Shows
- Cost-based optimisation
- Service-based optimisation

10.5.5 Other features

- CSV input file parsing

10.6 Make a Difference

Do not ask what RMOL can do for you. Ask what you can do for RMOL.

You can help us to develop the RMOL library. There are always a lot of things you can do:

- Start using RMOL
- Tell your friends about RMOL and help them to get started using it
- If you find a bug, report it to us. 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 RMOL discussion forums on SourceForge. If you know the answer to a question, help others to overcome their RMOL 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 us to port RMOL to new platforms. If you manage to compile RMOL on a new platform, then tell us how you did it.
- Send us your code. If you have a good RMOL compatible code, which you can release under the LGPL, and you think it should be included in RMOL, then send it to us.
- Become an RMOL developer. Send us an e-mail and tell what you can do for RMOL.

10.7 Make a new release

10.7 Make a new release

10.7.1 Introduction

This document describes briefly the recommended procedure of releasing a new version of RMOL using a Linux development machine and the SourceForge project site.

The following steps are required to make a release of the distribution package.

- Initialisation
- Release branch maintenance
- Commit and publish the release branch
- Create source packages (tar-balls)
- Upload the HTML documentation to SourceForge
- Generate the RPM packages
- Update distributed change log
- Create the binary package, including the documentation
- Upload the files to SourceForge
- Make a new post
- Send an email on the announcement mailing-list

10.7.2 Initialisation

Clone locally the full `Git project`:

```
cd ~
mkdir -p dev/sim
cd ~/dev/sim
git clone git://rmol.git.sourceforge.net/gitroot/rmol/rmol rmolgit
cd rmolgit
git checkout trunk
```

10.7.3 Release branch maintenance

Switch to the release branch, on your local clone, and merge the latest updates from the trunk. Decide about the new version to be released.

```
cd ~/dev/sim/rmolgit
git checkout releases
git merge trunk
```

Update the version in the various build system files, replacing the old version numbers by the correct ones:

```
vi CMakeLists.txt
vi autogen.sh
vi README
```

Update the version, add some news in the `NEWS` file, add a change-log in the `ChangeLog` file and in the RPM specification files:

```
vi NEWS
vi ChangeLog
vi rmol.spec
```

10.7.4 Commit and publish the release branch

Commit the new release:

```
cd ~/dev/sim/rmolgit
git add -A
git commit -m "[Release 0.5.0] Release of the 0.5.0 version of RMOL."
git push
```

10.7.5 Create source packages (tar-balls)

Create the distribution packages using the following command:

```
cd ~/dev/sim/rmolgit
git checkout releases
rm -rf build && mkdir -p build
cd build
export INSTALL_BASEDIR=/home/user/dev/deliveries
export LIBSUFFIX_4_CMAKE="-DLIB_SUFFIX=64"
cmake -DCMAKE_INSTALL_PREFIX=${INSTALL_BASEDIR}/rmol-0.5.0 \
 -DWITH_STDAIR_PREFIX=${INSTALL_BASEDIR}/stdair-stable \
 -DWITH_AIRRAC_PREFIX=${INSTALL_BASEDIR}/airsched-stable \
 -DWITH_AIRRAC_PREFIX=${INSTALL_BASEDIR}/airrac-stable \
 -DWITH_RMOL_PREFIX=${INSTALL_BASEDIR}/rmol-stable \
 -DWITH_RMOL_PREFIX=${INSTALL_BASEDIR}/airinv-stable \
 -DWITH_RMOL_PREFIX=${INSTALL_BASEDIR}/simfqt-stable \
 -DCMAKE_BUILD_TYPE:STRING=Debug -DINSTALL_DOC:BOOL=ON \
 ${LIBSUFFIX_4_CMAKE} ..
make check && make dist
make install
```

This will configure, compile and check the package. The output packages will be named, for instance, `rmol-0.5.0.tar.gz` and `rmol-0.5.0.tar.bz2`.

10.7.6 Upload the HTML documentation to SourceForge

In order to update the Web site files, either:

- synchronise them with rsync and SSH: Upload the just generated HTML (and PDF) documentation onto the SourceForge Web site.

  ```
  cd ~/dev/sim/rmolgit/build
  git checkout releases
  rsync -aiv ${INSTALL_BASEDIR}/rmol-0.5.0/share/doc/rmol-0.5.0/html/ \
   your_sf_user,rmol@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.

10.7.7 Generate the RPM packages

Optionally, generate the RPM package (for instance, for Fedora/RedHat):

```
cd ~/dev/sim/rmolgit/build
git checkout releases
make dist
```

To perform this step, rpm-build, rpmlint and rpmdevtools have to be available on the system.

```
cp ../rmol.spec ~/dev/packages/SPECS \
  && cp rmol-0.5.0.tar.bz2 ~/dev/packages/SOURCES
cd ~/dev/packages/SPECS
rpmbuild -ba rmol.spec
cd ~/dev/packages
rpmlint -i SPECS/rmol.spec SRPMS/rmol-0.5.0-1.fc16.src.rpm \
  RPMS/noarch/rmol-* RPMS/i686/rmol-*
```

10.7.8 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 RMOL's Git repository.

10.7.9 Create the binary package, including the documentation

Create the binary package, which includes HTML and PDF documentation, using the following command:

```
cd ~/dev/sim/rmolgit/build
git checkout releases
make package
```

The output binary package will be named, for instance, rmol-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.

10.7.10 Upload the files to SourceForge

Upload the distribution and documentation packages to the SourceForge server. Check SourceForge help page on uploading software.

10.7.11 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.

10.7.12 Send an email on the announcement mailing-list

Finally, you should send an announcement to rmol-announce@lists.sourceforge.net (see https://lists.sourceforge.net/lists/listinfo/rmol-announce for the archives)

10.8 Installation

10.8 Installation

10.8.1 Table of Contents

10.8.2 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 rmol-devel rmol-doc
```

RPM packages can also be available on the SourceForge download site.

10.8.3 RMOL Requirements

RMOL 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'`)
- `SOCI` - 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 RMOL.

10.8.4 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.

10.8.5 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.

10.8.6 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.

10.8.7 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.

10.8.8 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'.

10.8.9 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
```

10.8.10 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'.

10.8.11 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.

10.8.12 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
```

10.8.13 '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 ouput like this:

```
cmake -DCMAKE_INSTALL_PREFIX=/home/user/dev/deliveries/rmol-99.99.99 -DLIB_SUFFIX=64 -DCMAKE_BUILD_TYPE:ST
-- 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: 56c6c98cf2cfb4008a0acd35d08075cf5f79e693 trunk
-- 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 MySQL without specifying any version
-- Using mysql-config: /usr/bin/mysql_config
-- Found MySQL: /usr/lib64/mysql/libmysqlclient.so
-- 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 StdAir-0.35
-- Found StdAir version: 0.37.1
-- Requires Doxygen without specifying any version
-- Found Doxygen: /usr/bin/doxygen
-- Found DoxygenWrapper: /usr/bin/doxygen
-- Found Doxygen version: 1.7.4
-- Had to set the linker language for 'airraclib' to CXX
-- Had to set the linker language for 'rmollib' to CXX
-- Test 'UnconstrainerTest' to be built with 'UnconstrainerTestSuite.cpp'
-- Test 'ForecasterTest' to be built with 'ForecasterTestSuite.cpp'
-- Test 'OptimiseTest' to be built with 'OptimiseTestSuite.cpp'
-- Test 'BOMsForForecasterTest' to be built with 'bomsforforecaster.cpp'
--
-- ============================================================
-- ---------------------------------
-- ---     Project Information     ---
-- ---------------------------------
-- PROJECT_NAME ................... : rmol
```

```
-- PACKAGE_PRETTY_NAME ............ : RMOL
-- PACKAGE ........................ : rmol
-- PACKAGE_NAME ................... : RMOL
-- PACKAGE_BRIEF .................. : C++ library of Revenue Management and Optimisation classes and funct
-- PACKAGE_VERSION ................ : 99.99.99
-- GENERIC_LIB_VERSION ............ : 99.99.99
-- GENERIC_LIB_SOVERSION .......... : 99.99
--
-- ----------------------------------
-- ---     Build Configuration    ---
-- ----------------------------------
-- Modules to build ............... : airrac;rmol
-- Libraries to build/install ..... : airraclib;rmollib
-- Binaries to build/install ...... : airrac;rmol
-- Modules to test ................ : rmol
-- Binaries to test ............... : UnconstrainerTesttst;UnconstrainerTesttst;ForecasterTesttst;Unconstr
--
-- * Module ....................... : airrac
--   + Layers to build ............ : .;basic;bom;factory;command;service
--   + Dependencies on other layers :
--   + Libraries to build/install . : airraclib
--   + Executables to build/install : airrac
--   + Tests to perform ........... :
-- * Module ....................... : rmol
--   + Layers to build ............ : .;basic;bom;factory;command;service
--   + Dependencies on other layers : airraclib
--   + Libraries to build/install . : rmollib
--   + Executables to build/install : rmol
--   + Tests to perform ........... : UnconstrainerTesttst;UnconstrainerTesttst;ForecasterTesttst;Unconstr
--
-- BUILD_SHARED_LIBS .............. : ON
-- CMAKE_BUILD_TYPE ............... : Debug
--  * CMAKE_C_FLAGS ............... :
--  * CMAKE_CXX_FLAGS ............. : -Wall -Werror
--  * BUILD_FLAGS ................. :
--  * COMPILE_FLAGS ............... :
-- CMAKE_MODULE_PATH .............. : /home/user/dev/sim/rmol/rmolgithub/config/
-- CMAKE_INSTALL_PREFIX ........... : /home/user/dev/deliveries/rmol-99.99.99
--
-- * 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/rmol-99.99.99/lib64
-- INSTALL_BIN_DIR ................ : /home/user/dev/deliveries/rmol-99.99.99/bin
-- INSTALL_INCLUDE_DIR ............ : /home/user/dev/deliveries/rmol-99.99.99/include
-- INSTALL_DATA_DIR ............... : /home/user/dev/deliveries/rmol-99.99.99/share
-- INSTALL_SAMPLE_DIR ............. : /home/user/dev/deliveries/rmol-99.99.99/share/rmol/samples
-- INSTALL_DOC .................... : ON
--
-- ----------------------------------
-- ---   Packaging Configuration  ---
-- ----------------------------------
-- CPACK_PACKAGE_CONTACT .......... : Denis Arnaud <denis_arnaud - at - users dot sourceforge dot net>
-- CPACK_PACKAGE_VENDOR ........... : Denis Arnaud
-- CPACK_PACKAGE_VERSION .......... : 99.99.99
-- CPACK_PACKAGE_DESCRIPTION_FILE . : /home/user/dev/sim/rmol/rmolgithub/README
-- CPACK_RESOURCE_FILE_LICENSE .... : /home/user/dev/sim/rmol/rmolgithub/COPYING
-- CPACK_GENERATOR ................ : TBZ2
-- CPACK_DEBIAN_PACKAGE_DEPENDS ... :
-- CPACK_SOURCE_GENERATOR ......... : TBZ2;TGZ
-- CPACK_SOURCE_PACKAGE_FILE_NAME . : rmol-99.99.99
```

```
--
-- --------------------------------
-- ---     External libraries    ---
-- --------------------------------
--
-- * 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;unit_te
--   - Boost required libraries ... : optimized;/usr/lib64/libboost_iostreams-mt.so;debug;/usr/lib64/libbo
--
-- * 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
--   - SOCIMYSQL_INCLUDE_DIR ...... : /usr/include/soci
--   - SOCI_LIBRARIES ............. : /usr/lib64/libsoci_core.so
--   - SOCIMYSQL_LIBRARIES ........ : /usr/lib64/libsoci_mysql.so
--
-- * StdAir:
--   - STDAIR_VERSION ............. : 0.37.1
--   - STDAIR_BINARY_DIRS ......... : /home/user/dev/deliveries/stdair-0.37.1/bin
--   - STDAIR_EXECUTABLES ......... : stdair
--   - STDAIR_LIBRARY_DIRS ........ : /home/user/dev/deliveries/stdair-0.37.1/lib64
--   - STDAIR_LIBRARIES ........... : stdairlib;stdairuicllib
--   - STDAIR_INCLUDE_DIRS ........ : /home/user/dev/deliveries/stdair-0.37.1/include
--   - STDAIR_SAMPLE_DIR .......... : /home/user/dev/deliveries/stdair-0.37.1/share/stdair/samples
--
-- Change a value with: cmake -D<Variable>=<Value>
-- ================================================================
--
-- Configuring done
-- Generating done
-- Build files have been written to: /home/user/dev/sim/rmol/rmolgithub/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_rmol
[  0%] Built target hdr_cfg_airrac
[ 30%] Built target airraclib
[ 86%] Built target rmollib
[ 90%] Built target BOMsForForecasterTesttst
[ 93%] Built target UnconstrainerTesttst
[ 96%] Built target ForecasterTesttst
[100%] Built target OptimiseTesttst
Scanning dependencies of target check_rmoltst
Test project /home/user/dev/sim/rmol/rmolgithub/build/test/rmol
    Start 1: UnconstrainerTesttst
1/4 Test #1: UnconstrainerTesttst .............   Passed    0.04 sec
    Start 2: ForecasterTesttst
2/4 Test #2: ForecasterTesttst ................   Passed    0.04 sec
    Start 3: OptimiseTesttst
3/4 Test #3: OptimiseTesttst ..................   Passed    0.44 sec
    Start 4: BOMsForForecasterTesttst
4/4 Test #4: BOMsForForecasterTesttst .........   Passed    0.02 sec
```

```
100% tests passed, 0 tests failed out of 4

Total Test time (real) =   0.78 sec
[100%] Built target check_rmoltst
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/rmolgit
rm -rf build && mkdir build
cd build
```

to remove everything.

10.9 Linking with RMOL

10.9 Linking with RMOL

10.9.1 Table of Contents

10.9.2 Introduction

There are two convenient methods of linking your programs with the RMOL library. The first one employs the 'pkg-config' command (see http://pkgconfig.freedesktop.org/), whereas the second one uses 'rmol-config' script. These methods are shortly described below.

10.9.3 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 RMOL based program 'my_prog.cpp', you should use the following command:

```
g++ `pkg-config --cflags rmol` -o my_prog my_prog.cpp `pkg-config --libs rmol`
```

For more information see the 'pkg-config' man pages.

10.9.4 Using the rmol-config script

RMOL provides a shell script called 'rmol-config', which is installed by default in '$prefix/bin' ('/usr/local/bin') directory. It can be used to simplify compilation and linking of RMOL 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++ `rmol-config --cflags` -o my_prog_opt my_prog.cpp `rmol-config --libs`
```

A list of 'rmol-config' options can be obtained by typing:

```
rmol-config --help
```

If the 'rmol-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
```

10.9.5 M4 macro for the GNU Autotools

A M4 macro file is delivered with RMOL, namely 'rmol.m4', which can be found in, e.g., '/usr/share/aclocal'. When used by a 'configure' script, thanks to he 'AM_PATH_RMOL' macro (specified in the M4 macro file), the following Makefile variables are then defined:

- 'RMOL_VERSION' (e.g., defined to 0.23.0)
- 'RMOL_CFLAGS' (e.g., defined to '-I${prefix}/include')
- 'RMOL_LIBS' (e.g., defined to '-L${prefix}/lib -lrmol')

10.9.6 Using RMOL with dynamic linking

When using static linking some of the library routines in RMOL 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 RMOL library file during your program execution. If you install the RMOL 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=<RMOL installation prefix>/lib:$LD_LIBRARY_PATH
```

10.10 Test Rules

10.10 Test Rules

This section describes rules how the functionality of the IT++ library should be verified. In the 'tests' subdirectory test files are provided. All functionality should be tested using these test files.

10.10.1 The Test File

Each new IT++ module/class should be accompanied with a test file. The test file is an implementation in C++ that tests the functionality of a function/class or a group of functions/classes called modules. The test file should test relevant parameter settings and input/output relations to guarantee correct functionality of the corresponding classes/functions. The test files should be maintained using version control and updated whenever new functionality is added to the IT++ library.

The test 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 file should be placed in the 'tests' subdirectory and should have a name ending with '_test.cpp'.

10.10.2 The Reference File

Consider a test file named 'module_test.cpp'. A reference file named 'module_test.ref' should accompany the test 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 file.

10.10.3 Testing IT++ Library

One can compile and execute all test programs from 'tests' subdirectory by typing

```
% make check
```

after successful compilation of the IT++ library.

10.11 Users Guide

10.11 Users Guide

10.11.1 Table of Contents

10.11.2 Introduction

The RMOL library contains classes for revenue management. This document does not cover all the aspects of the RMOL library. It does however explain the most important things you need to know in order to start using RMOL.

10.11.3 Get Started

10.11.3.1 Get the RMOL library

10.11.3.2 Build the RMOL project To run the configuration script the first time, go to the top directory (where the RMOL package has been un-packed), and issue the following command:

- `mkdir -p build && cd build && cmake ..`
- `make`

Note:

The RMOL project can either be cloned from the Git Repository or downloaded as a tar-ball package from the Sourceforge Web site.

10.11.3.3 Build and Run the Tests

10.11.3.4 Install the RMOL Project (Binaries, Documentation)

10.11.4 Exploring the Predefined BOM Tree

RMOL predefines a BOM (Business Object Model) tree specific to the airline IT arena.

10.11.4.1 Forecaster BOM Tree

- RMOL::EMDetruncator
- RMOL::Detruncator
- RMOL::Forecaster

10.11.4.2 Optimiser BOM Tree

- RMOL::DPOptimiser
- RMOL::MCOptimiser
- RMOL::Optimiser

10.11.5 Extending the BOM Tree

10.12 Supported Systems

10.12 Supported Systems

10.12.1 Table of Contents

10.12.2 Introduction

This page is intended to provide a list of RMOL supported systems, i.e. the systems on which configuration, installation and testing process of the RMOL library has been sucessful. 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 RMOL library on a system not mentioned below, please let us know, so we could update this database.

10.12.3 RMOL 0.23.x

10.12.3.1 Linux Systems

Fedora Core 4 with ATLAS

- **Platform**: Intel Pentium 4
- **Operating System**: Fedora Core 4 (x86)
- **Compiler**: g++ (GCC) 4.0.2 20051125
- **RMOL release**: 0.23.0
- **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**: RMOL configured with:

```
% CXXFLAGS="-O3 -pipe -march=pentium4" ./configure
```

- **Date**: March 7, 2006
- **Tester**: Tony Ottosson

Gentoo Linux with ACML

- **Platform**: AMD Sempron 3000+
- **Operating System**: Gentoo Linux 2006.0 (x86 arch)
- **Compiler(s)**: g++ (GCC) 3.4.5
- **RMOL release**: 0.23.1

- **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
```

RMOL configured with:

```
% export CPPFLAGS="-I/usr/include/acml"
% ./configure --with-blas="-lblas"
```

- **Date**: March 31, 2006
- **Tester**: Adam Piatyszek (ediap)

Gentoo Linux with ATLAS

- **Platform**: Intel Pentium M Centrino
- **Operating System**: Gentoo Linux 2006.0 (x86)
- **Compiler**: g++ (GCC) 3.4.5
- **RMOL release**: 0.23.1
- **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
```

RMOL configured with:

```
% ./configure --with-blas="-lblas"
```

- **Date**: March 31, 2006
- **Tester**: Adam Piatyszek (ediap)

Gentoo Linux with MKL

- **Platform**: Intel Pentium M Centrino
- **Operating System**: Gentoo Linux 2006.0 (x86 arch)
- **Compiler**: g++ (GCC) 3.4.5
- **RMOL release**: 0.23.0
- **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**: RMOL 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)

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
- **RMOL release**: 0.23.1
- **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
```

RMOL configured with:

```
% ./configure --with-blas="-lblas"
```

- **Date**: March 31, 2006
- **Tester**: Adam Piatyszek (ediap)

Red Hat Enterprise Linux with RMOL 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)
- **RMOL release**: 0.23.0
- **External Libraries**: BLAS, CBLAS, LAPACK and FFTW libraries from RMOL External 2.1.1 package
- **Tests Status**: All tests PASSED
- **Date**: March 7, 2006
- **Tester**: Erik G. Larsson

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
- **RMOL release**: 0.23.0
- **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**: RMOL 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)

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
- **RMOL release**: 0.23.0
- **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**: RMOL 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)

10.12.3.2 Windows Systems

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)
- **RMOL release**: 0.23.1
- **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. RMOL configured with:

```
% ./configure
```

- **Date**: March 31, 2006
- **Tester**: Adam Piatyszek (ediap)

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)
- **RMOL release**: 0.23.1
- **External Libraries**: Installed from Cygwin's repository:
  - `fftw-3.0.1-2`
  - `fftw-dev-3.0.1-1`

  ATLAS BLAS and LAPACK libraries from RMOL External 2.1.1 package configured using:

```
% ./configure --enable-atlas --disable-fftw
```

- **Tests Status**: All tests PASSED
- **Comments**: Only static library can be built. RMOL configured with:

```
% export LDFLAGS="-L/usr/local/lib"
% ./configure
```

- **Date**: March 31, 2006
- **Tester**: Adam Piatyszek (ediap)

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)
- **RMOL release**: 0.23.2
- **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. RMOL 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)

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)
- **RMOL release**: 0.23.2
- **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. RMOL 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)

Microsoft Windows XP with MinGW, MSYS and RMOL 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)
- **RMOL release**: 0.23.5
- **External Libraries**: BLAS, CBLAS, LAPACK and FFTW libraries from RMOL External 2.2.0 package
- **Tests Status**: All tests PASSED
- **Comments**: Only static library can be built. RMOL 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)

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
- **RMOL release**: 0.23.5
- **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 RMOL based programs compiled and run with success.
- **Comments**: Only static library can be built. RMOL built by opening the `"win32\rmol.vcproj"` project file in MSVC++ and executing `"Build → Build Solution"` command from menu.
- **Date**: August 11, 2006
- **Tester**: Adam Piatyszek (ediap)

10.12.3.3 Unix Systems

SunOS 5.9 with RMOL External

- **Platform**: SUNW, Sun-Blade-100 (SPARC)
- **Operating System**: SunOS 5.9 Generic_112233-10
- **Compiler(s)**: g++ (GCC) 3.4.5
- **RMOL release**: 0.23.2
- **External Libraries**: BLAS, CBLAS, LAPACK and FFTW libraries from RMOL 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**: RMOL 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)

10.13 RMOL Supported Systems (Previous Releases)

10.13 RMOL Supported Systems (Previous Releases)

10.13.1 RMOL 3.9.1

10.13.2 RMOL 3.9.0

10.13.3 RMOL 3.8.1

10.14 Tutorials

10.14 Tutorials

10.14.1 Table of Contents

10.14.2 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).

10.14.2.1 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.

10.14.3 Build a Predefined BOM Tree

A few steps:

- Instanciate the BOM Root Object
- Instanciate the (Airline) Inventory Object
- Link the Inventory Object with the BOM Root

10.14.3.1 Instanciate 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:

10.14.3.2 Instanciate 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.

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`.

10.14.3.3 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:

10.14.3.4 Build Another Airline Inventory

Another airline inventory object, corresponding to the Air France (Air France) company, is instantiated the same way:

See the corresponding full program (cmd_bom_manager_cpp) for more details.

10.14.3.5 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...

... and browsed:

See the corresponding full program (bom_display_cpp) for more details.

10.14.3.6 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]../../../stdair/command/CmdBomManager.cpp:41: StdAir will build the BOM tree from built-in specificatio
[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, Elapsed, 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, CommSpace, AvPool, Avl, NAV,
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,
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, Elapsed, 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, CommSpace, AvPool, Avl, NAV,
AF84 2011-Mar-20, CDG-SFO 2011-Mar-20, 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,
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,
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 (batch_stdair_cpp) for more details.

10.14.4 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.

10.14.4.1 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:

The STL containers have to be defined accordingly too:

See the full class definition (test_archi_inv_hpp) and implementation (test_archi_inv_cpp) for more details.

10.14.4.2 Build the Specific BOM Objects The BOM root object (`stdair::BomRoot`) is instantiated the classical way:

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:

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:

Another specific airline inventory object is instantiated the same way:

From the BomRoot (of type stdair::BomRoot) object instance, the list of specific airline inventories (of type stdair::Inventory) can then be retrieved...

... and browsed:

10.14.4.3 Result of the Tutorial Program

When this program is run, the output should look like:

```
Inventory: BA
Inventory: AF
```

See the corresponding full program (StandardAirlineITTestSuite_cpp) for more details.

10.15 Command-Line Test to Demonstrate How To Test the RMOL Project

10.15 Command-Line Test to Demonstrate How To Test the RMOL Project

```
 */
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <limits>
#include <sstream>
#include <fstream>
#include <string>
// Boost Unit Test Framework (UTF)
#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MAIN
#define BOOST_TEST_MODULE OptimiseTestSuite
#include <boost/test/unit_test.hpp>
// StdAir
#include <stdair/basic/BasLogParams.hpp>
#include <stdair/basic/BasDBParams.hpp>
#include <stdair/service/Logger.hpp>
// RMOL
#include <rmol/RMOL_Service.hpp>
#include <rmol/config/rmol-paths.hpp>

namespace boost_utf = boost::unit_test;

// (Boost) Unit Test XML Report
std::ofstream utfReportStream ("bomsforforecaster_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() {
  }
};

namespace RMOL {
```

```
struct BookingClassData {

  // Attributes
  double _bookingCount;
  double _fare;
  double _sellupFactor;
  bool _censorshipFlag;

  // Constructer
  BookingClassData (const double iBookingCount, const double iFare,
                    const double iSellupFactor, const bool iCensorshipFlag)
    : _bookingCount(iBookingCount), _fare(iFare),
      _sellupFactor(iSellupFactor), _censorshipFlag(iCensorshipFlag) {
  }

  // Getters
  double getFare () const {
    return _fare;
  }

  bool getCensorshipFlag () const {
    return _censorshipFlag;
  }

  // Display
  std::string toString() const {
    std::ostringstream oStr;
    oStr <<  std::endl
         << "[Booking class data information]" << std::endl
         << "Booking counter: " << _bookingCount << std::endl
         << "Fare: " << _fare << std::endl
         << "Sell-up Factor: " << _sellupFactor << std::endl
         << "censorshipFlag: " << _censorshipFlag << std::endl;
    return oStr.str();
  }

};

struct BookingClassDataSet {

  typedef std::vector<BookingClassData*> BookingClassDataList_T;

  // Attributes
  int _numberOfClass;
  double _minimumFare;
  bool _censorshipFlag; // true if any of the classes is censored
  BookingClassDataList_T _bookingClassDataList;

  // Constructor
  BookingClassDataSet ()
    : _numberOfClass(0), _minimumFare(0),
      _censorshipFlag(false) {
  }

  // Add BookingClassData
  void addBookingClassData (BookingClassData& ioBookingClassData) {
    _bookingClassDataList.push_back (&ioBookingClassData);
  }

  // Getters
  stdair::NbOfClasses_T getNumberOfClass () const {
    return _bookingClassDataList.size();
  }

  double getMinimumFare () const {
    return _minimumFare;
```

```
}

bool getCensorshipFlag () const {
  return _censorshipFlag;
}

// Setters
void setMinimumFare (const double iMinFare) {
  _minimumFare = iMinFare;
}

void setCensorshipFlag (const bool iCensorshipFlag) {
  _censorshipFlag = iCensorshipFlag;
}

// compute minimum fare
void updateMinimumFare() {
  double minFare = std::numeric_limits<double>::max();
  BookingClassDataList_T::iterator itBookingClassDataList;
  for (itBookingClassDataList = _bookingClassDataList.begin();
       itBookingClassDataList != _bookingClassDataList.end();
       ++itBookingClassDataList) {
    BookingClassData* lBookingClassData = *itBookingClassDataList;
    assert (lBookingClassData != NULL);

    const double lFare = lBookingClassData->getFare();
    if (lFare < minFare) {
      minFare = lFare;
    }
  }
  //
  setMinimumFare(minFare);
}

// compute censorship flag for the data set
void updateCensorshipFlag () {
  bool censorshipFlag = false;
  BookingClassDataList_T::iterator itBookingClassDataList;
  for (itBookingClassDataList = _bookingClassDataList.begin();
       itBookingClassDataList != _bookingClassDataList.end();
       ++itBookingClassDataList) {
    BookingClassData* lBookingClassData = *itBookingClassDataList;
    assert (lBookingClassData != NULL);

    const bool lCensorshipFlagOfAClass =
      lBookingClassData->getCensorshipFlag();
    if (lCensorshipFlagOfAClass) {
      censorshipFlag = true;
      break;
    }
  }
  //
  setCensorshipFlag(censorshipFlag);
}

// Display
std::string toString() const {
  std::ostringstream oStr;
  oStr << std::endl
       << "[Booking class data set information]" << std::endl
       << "Number of classes: " << _numberOfClass << std::endl
       << "Minimum fare: " << _minimumFare << std::endl
       << "The data of the class set are sensored: " << _censorshipFlag
       << std::endl;
  return oStr.str();
}
```

```
  };

  // /**-------------- BOM : Q-Forecaster ----------------------- */
  // struct QForecaster {

  //   // Function focused BOM

  //   // 1. calculate sell up probability for Q-eq

  //   // 2. calculate Q-Equivalent Booking
  //   double calculateQEqBooking (BookingClassDataSet& iBookingClassDataSet) {
  //     double lQEqBooking = 0.0;
  //     double lMinFare = iBookingClassDataSet.getMinimumFare();


  //     return lQEqBooking;
  //   }

  //   /* Calculate Q-equivalent demand
  //      [<- performed by unconstrainer if necessary (Using ExpMax BOM)]
  //   */


  //   // 3. Partition to each class

  //   //

  // };

}

// ///////////////// Main: Unit Test Suite //////////////

// Set the UTF configuration (re-direct the output to a specific file)
BOOST_GLOBAL_FIXTURE (UnitTestConfig);

BOOST_AUTO_TEST_SUITE (master_test_suite)


BOOST_AUTO_TEST_CASE (rmol_forecaster) {

  // Output log File
  std::string lLogFilename ("bomsforforecaster.log");
  std::ofstream logOutputFile;

  // Open and clean the log outputfile
  logOutputFile.open (lLogFilename.c_str());
  logOutputFile.clear();

  // Initialise the RMOL service
  const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);

  // Initialise the RMOL service
  RMOL::RMOL_Service rmolService (lLogParams);

  // Build a sample BOM tree
  rmolService.buildSampleBom();

  // Register BCDataSet
  RMOL::BookingClassDataSet lBookingClassDataSet;

  // Register BookingClassData
  RMOL::BookingClassData QClassData (10, 100, 1, false);
  RMOL::BookingClassData MClassData (5, 150, 0.8, true);
  RMOL::BookingClassData BClassData (0, 200, 0.6, false);
  RMOL::BookingClassData YClassData (0, 300, 0.3, false);
```

```
  // Display
  STDAIR_LOG_DEBUG (QClassData.toString());
  STDAIR_LOG_DEBUG (MClassData.toString());
  STDAIR_LOG_DEBUG (BClassData.toString());
  STDAIR_LOG_DEBUG (YClassData.toString());

  // Add BookingClassData into the BCDataSet
  lBookingClassDataSet.addBookingClassData (QClassData);
  lBookingClassDataSet.addBookingClassData (MClassData);
  lBookingClassDataSet.addBookingClassData (BClassData);
  lBookingClassDataSet.addBookingClassData (YClassData);

  // DEBUG
  STDAIR_LOG_DEBUG (lBookingClassDataSet.toString());

  // Number of classes
  const stdair::NbOfClasses_T lNbOfClass = lBookingClassDataSet.getNumberOfClass();

  // DEBUG
  STDAIR_LOG_DEBUG ("Number of Classes: " << lNbOfClass);

  // Minimum fare
  BOOST_CHECK_NO_THROW (lBookingClassDataSet.updateMinimumFare());
  const double lMinFare = lBookingClassDataSet.getMinimumFare();

  // DEBUG
  STDAIR_LOG_DEBUG ("Minimum fare: " << lMinFare);

  // Censorship flag
  BOOST_CHECK_NO_THROW (lBookingClassDataSet.updateCensorshipFlag());
  const bool lCensorshipFlag = lBookingClassDataSet.getCensorshipFlag();

  // DEBUG
  STDAIR_LOG_DEBUG ("Censorship Flag: " << lCensorshipFlag);

  // Close the log output file
  logOutputFile.close();
}

// End the test suite
BOOST_AUTO_TEST_SUITE_END()

/*!
```

10.16 Command-Line Test to Demonstrate How To Test the RMOL Project

10.16 Command-Line Test to Demonstrate How To Test the RMOL Project

```
 */
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <sstream>
#include <fstream>
#include <string>
#include <vector>
#include <cmath>
// Boost Unit Test Framework (UTF)
#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MAIN
#define BOOST_TEST_MODULE ForecasterTestSuite
#include <boost/test/unit_test.hpp>
// StdAir
#include <stdair/basic/BasLogParams.hpp>
#include <stdair/basic/BasDBParams.hpp>
```

```
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/service/Logger.hpp>
// RMOL
#include <rmol/RMOL_Service.hpp>

namespace boost_utf = boost::unit_test;

// (Boost) Unit Test XML Report
std::ofstream utfReportStream ("ForecasterTestSuite_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);

BOOST_AUTO_TEST_SUITE (master_test_suite)


BOOST_AUTO_TEST_CASE (rmol_forecaster_q_forecasting) {
  const bool lTestFlag = true; //testForecasterHelper(0);
  BOOST_CHECK_EQUAL (lTestFlag, true);
  BOOST_CHECK_MESSAGE (lTestFlag == true,
                       "The test has failed. Please see the log file for "
                       << "more details");
}

// End the test suite
BOOST_AUTO_TEST_SUITE_END()

/*!
```

10.17 Command-Line Test to Demonstrate How To Test the RMOL Project

10.17 Command-Line Test to Demonstrate How To Test the RMOL Project

```
 */
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <sstream>
#include <fstream>
#include <string>
// Boost Unit Test Framework (UTF)
#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MAIN
#define BOOST_TEST_MODULE OptimiseTestSuite
#include <boost/test/unit_test.hpp>
// StdAir
#include <stdair/basic/BasLogParams.hpp>
#include <stdair/basic/BasDBParams.hpp>
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/service/Logger.hpp>
```

```
// RMOL
#include <rmol/basic/BasConst_General.hpp>
#include <rmol/RMOL_Service.hpp>
#include <rmol/config/rmol-paths.hpp>

namespace boost_utf = boost::unit_test;

// (Boost) Unit Test XML Report
std::ofstream utfReportStream ("OptimiseTestSuite_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() {
  }
};


// ////////////////////////////////////////////////////////////////////
int testOptimiseHelper (const unsigned short optimisationMethodFlag,
                        const bool isBuiltin) {

  // Return value
  int oExpectedBookingLimit = 0;

  // Output log File
  std::ostringstream oStr;
  oStr << "OptimiseTestSuite_" << optimisationMethodFlag << "_" << isBuiltin << ".log";
  const stdair::Filename_T lLogFilename (oStr.str());

  // Number of random draws to be generated (best if greater than 100)
  const int K = RMOL::DEFAULT_NUMBER_OF_DRAWS_FOR_MC_SIMULATION;

  // Methods of optimisation (0 = Monte-Carlo, 1 = Dynamic Programming,
  // 2 = EMSR, 3 = EMSR-a, 4 = EMSR-b, 5 = EMSR-a with sellup prob.)
  const unsigned short METHOD_FLAG = optimisationMethodFlag;

  // Cabin Capacity (it must be greater then 100 here)
  const double cabinCapacity = 100.0;

  // Set the log parameters
  std::ofstream logOutputFile;
  // Open and clean the log outputfile
  logOutputFile.open (lLogFilename.c_str());
  logOutputFile.clear();

  // Initialise the RMOL service
  const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
  RMOL::RMOL_Service rmolService (lLogParams);

  // Check wether or not a (CSV) input file should be read
  if (isBuiltin == true) {

    // Build the default sample BOM tree and build a dummy BOM tree.
    rmolService.buildSampleBom();

  } else {

    // Parse the optimisation data and build a dummy BOM tree
    const stdair::Filename_T lRMInputFileName (STDAIR_SAMPLE_DIR "/rm02.csv");
    rmolService.parseAndLoad (cabinCapacity, lRMInputFileName);
  }
```

```
  switch (METHOD_FLAG) {
  case 0: {
    // DEBUG
    STDAIR_LOG_DEBUG ("Optimisation by Monte-Carlo (MC)");

    // Calculate the optimal protections by the Monte Carlo
    // Integration approach
    rmolService.optimalOptimisationByMCIntegration (K);
    break;
  }

  case 1: {
    // DEBUG
    STDAIR_LOG_DEBUG ("Optimisation by Dynamic Programming (DP)");

    // Calculate the optimal protections by DP.
    rmolService.optimalOptimisationByDP ();
    break;
  }

  case 2: {
    // DEBUG
    STDAIR_LOG_DEBUG ("Calculate the Bid-Price Vectors (BPV) by EMSR");

    // Calculate the Bid-Price Vector by EMSR
    rmolService.heuristicOptimisationByEmsr ();
    break;
  }

  case 3: {
    // DEBUG
    STDAIR_LOG_DEBUG ("Calculate the Authorisation Levels (AUs) by EMSRa");

    // Calculate the protections by EMSR-a
    // Test the EMSR-a algorithm implementation
    rmolService.heuristicOptimisationByEmsrA ();

    // Return a cumulated booking limit value to test
    // oExpectedBookingLimit = static_cast<int> (lBookingLimitVector.at(2));
    break;
  }

  case 4: {
    // DEBUG
    STDAIR_LOG_DEBUG ("Calculate the Authorisation Levels (AUs) by EMSRb");

    // Calculate the protections by EMSR-b
    rmolService.heuristicOptimisationByEmsrB ();
    break;
  }

  default: rmolService.optimalOptimisationByMCIntegration (K);
  }

  // Close the log file
  logOutputFile.close();

  return oExpectedBookingLimit;
}


// /////////////// Main: Unit Test Suite //////////////

// Set the UTF configuration (re-direct the output to a specific file)
BOOST_GLOBAL_FIXTURE (UnitTestConfig);
```

```
// //////////////////////////////////////////////////////////////////
// Tests are based on the following input values
// price; mean; standard deviation;
// 1050; 17.3; 5.8;
// 567; 45.1; 15.0;
// 534; 39.6; 13.2;
// 520; 34.0; 11.3;
// //////////////////////////////////////////////////////////////////

BOOST_AUTO_TEST_SUITE (master_test_suite)


BOOST_AUTO_TEST_CASE (rmol_optimisation_monte_carlo) {

  // State whether the BOM tree should be built-in or parsed from an input file
  const bool isBuiltin = false;

  BOOST_CHECK_NO_THROW (testOptimiseHelper(0, isBuiltin););
}

BOOST_AUTO_TEST_CASE (rmol_optimisation_dynamic_programming) {

  // State whether the BOM tree should be built-in or parsed from an input file
  const bool isBuiltin = false;

  BOOST_CHECK_NO_THROW (testOptimiseHelper(1, isBuiltin););
}

BOOST_AUTO_TEST_CASE (rmol_optimisation_emsr_bpv) {

  // State whether the BOM tree should be built-in or parsed from an input file
  const bool isBuiltin = false;

  BOOST_CHECK_NO_THROW (testOptimiseHelper(2, isBuiltin););
}

BOOST_AUTO_TEST_CASE (rmol_optimisation_emsr_a) {

  // State whether the BOM tree should be built-in or parsed from an input file
  const bool isBuiltin = false;

  BOOST_CHECK_NO_THROW (testOptimiseHelper(3, isBuiltin););
  // const int lBookingLimit = testOptimiseHelper(3);
  // const int lExpectedBookingLimit = 61;
  // BOOST_CHECK_EQUAL (lBookingLimit, lExpectedBookingLimit);
  // BOOST_CHECK_MESSAGE (lBookingLimit == lExpectedBookingLimit,
  //                      "The booking limit is " << lBookingLimit
  //                      << ", but it is expected to be "
  //                      << lExpectedBookingLimit);
}

BOOST_AUTO_TEST_CASE (rmol_optimisation_emsr_b) {

  // State whether the BOM tree should be built-in or parsed from an input file
  const bool isBuiltin = false;

  BOOST_CHECK_NO_THROW (testOptimiseHelper(4, isBuiltin););
}

BOOST_AUTO_TEST_CASE (rmol_optimisation_monte_carlo_built_in) {

  // State whether the BOM tree should be built-in or parsed from an input file
  const bool isBuiltin = true;

  BOOST_CHECK_NO_THROW (testOptimiseHelper(0, isBuiltin););
}
```

```
BOOST_AUTO_TEST_CASE (rmol_optimisation_dynamic_programming_built_in) {

  // State whether the BOM tree should be built-in or parsed from an input file
  const bool isBuiltin = true;

  BOOST_CHECK_NO_THROW (testOptimiseHelper(1, isBuiltin););
}

BOOST_AUTO_TEST_CASE (rmol_optimisation_emsr_bpv_built_in) {

  // State whether the BOM tree should be built-in or parsed from an input file
  const bool isBuiltin = true;

  BOOST_CHECK_NO_THROW (testOptimiseHelper(2, isBuiltin););
}

BOOST_AUTO_TEST_CASE (rmol_optimisation_emsr_a_built_in) {

  // State whether the BOM tree should be built-in or parsed from an input file
  const bool isBuiltin = true;

  BOOST_CHECK_NO_THROW (testOptimiseHelper(3, isBuiltin););
}

BOOST_AUTO_TEST_CASE (rmol_optimisation_emsr_b_built_in) {

  // State whether the BOM tree should be built-in or parsed from an input file
  const bool isBuiltin = true;

  BOOST_CHECK_NO_THROW (testOptimiseHelper(4, isBuiltin););
}

// End the test suite
BOOST_AUTO_TEST_SUITE_END()

/*!
```

10.18 Command-Line Test to Demonstrate How To Test the RMOL Project

10.18 Command-Line Test to Demonstrate How To Test the RMOL Project

```
 */
// //////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////
// STL
#include <sstream>
#include <fstream>
#include <string>
// Boost Unit Test Framework (UTF)
#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MAIN
#define BOOST_TEST_MODULE UnconstrainerTestSuite
#include <boost/test/unit_test.hpp>
// StdAir
#include <stdair/basic/BasLogParams.hpp>
#include <stdair/basic/BasDBParams.hpp>
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/service/Logger.hpp>
// RMOL
#include <rmol/RMOL_Service.hpp>

namespace boost_utf = boost::unit_test;

// (Boost) Unit Test XML Report
std::ofstream utfReportStream ("UnconstrainerTestSuite_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);

BOOST_AUTO_TEST_SUITE (master_test_suite)


BOOST_AUTO_TEST_CASE (rmol_unconstraining_em) {
  const bool lTestFlag = true;// testUnconstrainerHelper(0);
  BOOST_CHECK_EQUAL (lTestFlag, true);
  BOOST_CHECK_MESSAGE (lTestFlag == true,
                       "The test has failed. Please see the log file for "
                       << "more details");
}

// End the test suite
BOOST_AUTO_TEST_SUITE_END()

/*!
```