StdAir Reference Manual

1.00.2

Generated by Doxygen 1.4.7

Sun Jun 7 15:17:23 2015

Contents

1 StdAir Documentation

1.1 Getting Started

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

1.2 StdAir at SourceForge

- Project page
- Download StdAir
- Open a ticket fora bug or feature

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

1.3 StdAir Development

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

1.4 External Libraries

- Boost (C++ STL extensions)
- ZeroMQ (networking made easy)
- Python
- MySQL client
- SOCI (C++ DB API)

1.5 Support StdAir

1.6 About StdAir

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

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

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

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

Note:

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

2 StdAir Namespace Index

2.1 StdAir Namespace List

Here is a list of all namespaces with brief descriptions:

3 StdAir Hierarchical Index

3.1 StdAir Class Hierarchy

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

4 StdAir Class Index

4.1 StdAir Class List

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

5 StdAir File Index

5.1 StdAir File List

Here is a list of all files with brief descriptions:

6 StdAir Page Index

6.1 StdAir Related Pages

Here is a list of all related documentation pages:

7 StdAir Namespace Documentation

7.1 boost Namespace Reference

Forward declarations.

Namespaces

- namespace serialization

7.1.1 Detailed Description

Forward declarations.

7.2 boost::serialization Namespace Reference

7.3 bpt Namespace Reference

Typedefs

- typedef char ptree
- typedef char ptree
- typedef char ptree
- typedef char ptree
- typedef char ptree

7.3.1 Typedef Documentation

7.3.1.1 typedef char bpt::ptree

Definition at line 22 of file BomINIImport.cpp.

7.3.1.2 typedef char bpt::ptree

Definition at line 21 of file BomJSONExport.hpp.

7.3.1.3 typedef char bpt::ptree

Definition at line 24 of file BomJSONImport.cpp.

7.3.1.4 typedef char bpt::ptree

Definition at line 28 of file ConfigHolderStruct.hpp.

7.3.1.5 typedef char bpt::ptree

Definition at line 38 of file STDAIR_Service.cpp.

7.4 soci Namespace Reference

Classes

- struct type_conversion< stdair::AirlineStruct >

7.5 stdair Namespace Reference

Handle on the StdAir library context.

Classes

- struct BasChronometer
- struct DefaultDCPList
- struct DefaultDtdFratMap
- struct DefaultDtdProbMap
- struct DefaultMap
- struct BasDBParams

  Structure holding the parameters for connection to a database.

- struct BasFileMgr
- struct BasLogParams

  Structure holding parameters for logging.

- struct date_time_element
- struct ContinuousAttributeLite

  Class modeling the distribution of values that can be taken by a continuous attribute.

- struct DemandGenerationMethod

  Enumeration of demand (booking request) generation methods.

- class DictionaryManager

  Class wrapper of dictionary business methods.

- struct EventType
- struct ForecastingMethod
- struct JSonCommand

  Enumeration of json commands.

- struct OptimisationMethod
- struct PartnershipTechnique

  Enumeration of partnership techniques.

- struct PassengerChoiceModel
- struct PassengerType
- struct PreOptimisationMethod
- struct ProgressStatus
- struct ProgressStatusSet
- struct RandomGeneration

  Class holding a random generator.

- struct SampleType

  Enumeration of BOM sample types.

- struct ServiceInitialisationType

  Enumeration of service initialisation types.

- struct StructAbstract

  Base class for the light structures.

- struct UnconstrainingMethod
- class YieldRange
- class AirlineClassList

  Class representing the actual attributes for a segment-features.

- struct AirlineClassListKey

  Key of airport-pair.

- class AirlineFeature

  Class representing various configuration parameters (e.g., revenue management methods such EMSRb or Monte-Carlo) for a given airline for the simulation.

- struct AirlineFeatureKey
- struct AirlineStruct
- class AirportPair

  Class representing the actual attributes for an airport-pair.

- struct AirportPairKey

  Key of airport-pair.

- class BomAbstract

  Base class for the Business Object Model (BOM) layer.

- class BomArchive

  Utility class to archive/restore BOM objects with Boost serialisation.

- class BomDisplay

  Utility class to display StdAir objects with a pretty format.

- class BomHolder

  Class representing the holder of BOM object containers (list and map).

- struct BomHolderKey
- struct BomID

  Class wrapper of bom ID (e.g. pointer to object).

- class BomINIImport

  Utility class to import StdAir objects in a INI format.

- class BomJSONExport

  Utility class to export StdAir objects in a JSON format.

- class BomJSONImport

  Utility class to import StdAir objects in a JSON format.

- class BomKeyManager

  Utility class to extract key structures from strings.

- class BomManager

  Utility class for StdAir-based objects.

- class BomRetriever

  Utility class to retrieve StdAir objects.

- class BomRoot

  Class representing the actual attributes for the Bom root.

- struct BomRootKey

  Key of the BOM structure root.

- class BookingClass
- struct BookingClassKey
- struct BookingRequestStruct

  Structure holding the elements of a booking request.

- struct BreakPointStruct
- class Bucket

  Class representing the actual attributes for an airline booking class.

- struct BucketKey

  Key of booking-class.

- struct CancellationStruct

  Structure holding the elements of a travel solution.

- struct ConfigHolderStruct
- class DatePeriod

  Class representing the actual attributes for a fare date-period.

- struct DatePeriodKey

  Key of date-period.

- struct DoWStruct
- struct EventStruct
- class FareFamily

  Class representing the actual attributes for a family fare.

- struct FareFamilyKey

  Key of a given fare family, made of a fare family code.

- class FareFeatures

  Class representing the actual attributes for a fare date-period.

- struct FareFeaturesKey

  Key of date-period.

- struct FareOptionStruct

  Structure holding the elements of a fare option.

- struct FFDisutilityCurveHolderStruct
- class FlightDate

  Class representing the actual attributes for an airline flight-date.

- struct FlightDateKey

  Key of a given flight-date, made of a flight number and a departure date.

- class FlightPeriod
- struct FlightPeriodKey
- struct FRAT5CurveHolderStruct
- class Inventory

  Class representing the actual attributes for an airline inventory.

- struct InventoryKey

  Key of a given inventory, made of the airline code.

- struct KeyAbstract

  Base class for the keys of Business Object Model (BOM) layer.

- class LegCabin

  Class representing the actual attributes for an airline leg-cabin.

- struct LegCabinKey

  Key of a given leg-cabin, made of a cabin code (only).

- class LegDate
- struct LegDateKey
- class NestingNode
- struct NestingNodeKey

Key of a given policy, made of a policy code.

- struct NestingStructureKey

  Key of a given policy, made of a policy code.

- class OnDDate

  Class representing the actual attributes for an airline flight-date.

- struct OnDDateKey

  Key of a given O&D-date, made of a list of OnD strings. a OnD string contains the airline code, the flight number, the date and the segment (origin and destination).

- struct OptimisationNotificationStruct
- struct ParsedKey
- struct PeriodStruct
- class Policy
- struct PolicyKey

  Key of a given policy, made of a policy code.

- class PosChannel

  Class representing the actual attributes for a fare point of sale.

- struct PosChannelKey

  Key of point of sale and channel.

- struct RMEventStruct
- class SegmentCabin

  Class representing the actual attributes for an airline segment-cabin.

- struct SegmentCabinKey

  Key of a given segment-cabin, made of a cabin code (only).

- class SegmentDate

  Class representing the actual attributes for an airline segment-date.

- struct SegmentDateKey

  Key of a given segment-date, made of an origin and a destination airports.

- class SegmentPeriod
- struct SegmentPeriodKey
- class SegmentSnapshotTable

  Class representing the actual attributes for an airline segment data tables.

- struct SegmentSnapshotTableKey

  Key of a given guillotine block, made of a guillotine number.

- class SimpleNestingStructure
- struct SnapshotStruct
- class TimePeriod

  Class representing the actual attributes for a fare time-period.

- struct TimePeriodKey

  Key of time-period.

- struct TravelSolutionStruct

  Structure holding the elements of a travel solution.

- struct VirtualClassStruct
- class YieldFeatures

  Class representing the actual attributes for a yield date-period.

- struct YieldFeaturesKey

  Key of date-period.

- class YieldStore
- struct YieldStoreKey
- class CmdAbstract
- class CmdBomManager
- class CmdBomSerialiser
- class CmdCloneBomManager
- class DBManagerForAirlines
- class DbaAbstract
- class FacAbstract
- class FacBom

  Base class for Factory layer.

- class FacBomManager

  Utility class for linking StdAir-based objects.

- class FacCloneBom

  Base class for Factory layer.

- class DBSessionManager
- class FacServiceAbstract
- class FacSTDAIRServiceContext

  Factory for Bucket.

- class FacSupervisor
- class Logger
- class ServiceAbstract
- class STDAIR_ServiceContext

  Class holding the context of the Stdair services.

- class RootException

  Root of the stdair exceptions.

- class FileNotFoundException
- class NonInitialisedLogServiceException
- class NonInitialisedServiceException
- class NonInitialisedContainerException

- class NonInitialisedRelationShipException
- class MemoryAllocationException
- class ObjectLinkingException
- class DocumentNotFoundException
- class ParserException
- class SerialisationException
- class KeyNotFoundException
- class CodeConversionException
- class CodeDuplicationException
- class KeyDuplicationException
- class ObjectCreationgDuplicationException
- class ObjectNotFoundException
- class ParsingFileFailedException
- class SQLDatabaseException
- class NonInitialisedDBSessionManagerException
- class SQLDatabaseConnectionImpossibleException
- class EventException
- class SimpleNestingStructException
- class BookingClassListEmptyInNestingStructException
- class RootFilePath

  Root of the input and output files.

- class InputFilePath
- class ScheduleFilePath
- class ODFilePath
- class FRAT5FilePath
- class FFDisutilityFilePath
- class ConfigINIFile
- class JSONString

  JSON-formatted string.

- class STDAIR_Service

  Interface for the STDAIR Services.

Namespaces

- namespace LOG

Typedefs

- typedef date_time_element< 0, 23 > hour_t
- typedef date_time_element< 0, 59 > minute_t
- typedef date_time_element< 0, 59 > second_t
- typedef date_time_element< 1900, 2100 > year_t
- typedef date_time_element< 1, 12 > month_t
- typedef date_time_element< 1, 31 > day_t
- typedef std::istreambuf_iterator< char > base_iterator_t
- typedef boost::spirit::multi_pass< base_iterator_t > iterator_t
- typedef boost::spirit::qi::int_parser< unsigned int, 10, 1, 1 > int1_p_t

- typedef boost::spirit::qi::uint_parser< int, 10, 2, 2 > uint2_p_t
- typedef boost::spirit::qi::uint_parser< int, 10, 4, 4 > uint4_p_t
- typedef boost::spirit::qi::uint_parser< int, 10, 1, 4 > uint1_4_p_t
- typedef boost::spirit::qi::uint_parser< hour_t, 10, 2, 2 > hour_p_t
- typedef boost::spirit::qi::uint_parser< minute_t, 10, 2, 2 > minute_p_t
- typedef boost::spirit::qi::uint_parser< second_t, 10, 2, 2 > second_p_t
- typedef boost::spirit::qi::uint_parser< year_t, 10, 4, 4 > year_p_t
- typedef boost::spirit::qi::uint_parser< month_t, 10, 2, 2 > month_p_t
- typedef boost::spirit::qi::uint_parser< day_t, 10, 2, 2 > day_p_t
- typedef unsigned short DictionaryKey_T
- typedef std::list< AirlineClassList * > AirlineClassListList_T
- typedef std::map< const MapKey_T, AirlineClassList * > AirlineClassListMap_T
- typedef std::pair< MapKey_T, AirlineClassList * > AirlineClassListWithKey_T
- typedef std::list< AirlineClassListWithKey_T > AirlineClassListDetailedList_T
- typedef std::list< AirlineFeature * > AirlineFeatureList_T
- typedef std::map< const MapKey_T, AirlineFeature * > AirlineFeatureMap_T
- typedef std::list< AirportPair * > AirportPairList_T
- typedef std::map< const MapKey_T, AirportPair * > AirportPairMap_T
- typedef std::pair< MapKey_T, AirportPair * > AirportPairWithKey_T
- typedef std::list< AirportPairWithKey_T > AirportPairDetailedList_T
- typedef std::map< const std::type_info *, BomAbstract * > HolderMap_T
- typedef BomID< BookingClass > BookingClassID_T
- typedef std::list< BookingClassID_T > BookingClassIDList_T
- typedef boost::tokenizer< boost::char_separator< char > > Tokeniser_T
- typedef std::list< BookingClass * > BookingClassList_T
- typedef std::map< const MapKey_T, BookingClass * > BookingClassMap_T
- typedef boost::shared_ptr< BookingRequestStruct > BookingRequestPtr_T
- typedef std::string DemandGeneratorKey_T
- typedef boost::shared_ptr< BreakPointStruct > BreakPointPtr_T
- typedef std::list< BreakPointStruct > BreakPointList_T
- typedef std::list< Bucket * > BucketList_T
- typedef std::map< const MapKey_T, Bucket * > BucketMap_T
- typedef boost::shared_ptr< CancellationStruct > CancellationPtr_T
- typedef boost::shared_ptr< ConfigHolderStruct > ConfigHolderPtr_T
- typedef std::list< DatePeriod * > DatePeriodList_T
- typedef std::map< const MapKey_T, DatePeriod * > DatePeriodMap_T
- typedef std::pair< MapKey_T, DatePeriod * > DatePeriodWithKey_T
- typedef std::list< DatePeriodWithKey_T > DatePeriodDetailedList_T
- typedef std::pair< const LongDuration_T, EventStruct > EventListElement_T
- typedef std::map< const LongDuration_T, EventStruct > EventList_T
- typedef std::list< FareFamily * > FareFamilyList_T
- typedef std::map< const MapKey_T, FareFamily * > FareFamilyMap_T
- typedef std::list< FareFeatures * > FareFeaturesList_T
- typedef std::map< const MapKey_T, FareFeatures * > FareFeaturesMap_T
- typedef std::pair< MapKey_T, FareFeatures * > FareFeaturesWithKey_T
- typedef std::list< FareFeaturesWithKey_T > FareFeaturesDetailedList_T
- typedef std::list< FareOptionStruct > FareOptionList_T
- typedef std::map< const std::string, FFDisutilityCurve_T > FFDisutilityCurveHolder_T
- typedef std::list< FlightDate * > FlightDateList_T
- typedef std::map< const MapKey_T, FlightDate * > FlightDateMap_T

- typedef std::list< FlightPeriod * > FlightPeriodList_T
- typedef std::map< const MapKey_T, FlightPeriod * > FlightPeriodMap_T
- typedef std::map< const std::string, FRAT5Curve_T > FRAT5CurveHolder_T
- typedef std::list< Inventory * > InventoryList_T
- typedef std::map< const MapKey_T, Inventory * > InventoryMap_T
- typedef std::string MapKey_T
- typedef std::list< std::string > KeyList_T
- typedef std::list< LegCabin * > LegCabinList_T
- typedef std::map< const MapKey_T, LegCabin * > LegCabinMap_T
- typedef std::list< LegDate * > LegDateList_T
- typedef std::map< const MapKey_T, LegDate * > LegDateMap_T
- typedef std::list< NestingNode * > NestingNodeList_T
- typedef std::map< const MapKey_T, NestingNode * > NestingNodeMap_T
- typedef std::list< OnDDate * > OnDDateList_T
- typedef std::map< const MapKey_T, OnDDate * > OnDDateMap_T
- typedef std::pair< std::string, YieldDemandPair_T > StringDemandStructPair_T
- typedef std::map< std::string, YieldDemandPair_T > StringDemandStructMap_T
- typedef std::map< std::string, CabinClassPairList_T > StringCabinClassPairListMap_T
- typedef std::pair< std::string, CabinClassPairList_T > StringCabinClassPair_T
- typedef std::map< CabinCode_T, WTPDemandPair_T > CabinForecastMap_T
- typedef std::pair< CabinCode_T, WTPDemandPair_T > CabinForecastPair_T
- typedef boost::shared_ptr< OptimisationNotificationStruct > OptimisationNotificationPtr_T
- typedef boost::tokenizer< boost::char_separator< char > > Tokeniser_T
- typedef std::list< Policy * > PolicyList_T
- typedef std::map< const MapKey_T, Policy * > PolicyMap_T
- typedef std::list< PosChannel * > PosChannelList_T
- typedef std::map< const MapKey_T, PosChannel * > PosChannelMap_T
- typedef std::pair< MapKey_T, PosChannel * > PosChannelWithKey_T
- typedef std::list< PosChannelWithKey_T > PosChannelDetailedList_T
- typedef boost::shared_ptr< RMEventStruct > RMEventPtr_T
- typedef std::list< RMEventStruct > RMEventList_T
- typedef std::list< SegmentCabin * > SegmentCabinList_T
- typedef std::map< const MapKey_T, SegmentCabin * > SegmentCabinMap_T
- typedef std::list< std::string > RoutingLegKeyList_T
- typedef std::list< SegmentDate * > SegmentDateList_T
- typedef std::map< const MapKey_T, SegmentDate * > SegmentDateMap_T
- typedef std::list< SegmentPeriod * > SegmentPeriodList_T
- typedef std::map< const MapKey_T, SegmentPeriod * > SegmentPeriodMap_T
- typedef std::pair< MapKey_T, SegmentPeriod * > SegmentPeriodWithKey_T
- typedef std::list< SegmentPeriodWithKey_T > SegmentPeriodDetailedList_T
- typedef std::list< SegmentSnapshotTable * > SegmentSnapshotTableList_T
- typedef std::map< const MapKey_T, SegmentSnapshotTable * > SegmentSnapshotTableMap_T
- typedef std::map< const SegmentCabin *, SegmentDataID_T > SegmentCabinIndexMap_T
- typedef std::map< const MapKey_T, ClassIndex_T > ClassIndexMap_T
- typedef std::list< SimpleNestingStructure * > SimpleNestingStructureList_T
- typedef std::map< const MapKey_T, SimpleNestingStructure * > SimpleNestingStructureMap_T
- typedef boost::shared_ptr< SnapshotStruct > SnapshotPtr_T
- typedef std::list< TimePeriod * > TimePeriodList_T
- typedef std::map< const MapKey_T, TimePeriod * > TimePeriodMap_T
- typedef std::pair< MapKey_T, TimePeriod * > TimePeriodWithKey_T

- typedef std::list< TimePeriodWithKey_T > TimePeriodDetailedList_T
- typedef std::list< TravelSolutionStruct > TravelSolutionList_T
- typedef KeyList_T SegmentPath_T
- typedef std::list< SegmentPath_T > SegmentPathList_T
- typedef std::map< const ClassCode_T, Availability_T > ClassAvailabilityMap_T
- typedef std::list< ClassAvailabilityMap_T > ClassAvailabilityMapHolder_T
- typedef std::map< const ClassCode_T, BookingClassID_T > ClassObjectIDMap_T
- typedef std::list< ClassObjectIDMap_T > ClassObjectIDMapHolder_T
- typedef std::map< const ClassCode_T, YieldValue_T > ClassYieldMap_T
- typedef std::list< ClassYieldMap_T > ClassYieldMapHolder_T
- typedef std::list< BidPriceVector_T > BidPriceVectorHolder_T
- typedef std::map< const ClassCode_T, const BidPriceVector_T * > ClassBpvMap_T
- typedef std::list< ClassBpvMap_T > ClassBpvMapHolder_T
- typedef std::list< VirtualClassStruct > VirtualClassList_T
- typedef std::map< const YieldLevel_T, VirtualClassStruct > VirtualClassMap_T
- typedef std::list< YieldFeatures * > YieldFeaturesList_T
- typedef std::map< const MapKey_T, YieldFeatures * > YieldFeaturesMap_T
- typedef std::pair< MapKey_T, YieldFeatures * > YieldFeaturesWithKey_T
- typedef std::list< YieldFeaturesWithKey_T > YieldFeaturesDetailedList_T
- typedef std::list< YieldStore * > YieldStoreList_T
- typedef std::map< const MapKey_T, YieldStore * > YieldStoreMap_T
- typedef std::string LocationCode_T
- typedef unsigned long int Distance_T
- typedef LocationCode_T AirportCode_T
- typedef LocationCode_T CityCode_T
- typedef std::string KeyDescription_T
- typedef std::string AirlineCode_T
- typedef unsigned short FlightNumber_T
- typedef unsigned short TableID_T
- typedef std::string CabinCode_T
- typedef std::string FamilyCode_T
- typedef std::string PolicyCode_T
- typedef std::string NestingStructureCode_T
- typedef std::string NestingNodeCode_T
- typedef std::string ClassCode_T
- typedef unsigned long Identity_T
- typedef std::string TripType_T
- typedef double MonetaryValue_T
- typedef double RealNumber_T
- typedef double Percentage_T
- typedef double PriceValue_T
- typedef double YieldValue_T
- typedef std::string PriceCurrency_T
- typedef double Revenue_T
- typedef double Multiplier_T
- typedef double NbOfSeats_T
- typedef unsigned int Count_T
- typedef short PartySize_T
- typedef double NbOfRequests_T
- typedef NbOfRequests_T NbOfBookings_T

- typedef NbOfRequests_T NbOfCancellations_T
- typedef unsigned short NbOfTravelSolutions_T
- typedef std::string ClassList_String_T
- typedef unsigned short NbOfSegments_T
- typedef unsigned short NbOfAirlines_T
- typedef double Availability_T
- typedef double Fare_T
- typedef bool Flag_T
- typedef unsigned int UnsignedIndex_T
- typedef unsigned int NbOfClasses_T
- typedef unsigned int NbOfFareFamilies_T
- typedef std::string Filename_T
- typedef std::string FileAddress_T
- typedef float ProgressPercentage_T
- typedef boost::posix_time::time_duration Duration_T
- typedef boost::gregorian::date Date_T
- typedef boost::posix_time::time_duration Time_T
- typedef boost::posix_time::ptime DateTime_T
- typedef boost::gregorian::date_period DatePeriod_T
- typedef std::string DOW_String_T
- typedef boost::gregorian::date_duration DateOffset_T
- typedef int DayDuration_T
- typedef bool SaturdayStay_T
- typedef long int IntDuration_T
- typedef long long int LongDuration_T
- typedef float FloatDuration_T
- typedef soci::session DBSession_T
- typedef soci::statement DBRequestStatement_T
- typedef std::string DBConnectionName_T
- typedef bool ChangeFees_T
- typedef bool NonRefundable_T
- typedef bool SaturdayStay_T
- typedef double SaturdayStayRatio_T
- typedef double ChangeFeesRatio_T
- typedef double NonRefundableRatio_T
- typedef double Disutility_T
- typedef std::string PassengerType_T
- typedef std::string DistributionPatternId_T
- typedef std::string CancellationRateCurveId_T
- typedef std::string AirlinePreferenceId_T
- typedef std::pair< Percentage_T, Percentage_T > CancellationNoShowRatePair_T
- typedef std::string CharacteristicsPatternId_T
- typedef std::string CharacteristicsIndex_T
- typedef double WTP_T
- typedef boost::tuples::tuple< double, WTP_T > CharacteristicsWTP_tuple_T
- typedef std::pair< WTP_T, MeanStdDevPair_T > WTPDemandPair_T
- typedef NbOfRequests_T NbOfCancellations_T
- typedef NbOfRequests_T NbOfNoShows_T
- typedef double MatchingIndicator_T
- typedef std::string DemandStreamKeyStr_T

- typedef std::string ChannelLabel_T
- typedef std::string FrequentFlyer_T
- typedef std::string RequestStatus_T
- typedef std::map< Identity_T, Identity_T > BookingTSIDMap_T
- typedef std::pair< CabinCode_T, ClassCode_T > CabinClassPair_T
- typedef std::list< CabinClassPair_T > CabinClassPairList_T
- typedef double ProportionFactor_T
- typedef std::list< ProportionFactor_T > ProportionFactorList_T
- typedef std::string OnDString_T
- typedef std::list< OnDString_T > OnDStringList_T
- typedef std::string EventName_T
- typedef double NbOfEvents_T
- typedef std::string EventGeneratorKey_T
- typedef double NbOfFareRules_T
- typedef std::string NetworkID_T
- typedef std::vector< AirlineCode_T > AirlineCodeList_T
- typedef std::vector< ClassList_String_T > ClassList_StringList_T
- typedef std::vector< ClassCode_T > ClassCodeList_T
- typedef unsigned short SubclassCode_T
- typedef std::string FlightPathCode_T
- typedef std::map< CabinCode_T, ClassList_String_T > CabinBookingClassMap_T
- typedef std::string CurveKey_T
- typedef double CabinCapacity_T
- typedef double NbOfFlightDates_T
- typedef double CommittedSpace_T
- typedef double UPR_T
- typedef double BookingLimit_T
- typedef double AuthorizationLevel_T
- typedef double CapacityAdjustment_T
- typedef double BlockSpace_T
- typedef bool AvailabilityStatus_T
- typedef std::vector< Availability_T > BucketAvailabilities_T
- typedef double NbOfYields_T
- typedef double NbOfInventoryControlRules_T
- typedef bool CensorshipFlag_T
- typedef short DTD_T
- typedef short DCP_T
- typedef std::list< DCP_T > DCPList_T
- typedef std::map< DTD_T, RealNumber_T > DTDFratMap_T
- typedef std::map< FloatDuration_T, float > DTDProbMap_T
- typedef std::vector< CensorshipFlag_T > CensorshipFlagList_T
- typedef double BookingRatio_T
- typedef double Yield_T
- typedef unsigned int YieldLevel_T
- typedef std::map< YieldLevel_T, MeanStdDevPair_T > YieldLevelDemandMap_T
- typedef std::pair< Yield_T, MeanStdDevPair_T > YieldDemandPair_T
- typedef double BidPrice_T
- typedef std::vector< BidPrice_T > BidPriceVector_T
- typedef unsigned int SeatIndex_T
- typedef std::string ControlMode_T

- typedef double OverbookingRate_T
- typedef double BookingLimit_T
- typedef double ProtectionLevel_T
- typedef std::vector< double > EmsrValueList_T
- typedef std::vector< double > BookingLimitVector_T
- typedef std::vector< double > ProtectionLevelVector_T
- typedef boost::multi_array< double, 2 > SnapshotBlock_T
- typedef SnapshotBlock_T::index_range SnapshotBlockRange_T
- typedef SnapshotBlock_T::array_view< 1 >::type SegmentCabinDTDSnapshotView_T
- typedef SnapshotBlock_T::array_view< 2 >::type SegmentCabinDTDRangeSnapshotView_T
- typedef SnapshotBlock_T::const_array_view< 1 >::type ConstSegmentCabinDTDSnapshotView_T
- typedef SnapshotBlock_T::const_array_view< 2 >::type ConstSegmentCabinDTDRangeSnapshotView_T
- typedef unsigned short SegmentDataID_T
- typedef unsigned short LegDataID_T
- typedef unsigned short ClassIndex_T
- typedef unsigned int ReplicationNumber_T
- typedef unsigned long int ExponentialSeed_T
- typedef unsigned long int UniformSeed_T
- typedef unsigned long int RandomSeed_T
- typedef boost::minstd_rand BaseGenerator_T
- typedef boost::uniform_real UniformDistribution_T
- typedef boost::variate_generator< BaseGenerator_T &, UniformDistribution_T > UniformGenerator_T
- typedef boost::normal_distribution NormalDistribution_T
- typedef boost::variate_generator< BaseGenerator_T &, NormalDistribution_T > NormalGenerator_T
- typedef boost::exponential_distribution ExponentialDistribution_T
- typedef boost::variate_generator< BaseGenerator_T &, ExponentialDistribution_T > ExponentialGenerator_T
- typedef double MeanValue_T
- typedef double StdDevValue_T
- typedef std::pair< MeanValue_T, StdDevValue_T > MeanStdDevPair_T
- typedef std::vector< MeanStdDevPair_T > MeanStdDevPairVector_T
- typedef float Probability_T
- typedef std::string ForecasterMode_T
- typedef short HistoricalDataLimit_T
- typedef std::string OptimizerMode_T
- typedef NbOfBookings_T PolicyDemand_T
- typedef std::vector< double > GeneratedDemandVector_T
- typedef std::vector< GeneratedDemandVector_T > GeneratedDemandVectorHolder_T
- typedef double SellupProbability_T
- typedef std::vector< NbOfRequests_T > UncDemVector_T
- typedef std::vector< NbOfBookings_T > BookingVector_T
- typedef double FRAT5_T
- typedef std::map< const DTD_T, FRAT5_T > FRAT5Curve_T
- typedef std::map< const DTD_T, double > FFDisutilityCurve_T
- typedef std::map< const DTD_T, double > SellUpCurve_T
- typedef std::map< const DTD_T, double > DispatchingCurve_T

- typedef std::map< BookingClass *, SellUpCurve_T > BookingClassSellUpCurveMap_T
- typedef std::map< BookingClass *, DispatchingCurve_T > BookingClassDispatchingCurveMap_T
- typedef std::map< const Yield_T, double > YieldDemandMap_T
- typedef double Revenue_T
- typedef unsigned int NbOfSamples_T
- typedef boost::shared_ptr< STDAIR_Service > STDAIR_ServicePtr_T

Functions

- const std::string DEFAULT_BOM_ROOT_KEY (" – ROOT – ")
- const double DEFAULT_EPSILON_VALUE (0.0001)
- const unsigned int DEFAULT_FLIGHT_SPEED (900)
- const NbOfFlightDates_T DEFAULT_NB_OF_FLIGHTDATES (0.0)
- const Duration_T NULL_BOOST_TIME_DURATION (-1,-1,-1)
- const Duration_T DEFAULT_NULL_DURATION (0, 0, 0)
- const unsigned int DEFAULT_NB_OF_DAYS_IN_A_YEAR (365)
- const unsigned int DEFAULT_NUMBER_OF_SUBDIVISIONS (1000)
- const DayDuration_T DEFAULT_DAY_DURATION (0)
- const DatePeriod_T BOOST_DEFAULT_DATE_PERIOD (Date_T(2007, 1, 1), Date_T(2007, 1, 1))
- const DOW_String_T DEFAULT_DOW_STRING ("0000000")
- const DateOffset_T DEFAULT_DATE_OFFSET (0)
- const Date_T DEFAULT_DATE (2010, boost::gregorian::Jan, 1)
- const DateTime_T DEFAULT_DATETIME (DEFAULT_DATE, NULL_BOOST_TIME_DURATION)
- const Duration_T DEFAULT_EPSILON_DURATION (0, 0, 0, 1)
- const Count_T SECONDS_IN_ONE_DAY (86400)
- const Count_T MILLISECONDS_IN_ONE_SECOND (1000)
- const RandomSeed_T DEFAULT_RANDOM_SEED (120765987)
- const AirportCode_T AIRPORT_LHR ("LHR")
- const AirportCode_T AIRPORT_SYD ("SYD")
- const CityCode_T POS_LHR ("LHR")
- const Date_T DATE_20110115 (2011, boost::gregorian::Jan, 15)
- const Date_T DATE_20111231 (2011, boost::gregorian::Dec, 31)
- const DayDuration_T NO_ADVANCE_PURCHASE (0)
- const SaturdayStay_T SATURDAY_STAY (true)
- const SaturdayStay_T NO_SATURDAY_STAY (false)
- const ChangeFees_T CHANGE_FEES (true)
- const ChangeFees_T NO_CHANGE_FEES (false)
- const NonRefundable_T NON_REFUNDABLE (true)
- const NonRefundable_T NO_NON_REFUNDABLE (false)
- const SaturdayStay_T DEFAULT_BOM_TREE_SATURDAY_STAY (true)
- const ChangeFees_T DEFAULT_BOM_TREE_CHANGE_FEES (true)
- const NonRefundable_T DEFAULT_BOM_TREE_NON_REFUNDABLE (true)
- const DayDuration_T NO_STAY_DURATION (0)
- const AirlineCode_T AIRLINE_CODE_BA ("BA")
- const CabinCode_T CABIN_Y ("Y")
- const ClassCode_T CLASS_CODE_Y ("Y")
- const ClassCode_T CLASS_CODE_Q ("Q")
- const AirportCode_T AIRPORT_SIN ("SIN")
- const AirportCode_T AIRPORT_BKK ("BKK")

- const CityCode_T POS_SIN ("SIN")
- const CabinCode_T CABIN_ECO ("Eco")
- const FrequentFlyer_T FREQUENT_FLYER_MEMBER ("M")
- const FamilyCode_T DEFAULT_FAMILY_CODE ("0")
- const PolicyCode_T DEFAULT_POLICY_CODE ("0")
- const NestingStructureCode_T DEFAULT_NESTING_STRUCTURE_CODE ("DEFAULT")
- const NestingStructureCode_T DISPLAY_NESTING_STRUCTURE_CODE ("Display Nesting")
- const NestingStructureCode_T YIELD_BASED_NESTING_STRUCTURE_CODE ("Yield-Based Nesting")
- const NestingNodeCode_T DEFAULT_NESTING_NODE_CODE ("0")
- const NbOfAirlines_T DEFAULT_NBOFAIRLINES (0)
- const FlightPathCode_T DEFAULT_FLIGHTPATH_CODE ("")
- const Distance_T DEFAULT_DISTANCE_VALUE (0)
- const ClassCode_T DEFAULT_CLOSED_CLASS_CODE ("CC")
- const NbOfBookings_T DEFAULT_CLASS_NB_OF_BOOKINGS (0)
- const NbOfBookings_T DEFAULT_CLASS_TOTAL_NB_OF_BOOKINGS (0)
- const NbOfBookings_T DEFAULT_CLASS_UNCONSTRAINED_DEMAND (0)
- const NbOfBookings_T DEFAULT_CLASS_REMAINING_DEMAND_MEAN (0)
- const NbOfBookings_T DEFAULT_CLASS_REMAINING_DEMAND_STANDARD_-DEVIATION (0)
- const NbOfCancellations_T DEFAULT_CLASS_NB_OF_CANCELLATIONS (0)
- const NbOfNoShows_T DEFAULT_CLASS_NB_OF_NOSHOWS (0)
- const CabinCapacity_T DEFAULT_CABIN_CAPACITY (100.0)
- const CommittedSpace_T DEFAULT_COMMITTED_SPACE (0.0)
- const BlockSpace_T DEFAULT_BLOCK_SPACE (0.0)
- const Availability_T DEFAULT_NULL_AVAILABILITY (0.0)
- const Availability_T DEFAULT_AVAILABILITY (9.0)
- const Availability_T MAXIMAL_AVAILABILITY (9999.0)
- const CensorshipFlag_T DEFAULT_CLASS_CENSORSHIPFLAG (false)
- const BookingLimit_T DEFAULT_CLASS_BOOKING_LIMIT (9999.0)
- const AuthorizationLevel_T DEFAULT_CLASS_AUTHORIZATION_LEVEL (9999.0)
- const AuthorizationLevel_T DEFAULT_CLASS_MAX_AUTHORIZATION_LEVEL (9999.0)
- const AuthorizationLevel_T DEFAULT_CLASS_MIN_AUTHORIZATION_LEVEL (0.0)
- const OverbookingRate_T DEFAULT_CLASS_OVERBOOKING_RATE (0.0)
- const BookingRatio_T DEFAULT_OND_BOOKING_RATE (0.0)
- const Fare_T DEFAULT_FARE_VALUE (0.0)
- const Yield_T DEFAULT_CLASS_YIELD_VALUE (0.0)
- const Revenue_T DEFAULT_REVENUE_VALUE (0.0)
- const Percentage_T DEFAULT_LOAD_FACTOR_VALUE (100.0)
- const Yield_T DEFAULT_YIELD_VALUE (0.0)
- const Yield_T DEFAULT_YIELD_MAX_VALUE (std::numeric_limits< double >::max())
- const NbOfBookings_T DEFAULT_YIELD_NB_OF_BOOKINGS (0.0)
- const Identity_T DEFAULT_BOOKING_NUMBER (0)
- const NbOfCancellations_T DEFAULT_YIELD_NB_OF_CANCELLATIONS (0.0)
- const NbOfNoShows_T DEFAULT_YIELD_NB_OF_NOSHOWS (0.0)
- const Availability_T DEFAULT_YIELD_AVAILABILITY (0.0)
- const CensorshipFlag_T DEFAULT_YIELD_CENSORSHIPFLAG (false)
- const BookingLimit_T DEFAULT_YIELD_BOOKING_LIMIT (0.0)
- const OverbookingRate_T DEFAULT_YIELD_OVERBOOKING_RATE (0.0)
- const Fare_T DEFAULT_OND_FARE_VALUE (0.0)

- const Count_T DEFAULT_PROGRESS_STATUS (0)
- const Percentage_T MAXIMUM_PROGRESS_STATUS (100)
- const Date_T DEFAULT_EVENT_OLDEST_DATE (2008, boost::gregorian::Jan, 1)
- const DateTime_T DEFAULT_EVENT_OLDEST_DATETIME (DEFAULT_EVENT_OLDEST_-DATE, NULL_BOOST_TIME_DURATION)
- const PartySize_T DEFAULT_PARTY_SIZE (1)
- const DayDuration_T DEFAULT_STAY_DURATION (7)
- const WTP_T DEFAULT_WTP (1000.0)
- const Date_T DEFAULT_PREFERRED_DEPARTURE_DATE (DEFAULT_DEPARTURE_DATE)
- const Duration_T DEFAULT_PREFERRED_DEPARTURE_TIME (8, 0, 0)
- const DateOffset_T DEFAULT_ADVANCE_PURCHASE (22)
- const Date_T DEFAULT_REQUEST_DATE (DEFAULT_PREFERRED_DEPARTURE_DATE-DEFAULT_ADVANCE_PURCHASE)
- const Duration_T DEFAULT_REQUEST_TIME (8, 0, 0)
- const DateTime_T DEFAULT_REQUEST_DATE_TIME (DEFAULT_REQUEST_DATE, DEFAULT_REQUEST_TIME)
- const CabinCode_T DEFAULT_PREFERRED_CABIN ("M")
- const CityCode_T DEFAULT_POS ("ALL")
- const ChannelLabel_T DEFAULT_CHANNEL ("DC")
- const ChannelLabel_T CHANNEL_DN ("DN")
- const ChannelLabel_T CHANNEL_IN ("IN")
- const TripType_T TRIP_TYPE_ONE_WAY ("OW")
- const TripType_T TRIP_TYPE_ROUND_TRIP ("RT")
- const TripType_T TRIP_TYPE_INBOUND ("RI")
- const TripType_T TRIP_TYPE_OUTBOUND ("RO")
- const FrequentFlyer_T DEFAULT_FF_TIER ("N")
- const PriceValue_T DEFAULT_VALUE_OF_TIME (100.0)
- const IntDuration_T HOUR_CONVERTED_IN_SECONDS (3600)
- const Duration_T DEFAULT_MINIMAL_CONNECTION_TIME (0, 30, 0)
- const Duration_T DEFAULT_MAXIMAL_CONNECTION_TIME (24, 0, 0)
- const MatchingIndicator_T DEFAULT_MATCHING_INDICATOR (0.0)
- const PriceCurrency_T DEFAULT_CURRENCY ("EUR")
- const AvailabilityStatus_T DEFAULT_AVAILABILITY_STATUS (false)
- const AirlineCode_T DEFAULT_AIRLINE_CODE ("XX")
- const AirlineCode_T DEFAULT_NULL_AIRLINE_CODE ("")
- const FlightNumber_T DEFAULT_FLIGHT_NUMBER (9999)
- const FlightNumber_T DEFAULT_FLIGHT_NUMBER_FF (255)
- const TableID_T DEFAULT_TABLE_ID (9999)
- const Date_T DEFAULT_DEPARTURE_DATE (1900, boost::gregorian::Jan, 1)
- const AirportCode_T DEFAULT_AIRPORT_CODE ("XXX")
- const AirportCode_T DEFAULT_NULL_AIRPORT_CODE ("")
- const AirportCode_T DEFAULT_ORIGIN ("XXX")
- const AirportCode_T DEFAULT_DESTINATION ("YYY")
- const CabinCode_T DEFAULT_CABIN_CODE ("X")
- const FamilyCode_T DEFAULT_FARE_FAMILY_CODE ("EcoSaver")
- const FamilyCode_T DEFAULT_NULL_FARE_FAMILY_CODE ("NoFF")
- const ClassCode_T DEFAULT_CLASS_CODE ("X")
- const ClassCode_T DEFAULT_NULL_CLASS_CODE ("")
- const BidPrice_T DEFAULT_BID_PRICE (0.0)
- const unsigned short MAXIMAL_NUMBER_OF_LEGS_IN_FLIGHT (7)

- const unsigned short MAXIMAL_NUMBER_OF_SEGMENTS_IN_OND (3)
- const SeatIndex_T DEFAULT_SEAT_INDEX (1)
- const NbOfSeats_T DEFAULT_NULL_BOOKING_NUMBER (0)
- const CapacityAdjustment_T DEFAULT_NULL_CAPACITY_ADJUSTMENT (0)
- const UPR_T DEFAULT_NULL_UPR (0)
- const std::string DEFAULT_FARE_FAMILY_VALUE_TYPE ("FF")
- const std::string DEFAULT_SEGMENT_CABIN_VALUE_TYPE ("SC")
- const std::string DEFAULT_KEY_FLD_DELIMITER (";")
- const std::string DEFAULT_KEY_SUB_FLD_DELIMITER (",")
- const boost::char_separator< char > DEFAULT_KEY_TOKEN_DELIMITER (";, ")
- template<int MIN, int MAX> date_time_element< MIN, MAX > operator * (const date_time_element< MIN, MAX > &o1, const date_time_element< MIN, MAX > &o2)
- template<int MIN, int MAX> date_time_element< MIN, MAX > operator+ (const date_time_element< MIN, MAX > &o1, const date_time_element< MIN, MAX > &o2)
- template void AirlineClassListKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void AirlineClassListKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void BomRootKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void BomRootKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- void intDisplay (std::ostream &oStream, const int &iInt)
- template void BucketKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void BucketKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void FareFamilyKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void FareFamilyKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void FlightDateKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void FlightDateKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void InventoryKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void InventoryKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void NestingNodeKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void NestingNodeKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void NestingStructureKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void NestingStructureKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void OnDDateKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void OnDDateKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- const boost::char_separator< char > TokeniserDashSeparator ("-")
- const boost::char_separator< char > TokeniserTimeSeparator (":")
- template void PolicyKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void PolicyKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void SegmentCabinKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void SegmentCabinKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void SegmentDateKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void SegmentDateKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void SegmentSnapshotTableKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void SegmentSnapshotTableKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template<class Archive, class BOM_OBJECT1, class BOM_OBJECT2> void serialiseHelper (BOM_OBJECT1 &ioObject1, Archive &ioArchive, const unsigned int iFileVersion)

- template void BomRoot::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void BomRoot::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void Inventory::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void Inventory::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void FlightDate::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void FlightDate::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void SegmentDate::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void SegmentDate::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void SegmentCabin::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void SegmentCabin::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)

Variables

- const std::string DOW_STR [ ]
- const UnconstrainingMethod DEFAULT_UNCONSTRAINING_METHOD ('E')
- const PartnershipTechnique DEFAULT_PARTNERSHIP_TECHNIQUE ('N')
- const ForecastingMethod DEFAULT_FORECASTING_METHOD ('Q')
- const PreOptimisationMethod DEFAULT_PREOPTIMISATION_METHOD ('N')
- const OptimisationMethod DEFAULT_OPTIMISATION_METHOD ('M')
- const CensorshipFlagList_T DEFAULT_CLASS_CENSORSHIPFLAG_LIST
- const Date_T DEFAULT_DICO_STUDIED_DATE
- const AirlineCodeList_T DEFAULT_AIRLINE_CODE_LIST
- const ClassList_StringList_T DEFAULT_CLASS_CODE_LIST
- const BidPriceVector_T DEFAULT_BID_PRICE_VECTOR = std::vector<BidPrice_T>()
- const int DEFAULT_MAX_DTD = 365
- const DCPList_T DEFAULT_DCP_LIST = DefaultDCPList::init()
- const FRAT5Curve_T FRAT5_CURVE_A
- const FRAT5Curve_T FRAT5_CURVE_B
- const FRAT5Curve_T FRAT5_CURVE_C
- const FRAT5Curve_T FRAT5_CURVE_D
- const FFDisutilityCurve_T FF_DISUTILITY_CURVE_A
- const FFDisutilityCurve_T FF_DISUTILITY_CURVE_B
- const FFDisutilityCurve_T FF_DISUTILITY_CURVE_C
- const FFDisutilityCurve_T FF_DISUTILITY_CURVE_D
- const FFDisutilityCurve_T FF_DISUTILITY_CURVE_E
- const FFDisutilityCurve_T FF_DISUTILITY_CURVE_F
- const DTDFratMap_T DEFAULT_DTD_FRAT5COEF_MAP
- const DTDProbMap_T DEFAULT_DTD_PROB_MAP
- const OnDStringList_T DEFAULT_OND_STRING_LIST
- const std::string DISPLAY_LEVEL_STRING_ARRAY [51]
- const std::string DISPLAY_LEVEL_STRING_ARRAY [51]
- const std::string DEFAULT_KEY_FLD_DELIMITER
- const std::string DEFAULT_KEY_SUB_FLD_DELIMITER
- const boost::char_separator< char > DEFAULT_KEY_TOKEN_DELIMITER
- const Distance_T DEFAULT_DISTANCE_VALUE
- const ClassCode_T DEFAULT_CLOSED_CLASS_CODE
- const NbOfBookings_T DEFAULT_CLASS_NB_OF_BOOKINGS
- const NbOfBookings_T DEFAULT_CLASS_TOTAL_NB_OF_BOOKINGS
- const NbOfBookings_T DEFAULT_CLASS_UNCONSTRAINED_DEMAND
- const NbOfBookings_T DEFAULT_CLASS_REMAINING_DEMAND_MEAN

- const NbOfBookings_T DEFAULT_CLASS_REMAINING_DEMAND_STANDARD_-DEVIATION
- const NbOfCancellations_T DEFAULT_CLASS_NB_OF_CANCELLATIONS
- const NbOfNoShows_T DEFAULT_CLASS_NB_OF_NOSHOWS
- const CabinCapacity_T DEFAULT_CABIN_CAPACITY
- const CommittedSpace_T DEFAULT_COMMITTED_SPACE
- const BlockSpace_T DEFAULT_BLOCK_SPACE
- const Availability_T DEFAULT_NULL_AVAILABILITY
- const Availability_T DEFAULT_AVAILABILITY
- const CensorshipFlag_T DEFAULT_CLASS_CENSORSHIPFLAG
- const CensorshipFlagList_T DEFAULT_CLASS_CENSORSHIPFLAG_LIST
- const BookingLimit_T DEFAULT_CLASS_BOOKING_LIMIT
- const AuthorizationLevel_T DEFAULT_CLASS_AUTHORIZATION_LEVEL
- const AuthorizationLevel_T DEFAULT_CLASS_MAX_AUTHORIZATION_LEVEL
- const AuthorizationLevel_T DEFAULT_CLASS_MIN_AUTHORIZATION_LEVEL
- const OverbookingRate_T DEFAULT_CLASS_OVERBOOKING_RATE
- const Fare_T DEFAULT_FARE_VALUE
- const Revenue_T DEFAULT_REVENUE_VALUE
- const PriceCurrency_T DEFAULT_CURRENCY
- const Percentage_T DEFAULT_LOAD_FACTOR_VALUE
- const DayDuration_T DEFAULT_DAY_DURATION
- const double DEFAULT_EPSILON_VALUE
- const AirportCode_T AIRPORT_LHR
- const AirportCode_T AIRPORT_SYD
- const CityCode_T POS_LHR
- const DayDuration_T NO_ADVANCE_PURCHASE
- const SaturdayStay_T SATURDAY_STAY
- const SaturdayStay_T NO_SATURDAY_STAY
- const ChangeFees_T CHANGE_FEES
- const ChangeFees_T NO_CHANGE_FEES
- const NonRefundable_T NON_REFUNDABLE
- const NonRefundable_T NO_NON_REFUNDABLE
- const DayDuration_T NO_STAY_DURATION
- const CabinCode_T CABIN_Y
- const AirlineCode_T AIRLINE_CODE_BA
- const ClassCode_T CLASS_CODE_Y
- const ClassCode_T CLASS_CODE_Q
- const AirportCode_T AIRPORT_SIN
- const AirportCode_T AIRPORT_BKK
- const CityCode_T POS_SIN
- const CabinCode_T CABIN_ECO
- const FrequentFlyer_T FREQUENT_FLYER_MEMBER
- const Count_T DEFAULT_PROGRESS_STATUS
- const Date_T DEFAULT_EVENT_OLDEST_DATE
- const DateTime_T DEFAULT_EVENT_OLDEST_DATETIME
- const Percentage_T MAXIMUM_PROGRESS_STATUS
- const std::string DEFAULT_BOM_ROOT_KEY
- const double DEFAULT_EPSILON_VALUE
- const CabinCapacity_T DEFAULT_CABIN_CAPACITY
- const NbOfFlightDates_T DEFAULT_NB_OF_FLIGHTDATES

- const NbOfBookings_T DEFAULT_CLASS_NB_OF_BOOKINGS
- const Distance_T DEFAULT_DISTANCE_VALUE
- const unsigned int DEFAULT_FLIGHT_SPEED
- const Fare_T DEFAULT_FARE_VALUE
- const PriceCurrency_T DEFAULT_CURRENCY
- const Revenue_T DEFAULT_REVENUE_VALUE
- const BookingRatio_T DEFAULT_OND_BOOKING_RATE
- const Count_T SECONDS_IN_ONE_DAY
- const Count_T MILLISECONDS_IN_ONE_SECOND
- const Date_T DEFAULT_DATE
- const DateTime_T DEFAULT_DATETIME
- const Duration_T DEFAULT_EPSILON_DURATION
- const RandomSeed_T DEFAULT_RANDOM_SEED
- const Duration_T NULL_BOOST_TIME_DURATION
- const Duration_T DEFAULT_NULL_DURATION
- const Fare_T DEFAULT_CLASS_FARE_VALUE
- const NbOfAirlines_T DEFAULT_NBOFAIRLINES
- const unsigned int DEFAULT_NB_OF_DAYS_IN_A_YEAR
- const NbOfBookings_T DEFAULT_CLASS_NB_OF_BOOKINGS
- const ChannelLabel_T DEFAULT_CHANNEL
- const OnDStringList_T DEFAULT_OND_STRING_LIST
- const unsigned int DEFAULT_NUMBER_OF_SUBDIVISIONS
- const AirlineCode_T DEFAULT_AIRLINE_CODE
- const AirlineCode_T DEFAULT_NULL_AIRLINE_CODE
- const AirlineCodeList_T DEFAULT_AIRLINE_CODE_LIST
- const FlightNumber_T DEFAULT_FLIGHT_NUMBER
- const FlightNumber_T DEFAULT_FLIGHT_NUMBER_FF
- const TableID_T DEFAULT_TABLE_ID
- const Date_T DEFAULT_DEPARTURE_DATE
- const AirportCode_T DEFAULT_AIRPORT_CODE
- const AirportCode_T DEFAULT_NULL_AIRPORT_CODE
- const AirportCode_T DEFAULT_ORIGIN
- const AirportCode_T DEFAULT_DESTINATION
- const CabinCode_T DEFAULT_CABIN_CODE
- const FamilyCode_T DEFAULT_FARE_FAMILY_CODE
- const FamilyCode_T DEFAULT_NULL_FARE_FAMILY_CODE
- const PolicyCode_T DEFAULT_POLICY_CODE
- const NestingStructureCode_T DEFAULT_NESTING_STRUCTURE_CODE
- const NestingStructureCode_T DISPLAY_NESTING_STRUCTURE_CODE
- const NestingStructureCode_T YIELD_BASED_NESTING_STRUCTURE_CODE
- const NestingNodeCode_T DEFAULT_NESTING_NODE_CODE
- const ClassCode_T DEFAULT_CLASS_CODE
- const ClassCode_T DEFAULT_NULL_CLASS_CODE
- const ClassList_StringList_T DEFAULT_CLASS_CODE_LIST
- const BidPrice_T DEFAULT_BID_PRICE
- const BidPriceVector_T DEFAULT_BID_PRICE_VECTOR
- const unsigned short MAXIMAL_NUMBER_OF_LEGS_IN_FLIGHT
- const unsigned short MAXIMAL_NUMBER_OF_SEGMENTS_IN_OND
- const Availability_T MAXIMAL_AVAILABILITY
- const SeatIndex_T DEFAULT_SEAT_INDEX

- const NbOfSeats_T DEFAULT_NULL_BOOKING_NUMBER
- const CapacityAdjustment_T DEFAULT_NULL_CAPACITY_ADJUSTMENT
- const UPR_T DEFAULT_NULL_UPR
- const std::string DEFAULT_FARE_FAMILY_VALUE_TYPE
- const std::string DEFAULT_SEGMENT_CABIN_VALUE_TYPE
- const int DEFAULT_MAX_DTD
- const DCPList_T DEFAULT_DCP_LIST
- const DTDFratMap_T DEFAULT_DTD_FRAT5COEF_MAP
- const DTDProbMap_T DEFAULT_DTD_PROB_MAP
- const ForecastingMethod DEFAULT_FORECASTING_METHOD
- const UnconstrainingMethod DEFAULT_UNCONSTRAINING_METHOD
- const PreOptimisationMethod DEFAULT_PREOPTIMISATION_METHOD
- const OptimisationMethod DEFAULT_OPTIMISATION_METHOD
- const PartnershipTechnique DEFAULT_PARTNERSHIP_TECHNIQUE
- const DatePeriod_T BOOST_DEFAULT_DATE_PERIOD
- const std::string DOW_STR [ ]
- const DOW_String_T DEFAULT_DOW_STRING
- const DateOffset_T DEFAULT_DATE_OFFSET
- const DayDuration_T DEFAULT_DAY_DURATION
- const PartySize_T DEFAULT_PARTY_SIZE
- const DayDuration_T DEFAULT_STAY_DURATION
- const WTP_T DEFAULT_WTP
- const CityCode_T DEFAULT_POS
- const Date_T DEFAULT_PREFERRED_DEPARTURE_DATE
- const Duration_T DEFAULT_PREFERRED_DEPARTURE_TIME
- const DateOffset_T DEFAULT_ADVANCE_PURCHASE
- const Date_T DEFAULT_REQUEST_DATE
- const Duration_T DEFAULT_REQUEST_TIME
- const DateTime_T DEFAULT_REQUEST_DATE_TIME
- const CabinCode_T DEFAULT_PREFERRED_CABIN
- const ChannelLabel_T DEFAULT_CHANNEL
- const ChannelLabel_T CHANNEL_DN
- const ChannelLabel_T CHANNEL_IN
- const TripType_T TRIP_TYPE_ONE_WAY
- const TripType_T TRIP_TYPE_ROUND_TRIP
- const TripType_T TRIP_TYPE_INBOUND
- const TripType_T TRIP_TYPE_OUTBOUND
- const FrequentFlyer_T DEFAULT_FF_TIER
- const PriceValue_T DEFAULT_VALUE_OF_TIME
- const IntDuration_T HOUR_CONVERTED_IN_SECONDS
- const FRAT5Curve_T FRAT5_CURVE_A
- const FRAT5Curve_T FRAT5_CURVE_B
- const FRAT5Curve_T FRAT5_CURVE_C
- const FRAT5Curve_T FRAT5_CURVE_D
- const FFDisutilityCurve_T FF_DISUTILITY_CURVE_A
- const FFDisutilityCurve_T FF_DISUTILITY_CURVE_B
- const FFDisutilityCurve_T FF_DISUTILITY_CURVE_C
- const FFDisutilityCurve_T FF_DISUTILITY_CURVE_D
- const FFDisutilityCurve_T FF_DISUTILITY_CURVE_E
- const FFDisutilityCurve_T FF_DISUTILITY_CURVE_F

- const Distance_T DEFAULT_DISTANCE_VALUE
- const Duration_T DEFAULT_MINIMAL_CONNECTION_TIME
- const Duration_T DEFAULT_MAXIMAL_CONNECTION_TIME
- const Duration_T NULL_BOOST_TIME_DURATION
- const FlightPathCode_T DEFAULT_FLIGHTPATH_CODE
- const Availability_T DEFAULT_CLASS_AVAILABILITY
- const AvailabilityStatus_T DEFAULT_AVAILABILITY_STATUS
- const unsigned short DEFAULT_NUMBER_OF_REQUIRED_SEATS
- const MatchingIndicator_T DEFAULT_MATCHING_INDICATOR
- const Revenue_T DEFAULT_REVENUE_VALUE
- const AirlineCode_T DEFAULT_DICO_STUDIED_AIRLINE
- const Date_T DEFAULT_DICO_STUDIED_DATE
- const Yield_T DEFAULT_YIELD_VALUE
- const Yield_T DEFAULT_YIELD_MAX_VALUE

7.5.1 Detailed Description

Handle on the StdAir library context.

Author:

Anh Quan Nguyen <quannaus@users.sourceforge.net>

Date:

20/01/2010 StdAir aims at providing a clean API, and the corresponding C++ implementation, for the basis of Airline IT Business Object Model (BOM), that is, to be used by several other Open Source projects, such as RMOL and OpenTREP.

Install the StdAir library for Airline IT Standard C++ fundaments.

7.5.2 Typedef Documentation

7.5.2.1 typedef date_time_element<0, 23> stdair::hour_t

Type definitions for the date and time elements.

Definition at line 61 of file BasParserHelperTypes.hpp.

7.5.2.2 typedef date_time_element<0, 59> stdair::minute_t

Definition at line 62 of file BasParserHelperTypes.hpp.

7.5.2.3 typedef date_time_element<0, 59> stdair::second_t

Definition at line 63 of file BasParserHelperTypes.hpp.

7.5.2.4 typedef date_time_element<1900, 2100> stdair::year_t

Definition at line 64 of file BasParserHelperTypes.hpp.

7.5.2.5 typedef date_time_element<1, 12> stdair::month_t

Definition at line 65 of file BasParserHelperTypes.hpp.

7.5.2.6 typedef date_time_element<1, 31> stdair::day_t

Definition at line 66 of file BasParserHelperTypes.hpp.

7.5.2.7 typedef std::istreambuf_iterator<char> stdair::base_iterator_t

Definition at line 26 of file BasParserTypes.hpp.

7.5.2.8 typedef boost::spirit::multi_pass<base_iterator_t> stdair::iterator_t

Definition at line 27 of file BasParserTypes.hpp.

7.5.2.9 typedef boost::spirit::qi::int_parser<unsigned int, 10, 1, 1> stdair::int1_p_t

1-digit-integer parser

Definition at line 35 of file BasParserTypes.hpp.

7.5.2.10 typedef boost::spirit::qi::uint_parser<int, 10, 2, 2> stdair::uint2_p_t

2-digit-integer parser

Definition at line 38 of file BasParserTypes.hpp.

7.5.2.11 typedef boost::spirit::qi::uint_parser<int, 10, 4, 4> stdair::uint4_p_t

4-digit-integer parser

Definition at line 41 of file BasParserTypes.hpp.

7.5.2.12 typedef boost::spirit::qi::uint_parser<int, 10, 1, 4> stdair::uint1_4_p_t

Up-to-4-digit-integer parser

Definition at line 44 of file BasParserTypes.hpp.

7.5.2.13 typedef boost::spirit::qi::uint_parser<hour_t, 10, 2, 2> stdair::hour_p_t

Date & time element parsers.

Definition at line 47 of file BasParserTypes.hpp.

7.5.2.14 typedef boost::spirit::qi::uint_parser<minute_t, 10, 2, 2> stdair::minute_p_t

Definition at line 48 of file BasParserTypes.hpp.

7.5.2.15 typedef boost::spirit::qi::uint_parser<second_t, 10, 2, 2> stdair::second_p_t

Definition at line 49 of file BasParserTypes.hpp.

7.5.2.16 typedef boost::spirit::qi::uint_parser<year_t, 10, 4, 4> stdair::year_p_t

Definition at line 50 of file BasParserTypes.hpp.

7.5.2.17 typedef boost::spirit::qi::uint_parser<month_t, 10, 2, 2> stdair::month_p_t

Definition at line 51 of file BasParserTypes.hpp.

7.5.2.18 typedef boost::spirit::qi::uint_parser<day_t, 10, 2, 2> stdair::day_p_t

Definition at line 52 of file BasParserTypes.hpp.

7.5.2.19 typedef unsigned short stdair::DictionaryKey_T

Dictionary key.

Definition at line 17 of file DictionaryManager.hpp.

7.5.2.20 typedef std::list<AirlineClassList*> stdair::AirlineClassListList_T

Define the segment-features list.

Definition at line 17 of file AirlineClassListTypes.hpp.

7.5.2.21 typedef std::map<const MapKey_T, AirlineClassList*> stdair::AirlineClassListMap_T

Define the segment-features map.

Definition at line 23 of file AirlineClassListTypes.hpp.

7.5.2.22 typedef std::pair<MapKey_T, AirlineClassList*> stdair::AirlineClassListWithKey_T

Define the list of pair<MapKey_T, AirlineCodeList>.

Definition at line 26 of file AirlineClassListTypes.hpp.

7.5.2.23 typedef std::list<AirlineClassListWithKey_T> stdair::AirlineClassListDetailedList_T

Definition at line 27 of file AirlineClassListTypes.hpp.

7.5.2.24 typedef std::list<AirlineFeature*> stdair::AirlineFeatureList_T

Define the airline feature list.

Definition at line 17 of file AirlineFeatureTypes.hpp.

7.5.2.25 typedef std::map<const MapKey_T, AirlineFeature*> stdair::AirlineFeatureMap_T

Define the airline feature map.

Definition at line 23 of file AirlineFeatureTypes.hpp.

7.5.2.26 typedef std::list<AirportPair*> stdair::AirportPairList_T

Define the airport-pair list.

Definition at line 17 of file AirportPairTypes.hpp.

7.5.2.27 typedef std::map<const MapKey_T, AirportPair*> stdair::AirportPairMap_T

Define the airport-pair map.

Definition at line 23 of file AirportPairTypes.hpp.

7.5.2.28 typedef std::pair<MapKey_T, AirportPair*> stdair::AirportPairWithKey_T

Define the list of pair<MapKey_T, AirportPair>.

Definition at line 26 of file AirportPairTypes.hpp.

7.5.2.29 typedef std::list<AirportPairWithKey_T> stdair::AirportPairDetailedList_T

Definition at line 27 of file AirportPairTypes.hpp.

7.5.2.30 typedef std::map<const std::type_info*, BomAbstract*> stdair::HolderMap_T

Definition at line 63 of file BomAbstract.hpp.

7.5.2.31 typedef struct BomID< BookingClass > stdair::BookingClassID_T

Define the booking class ID.

Definition at line 21 of file BomIDTypes.hpp.

7.5.2.32 typedef std::list<BookingClassID_T> stdair::BookingClassIDList_T

Define the list of booking class ID's.

Definition at line 24 of file BomIDTypes.hpp.

7.5.2.33 typedef boost::tokenizer<boost::char_separator<char> > stdair::Tokeniser_T

Boost Tokeniser.

Definition at line 28 of file BomKeyManager.cpp.

7.5.2.34 typedef std::list<BookingClass*> stdair::BookingClassList_T

Define the booking class list.

Definition at line 17 of file BookingClassTypes.hpp.

7.5.2.35 typedef std::map<const MapKey_T, BookingClass*> stdair::BookingClassMap_T

Define the booking class map.

Definition at line 23 of file BookingClassTypes.hpp.

7.5.2.36 typedef boost::shared_ptr<BookingRequestStruct> stdair::BookingRequestPtr_T

Define the smart pointer to a booking request.

Definition at line 14 of file BookingRequestTypes.hpp.

7.5.2.37 typedef std::string stdair::DemandGeneratorKey_T

Define the hash key for the demand generator.

Definition at line 21 of file BookingRequestTypes.hpp.

7.5.2.38 typedef boost::shared_ptr<BreakPointStruct> stdair::BreakPointPtr_T

Define the smart pointer to a Break Point event .

Definition at line 16 of file BreakPointTypes.hpp.

7.5.2.39 typedef std::list<BreakPointStruct> stdair::BreakPointList_T

Define the list of Break Points.

Definition at line 23 of file BreakPointTypes.hpp.

7.5.2.40 typedef std::list<Bucket*> stdair::BucketList_T

Define the bucket list.

Definition at line 17 of file BucketTypes.hpp.

7.5.2.41 typedef std::map<const MapKey_T, Bucket*> stdair::BucketMap_T

Define the bucket map.

Definition at line 23 of file BucketTypes.hpp.

7.5.2.42 typedef boost::shared_ptr<CancellationStruct> stdair::CancellationPtr_T

Define the smart pointer to a cancellation .

Definition at line 14 of file CancellationTypes.hpp.

7.5.2.43 typedef boost::shared_ptr<ConfigHolderStruct> stdair::ConfigHolderPtr_T

Define the smart pointer to a Config Holder structure.

Definition at line 16 of file ConfigHolderTypes.hpp.

7.5.2.44 typedef std::list<DatePeriod*> stdair::DatePeriodList_T

Define the date-period list.

Definition at line 17 of file DatePeriodTypes.hpp.

7.5.2.45 typedef std::map<const MapKey_T, DatePeriod*> stdair::DatePeriodMap_T

Define the date-period map.

Definition at line 23 of file DatePeriodTypes.hpp.

7.5.2.46 typedef std::pair<MapKey_T, DatePeriod*> stdair::DatePeriodWithKey_T

Define the list of pair<MapKey_T, DatePeriod>.

Definition at line 26 of file DatePeriodTypes.hpp.

7.5.2.47 typedef std::list<DatePeriodWithKey_T> stdair::DatePeriodDetailedList_T

Definition at line 27 of file DatePeriodTypes.hpp.

7.5.2.48 typedef std::pair<const LongDuration_T, EventStruct> stdair::EventListElement_T

Define an element of a event list.

Definition at line 22 of file EventTypes.hpp.

7.5.2.49 typedef std::map<const LongDuration_T, EventStruct> stdair::EventList_T

Define a list of events.

Definition at line 32 of file EventTypes.hpp.

7.5.2.50 typedef std::list<FareFamily*> stdair::FareFamilyList_T

Define the fare family list.

Definition at line 17 of file FareFamilyTypes.hpp.

7.5.2.51 typedef std::map<const MapKey_T, FareFamily*> stdair::FareFamilyMap_T

Define the fare family map.

Definition at line 23 of file FareFamilyTypes.hpp.

7.5.2.52 typedef std::list<FareFeatures*> stdair::FareFeaturesList_T

Define the date-period list.

Definition at line 17 of file FareFeaturesTypes.hpp.

7.5.2.53 typedef std::map<const MapKey_T, FareFeatures*> stdair::FareFeaturesMap_T

Define the date-period map.

Definition at line 23 of file FareFeaturesTypes.hpp.

7.5.2.54 typedef std::pair<MapKey_T, FareFeatures*> stdair::FareFeaturesWithKey_T

Define the list of pair<MapKey_T, FareFeatures>.

Definition at line 26 of file FareFeaturesTypes.hpp.

7.5.2.55 typedef std::list<FareFeaturesWithKey_T> stdair::FareFeaturesDetailedList_T

Definition at line 27 of file FareFeaturesTypes.hpp.

7.5.2.56 typedef std::list<FareOptionStruct> stdair::FareOptionList_T

Define the booking class list.

Definition at line 18 of file FareOptionTypes.hpp.

7.5.2.57 typedef std::map<const std::string, FFDisutilityCurve_T> stdair::FFDisutilityCurveHolder_T

Definition at line 16 of file FFDisutilityCurveHolderStruct.hpp.

7.5.2.58 typedef std::list<FlightDate*> stdair::FlightDateList_T

Define the flight-date list.

Definition at line 17 of file FlightDateTypes.hpp.

7.5.2.59 typedef std::map<const MapKey_T, FlightDate*> stdair::FlightDateMap_T

Define the flight-date map.

Definition at line 24 of file FlightDateTypes.hpp.

7.5.2.60 typedef std::list<FlightPeriod*> stdair::FlightPeriodList_T

Define the flight-period list.

Definition at line 17 of file FlightPeriodTypes.hpp.

7.5.2.61 typedef std::map<const MapKey_T, FlightPeriod*> stdair::FlightPeriodMap_T

Define the flight-period map.

Definition at line 23 of file FlightPeriodTypes.hpp.

7.5.2.62 typedef std::map<const std::string, FRAT5Curve_T> stdair::FRAT5CurveHolder_T

Definition at line 16 of file FRAT5CurveHolderStruct.hpp.

7.5.2.63 typedef std::list<Inventory*> stdair::InventoryList_T

Define the Inventory list.

Definition at line 17 of file InventoryTypes.hpp.

7.5.2.64 typedef std::map<const MapKey_T, Inventory*> stdair::InventoryMap_T

Define the Inventory map.

Definition at line 23 of file InventoryTypes.hpp.

7.5.2.65 typedef std::string stdair::MapKey_T

Key of a STL map.

Definition at line 15 of file key_types.hpp.

7.5.2.66 typedef std::list<std::string> stdair::KeyList_T

List of keys.

Definition at line 18 of file key_types.hpp.

7.5.2.67 typedef std::list<LegCabin*> stdair::LegCabinList_T

Define the leg-cabin list.

Definition at line 17 of file LegCabinTypes.hpp.

7.5.2.68 typedef std::map<const MapKey_T, LegCabin*> stdair::LegCabinMap_T

Define the leg-cabin map.

Definition at line 23 of file LegCabinTypes.hpp.

7.5.2.69 typedef std::list<LegDate*> stdair::LegDateList_T

Define the leg-date list.

Definition at line 17 of file LegDateTypes.hpp.

7.5.2.70 typedef std::map<const MapKey_T, LegDate*> stdair::LegDateMap_T

Define the leg-date map.

Definition at line 23 of file LegDateTypes.hpp.

7.5.2.71 typedef std::list<NestingNode*> stdair::NestingNodeList_T

Define the fare family list.

Definition at line 17 of file NestingNodeTypes.hpp.

7.5.2.72 typedef std::map<const MapKey_T, NestingNode*> stdair::NestingNodeMap_T

Define the fare family map.

Definition at line 23 of file NestingNodeTypes.hpp.

7.5.2.73 typedef std::list<OnDDate*> stdair::OnDDateList_T

Define the O&D date list.

Definition at line 19 of file OnDDateTypes.hpp.

7.5.2.74 typedef std::map<const MapKey_T, OnDDate*> stdair::OnDDateMap_T

Define the OnD date map.

Definition at line 25 of file OnDDateTypes.hpp.

7.5.2.75 typedef std::pair<std::string, YieldDemandPair_T> stdair::StringDemandStructPair_T

Define the yield mean and standard deviation for a certain cabin/class path. This map is mandatory when using the default BOM tree. This map can be empty if yields are charged otherwise (input file, ...)

Definition at line 32 of file OnDDateTypes.hpp.

7.5.2.76 typedef std::map<std::string, YieldDemandPair_T> stdair::StringDemandStructMap_T

Definition at line 33 of file OnDDateTypes.hpp.

7.5.2.77 typedef std::map<std::string, CabinClassPairList_T> stdair::StringCabinClassPairList-Map_T

Define the string matching a (cabin,class) path. (i.e, the string is "Y:M;" for a one leg O&D with the cabin Y and the class M; the string is "Y:M;Y:Y;" for a two legs O&D with the cabin Y and the class M for the first leg, and the cabin Y and the class Y for the second leg).

Definition at line 41 of file OnDDateTypes.hpp.

7.5.2.78 typedef std::pair<std::string, CabinClassPairList_T> stdair::StringCabinClassPair_T

Definition at line 42 of file OnDDateTypes.hpp.

7.5.2.79 typedef std::map<CabinCode_T, WTPDemandPair_T> stdair::CabinForecastMap_T

Define the WTP mean and standard deviation for a certain cabin code. This information is needed to forecast O&D demand per cabin.

Definition at line 48 of file OnDDateTypes.hpp.

7.5.2.80 typedef std::pair<CabinCode_T, WTPDemandPair_T> stdair::CabinForecastPair_T

Definition at line 49 of file OnDDateTypes.hpp.

7.5.2.81 typedef boost:: shared_ptr<OptimisationNotificationStruct> stdair::Optimisation-NotificationPtr_T

Define the smart pointer to a optimisation notification.

Definition at line 14 of file OptimisationNotificationTypes.hpp.

7.5.2.82 typedef boost::tokenizer<boost::char_separator<char> > stdair::Tokeniser_T

Boost Tokeniser.

Definition at line 28 of file ParsedKey.cpp.

7.5.2.83 typedef std::list<Policy*> stdair::PolicyList_T

Define the fare family list.

Definition at line 17 of file PolicyTypes.hpp.

7.5.2.84 typedef std::map<const MapKey_T, Policy*> stdair::PolicyMap_T

Define the fare family map.

Definition at line 23 of file PolicyTypes.hpp.

7.5.2.85 typedef std::list<PosChannel*> stdair::PosChannelList_T

Define the fare-point_of_sale list.

Definition at line 17 of file PosChannelTypes.hpp.

7.5.2.86 typedef std::map<const MapKey_T, PosChannel*> stdair::PosChannelMap_T

Define the fare-point_of_sale map.

Definition at line 23 of file PosChannelTypes.hpp.

7.5.2.87 typedef std::pair<MapKey_T, PosChannel*> stdair::PosChannelWithKey_T

Define the list of pair<MapKey_T, PosChannel>.

Definition at line 26 of file PosChannelTypes.hpp.

7.5.2.88 typedef std::list<PosChannelWithKey_T> stdair::PosChannelDetailedList_T

Definition at line 27 of file PosChannelTypes.hpp.

7.5.2.89 typedef boost::shared_ptr<RMEventStruct> stdair::RMEventPtr_T

Define the smart pointer to a RM event .

Definition at line 16 of file RMEventTypes.hpp.

7.5.2.90 typedef std::list<RMEventStruct> stdair::RMEventList_T

Define the list of RM events.

Definition at line 23 of file RMEventTypes.hpp.

7.5.2.91 typedef std::list<SegmentCabin*> stdair::SegmentCabinList_T

Define the segment-cabin list.

Definition at line 17 of file SegmentCabinTypes.hpp.

7.5.2.92 typedef std::map<const MapKey_T, SegmentCabin*> stdair::SegmentCabinMap_T

Define the segment-cabin map.

Definition at line 23 of file SegmentCabinTypes.hpp.

7.5.2.93 typedef std::list<std::string> stdair::RoutingLegKeyList_T

Definition at line 27 of file SegmentDate.hpp.

7.5.2.94 typedef std::list<SegmentDate*> stdair::SegmentDateList_T

Define the segment-date list.

Definition at line 17 of file SegmentDateTypes.hpp.

7.5.2.95 typedef std::map<const MapKey_T, SegmentDate*> stdair::SegmentDateMap_T

Define the segment-date map.

Definition at line 23 of file SegmentDateTypes.hpp.

7.5.2.96 typedef std::list<SegmentPeriod*> stdair::SegmentPeriodList_T

Define the segment-period list.

Definition at line 17 of file SegmentPeriodTypes.hpp.

7.5.2.97 typedef std::map<const MapKey_T, SegmentPeriod*> stdair::SegmentPeriodMap_T

Define the segment-period map.

Definition at line 23 of file SegmentPeriodTypes.hpp.

7.5.2.98 typedef std::pair<MapKey_T, SegmentPeriod*> stdair::SegmentPeriodWithKey_T

Define the list of pair<MapKey_T, SegmentPeriod>.

Definition at line 26 of file SegmentPeriodTypes.hpp.

7.5.2.99 typedef std::list<SegmentPeriodWithKey_T> stdair::SegmentPeriodDetailedList_T

Definition at line 27 of file SegmentPeriodTypes.hpp.

7.5.2.100 typedef std::list<SegmentSnapshotTable*> stdair::SegmentSnapshotTableList_T

Define the guillotine-block list.

Definition at line 20 of file SegmentSnapshotTableTypes.hpp.

7.5.2.101 typedef std::map<const MapKey_T, SegmentSnapshotTable*> stdair::SegmentSnapshotTableMap_T

Define the guillotine-block map.

Definition at line 27 of file SegmentSnapshotTableTypes.hpp.

7.5.2.102 typedef std::map<const SegmentCabin*, SegmentDataID_T> stdair::SegmentCabinIndexMap_T

Define the map between the segment-cabins and the segment data ID.

Definition at line 30 of file SegmentSnapshotTableTypes.hpp.

7.5.2.103 typedef std::map<const MapKey_T, ClassIndex_T> stdair::ClassIndexMap_T

Define the map between the class and their index.

Definition at line 33 of file SegmentSnapshotTableTypes.hpp.

7.5.2.104 typedef std::list<SimpleNestingStructure*> stdair::SimpleNestingStructureList_T

Define the fare family list.

Definition at line 17 of file SimpleNestingStructureTypes.hpp.

7.5.2.105 typedef std::map<const MapKey_T, SimpleNestingStructure*> stdair::SimpleNestingStructureMap_T

Define the fare family map.

Definition at line 23 of file SimpleNestingStructureTypes.hpp.

7.5.2.106 typedef boost::shared_ptr<SnapshotStruct> stdair::SnapshotPtr_T

Define the smart pointer to a snapshot .

Definition at line 14 of file SnapshotTypes.hpp.

7.5.2.107 typedef std::list<TimePeriod*> stdair::TimePeriodList_T

Define the time-period list.

Definition at line 17 of file TimePeriodTypes.hpp.

7.5.2.108 typedef std::map<const MapKey_T, TimePeriod*> stdair::TimePeriodMap_T

Define the time-period map.

Definition at line 23 of file TimePeriodTypes.hpp.

7.5.2.109 typedef std::pair<MapKey_T, TimePeriod*> stdair::TimePeriodWithKey_T

Define the list of pair<MapKey_T, TimePeriod>.

Definition at line 26 of file TimePeriodTypes.hpp.

7.5.2.110 typedef std::list<TimePeriodWithKey_T> stdair::TimePeriodDetailedList_T

Definition at line 27 of file TimePeriodTypes.hpp.

7.5.2.111 typedef std::list<TravelSolutionStruct> stdair::TravelSolutionList_T

Define the booking class list.

Definition at line 20 of file TravelSolutionTypes.hpp.

7.5.2.112 typedef KeyList_T stdair::SegmentPath_T

Define the segment path key.

Definition at line 26 of file TravelSolutionTypes.hpp.

7.5.2.113 typedef std::list<SegmentPath_T> stdair::SegmentPathList_T

Define the list of segment paths.

Definition at line 29 of file TravelSolutionTypes.hpp.

7.5.2.114 typedef std::map<const ClassCode_T, Availability_T> stdair::ClassAvailabilityMap_T

Define booking class - availability map.

Definition at line 32 of file TravelSolutionTypes.hpp.

7.5.2.115 typedef std::list<ClassAvailabilityMap_T> stdair::ClassAvailabilityMapHolder_T

Define list of booking class - availability maps.

Definition at line 35 of file TravelSolutionTypes.hpp.

7.5.2.116 typedef std::map<const ClassCode_T, BookingClassID_T> stdair::ClassObjectIDMap_T

Define booking class - object ID map.

Definition at line 38 of file TravelSolutionTypes.hpp.

7.5.2.117 typedef std::list<ClassObjectIDMap_T> stdair::ClassObjectIDMapHolder_T

Define list of boking class - object ID maps.

Definition at line 41 of file TravelSolutionTypes.hpp.

7.5.2.118 typedef std::map<const ClassCode_T, YieldValue_T> stdair::ClassYieldMap_T

Define booking class - yield map.

Definition at line 44 of file TravelSolutionTypes.hpp.

7.5.2.119 typedef std::list<ClassYieldMap_T> stdair::ClassYieldMapHolder_T

Define list of booking class - yield maps.

Definition at line 47 of file TravelSolutionTypes.hpp.

7.5.2.120 typedef std::list<BidPriceVector_T> stdair::BidPriceVectorHolder_T

Define list of bid price vectors.

Definition at line 50 of file TravelSolutionTypes.hpp.

7.5.2.121 typedef std::map<const ClassCode_T, const BidPriceVector_T*> stdair::ClassBpvMap_T

Define booking class - bid price reference map.

Definition at line 53 of file TravelSolutionTypes.hpp.

7.5.2.122 typedef std::list<ClassBpvMap_T> stdair::ClassBpvMapHolder_T

Define list of booking class - bid price reference maps.

Definition at line 56 of file TravelSolutionTypes.hpp.

7.5.2.123 typedef std::list<VirtualClassStruct> stdair::VirtualClassList_T

Define the booking class list.

Definition at line 17 of file VirtualClassTypes.hpp.

7.5.2.124 typedef std::map<const YieldLevel_T, VirtualClassStruct> stdair::VirtualClassMap_T

Define the booking class map.

Definition at line 23 of file VirtualClassTypes.hpp.

7.5.2.125 typedef std::list<YieldFeatures*> stdair::YieldFeaturesList_T

Define the date-period list.

Definition at line 17 of file YieldFeaturesTypes.hpp.

7.5.2.126 typedef std::map<const MapKey_T, YieldFeatures*> stdair::YieldFeaturesMap_T

Define the date-period map.

Definition at line 23 of file YieldFeaturesTypes.hpp.

7.5.2.127 typedef std::pair<MapKey_T, YieldFeatures*> stdair::YieldFeaturesWithKey_T

Define the list of pair<MapKey_T, YieldFeatures>.

Definition at line 26 of file YieldFeaturesTypes.hpp.

7.5.2.128 typedef std::list<YieldFeaturesWithKey_T> stdair::YieldFeaturesDetailedList_T

Definition at line 27 of file YieldFeaturesTypes.hpp.

7.5.2.129 typedef std::list<YieldStore*> stdair::YieldStoreList_T

Define the Inventory list.

Definition at line 17 of file YieldStoreTypes.hpp.

7.5.2.130 typedef std::map<const MapKey_T, YieldStore*> stdair::YieldStoreMap_T

Define the Inventory map.

Definition at line 23 of file YieldStoreTypes.hpp.

7.5.2.131 typedef std::string stdair::LocationCode_T

Location code (3-letter-code, e.g., LON).

Definition at line 16 of file stdair_basic_types.hpp.

7.5.2.132 typedef unsigned long int stdair::Distance_T

Define a distance (kilometers).

Definition at line 19 of file stdair_basic_types.hpp.

7.5.2.133 typedef LocationCode_T stdair::AirportCode_T

Define the Airport Code type (3-letter-code, e.g., LHR).

Definition at line 22 of file stdair_basic_types.hpp.

7.5.2.134 typedef LocationCode_T stdair::CityCode_T

City code

Definition at line 25 of file stdair_basic_types.hpp.

7.5.2.135 typedef std::string stdair::KeyDescription_T

Define the key description.

Definition at line 28 of file stdair_basic_types.hpp.

7.5.2.136 typedef std::string stdair::AirlineCode_T

Define the Airline Code type (2-letter-code, e.g., BA).

Definition at line 31 of file stdair_basic_types.hpp.

7.5.2.137 typedef unsigned short stdair::FlightNumber_T

Define the type for flight numbers.

Definition at line 34 of file stdair_basic_types.hpp.

7.5.2.138 typedef unsigned short stdair::TableID_T

Define the type for data table numbers.

Definition at line 37 of file stdair_basic_types.hpp.

7.5.2.139 typedef std::string stdair::CabinCode_T

Define the cabin code (class of service, e.g., first, business, economy).

Definition at line 41 of file stdair_basic_types.hpp.

7.5.2.140 typedef std::string stdair::FamilyCode_T

Define the code of the fare family (e.g., 1, 2, 3, etc.).

Definition at line 44 of file stdair_basic_types.hpp.

7.5.2.141 typedef std::string stdair::PolicyCode_T

Define the code of the policy (e.g., 1, 2, 3, etc.).

Definition at line 47 of file stdair_basic_types.hpp.

7.5.2.142 typedef std::string stdair::NestingStructureCode_T

Define the code of the nesting structure (e.g., "default").

Definition at line 50 of file stdair_basic_types.hpp.

7.5.2.143 typedef std::string stdair::NestingNodeCode_T

Define the code of the nesting node (e.g., 1, 2, 3, etc).

Definition at line 53 of file stdair_basic_types.hpp.

7.5.2.144 typedef std::string stdair::ClassCode_T

Define the booking class code (product segment class, e.g., H, B, K, etc.).

Definition at line 57 of file stdair_basic_types.hpp.

7.5.2.145 typedef unsigned long stdair::Identity_T

Define a identity number.

Definition at line 60 of file stdair_basic_types.hpp.

7.5.2.146 typedef std::string stdair::TripType_T

Type of trip type (RO=outbound of round-trip, RI=inbound of round-trip, OW=one way).

Definition at line 64 of file stdair_basic_types.hpp.

7.5.2.147 typedef double stdair::MonetaryValue_T

Monetary value

Definition at line 67 of file stdair_basic_types.hpp.

7.5.2.148 typedef double stdair::RealNumber_T

Real number

Definition at line 70 of file stdair_basic_types.hpp.

7.5.2.149 typedef double stdair::Percentage_T

Define a percentage value (between 0 and 100%).

Definition at line 73 of file stdair_basic_types.hpp.

7.5.2.150 typedef double stdair::PriceValue_T

Define a price value (e.g., 1000.0 Euros).

Definition at line 76 of file stdair_basic_types.hpp.

7.5.2.151 typedef double stdair::YieldValue_T

Define a yield value (e.g., 1000.0 Euros).

Definition at line 79 of file stdair_basic_types.hpp.

7.5.2.152 typedef std::string stdair::PriceCurrency_T

Define a price currency (e.g., EUR for Euros).

Definition at line 82 of file stdair_basic_types.hpp.

7.5.2.153 typedef double stdair::Revenue_T

Define an amount of revenue.

Definition at line 85 of file stdair_basic_types.hpp.

7.5.2.154 typedef double stdair::Multiplier_T

Define the name of a multiplier.

Definition at line 88 of file stdair_basic_types.hpp.

7.5.2.155 typedef double stdair::NbOfSeats_T

Define the number of seats (it can be non integer, because the overbooking can be applied at booking class or PNR level).

Definition at line 92 of file stdair_basic_types.hpp.

7.5.2.156 typedef unsigned int stdair::Count_T

Count

Definition at line 95 of file stdair_basic_types.hpp.

7.5.2.157 typedef short stdair::PartySize_T

Number of passengers (in a group) for a booking.

Definition at line 98 of file stdair_basic_types.hpp.

7.5.2.158 typedef double stdair::NbOfRequests_T

Define a number of requests.

Definition at line 101 of file stdair_basic_types.hpp.

7.5.2.159 typedef NbOfRequests_T stdair::NbOfBookings_T

Define a number of bookings.

Definition at line 104 of file stdair_basic_types.hpp.

7.5.2.160 typedef NbOfRequests_T stdair::NbOfCancellations_T

Define a number of cancellations.

Definition at line 107 of file stdair_basic_types.hpp.

7.5.2.161 typedef unsigned short stdair::NbOfTravelSolutions_T

Define a number of travel solutions (in a travel solution block).

Definition at line 111 of file stdair_basic_types.hpp.

7.5.2.162 typedef std::string stdair::ClassList_String_T

Define the list of class codes as a string.

Definition at line 114 of file stdair_basic_types.hpp.

7.5.2.163 typedef unsigned short stdair::NbOfSegments_T

Define a number of segment-dates (in a path).

Definition at line 117 of file stdair_basic_types.hpp.

7.5.2.164 typedef unsigned short stdair::NbOfAirlines_T

Define a number of airlines (in a path).

Definition at line 120 of file stdair_basic_types.hpp.

7.5.2.165 typedef double stdair::Availability_T

Define an availability.

Definition at line 123 of file stdair_basic_types.hpp.

7.5.2.166 typedef double stdair::Fare_T

Define the price of a travel solution.

Definition at line 126 of file stdair_basic_types.hpp.

7.5.2.167 typedef bool stdair::Flag_T

Define the censorship flag.

Definition at line 129 of file stdair_basic_types.hpp.

7.5.2.168 typedef unsigned int stdair::UnsignedIndex_T

Define the unsigned index type.

Definition at line 132 of file stdair_basic_types.hpp.

7.5.2.169 typedef unsigned int stdair::NbOfClasses_T

Define the number of booking classes.

Definition at line 135 of file stdair_basic_types.hpp.

7.5.2.170 typedef unsigned int stdair::NbOfFareFamilies_T

Define the number of fare families.

Definition at line 138 of file stdair_basic_types.hpp.

7.5.2.171 typedef std::string stdair::Filename_T

File or directory name.

It may contain paths, relative or absolute (e.g., /foo/bar or C:).

Definition at line 144 of file stdair_basic_types.hpp.

7.5.2.172 typedef std::string stdair::FileAddress_T

Define the file address type (e.g. "a_directory/a_filename").

NOTE: That type should be deprecated.

Definition at line 148 of file stdair_basic_types.hpp.

7.5.2.173 typedef float stdair::ProgressPercentage_T

Progress status (usually, a percentage expressed as a floating point number).

Definition at line 152 of file stdair_basic_types.hpp.

7.5.2.174 typedef boost::posix_time::time_duration stdair::Duration_T

Define the type for durations (e.g., elapsed in-flight time).

Definition at line 17 of file stdair_date_time_types.hpp.

7.5.2.175 typedef boost::gregorian::date stdair::Date_T

Define the type for date (e.g., departure date of a flight).

Definition at line 20 of file stdair_date_time_types.hpp.

7.5.2.176 typedef boost::posix_time::time_duration stdair::Time_T

Time

Definition at line 23 of file stdair_date_time_types.hpp.

7.5.2.177 typedef boost::posix_time::ptime stdair::DateTime_T

Define an accurate time (date+time).

Definition at line 26 of file stdair_date_time_types.hpp.

7.5.2.178 typedef boost::gregorian::date_period stdair::DatePeriod_T

Define the Period (e.g., period during which flights depart).

Definition at line 29 of file stdair_date_time_types.hpp.

7.5.2.179 typedef std::string stdair::DOW_String_T

Define the Day-Of-the-Week as a string.

Definition at line 32 of file stdair_date_time_types.hpp.

7.5.2.180 typedef boost::gregorian::date_duration stdair::DateOffset_T

Define the Date Offset (e.g., -1).

Definition at line 35 of file stdair_date_time_types.hpp.

7.5.2.181 typedef int stdair::DayDuration_T

Define a duration in number of days.

Definition at line 38 of file stdair_date_time_types.hpp.

7.5.2.182 typedef bool stdair::SaturdayStay_T

Define the Saturday stay status of a travel.

Definition at line 41 of file stdair_date_time_types.hpp.

7.5.2.183 typedef long int stdair::IntDuration_T

Time duration in (integer) number of seconds

Definition at line 44 of file stdair_date_time_types.hpp.

7.5.2.184 typedef long long int stdair::LongDuration_T

Time duration in (unsigned long long integer) number of milliseconds

Definition at line 47 of file stdair_date_time_types.hpp.

7.5.2.185 typedef float stdair::FloatDuration_T

Duration in (float) number of time units

Definition at line 50 of file stdair_date_time_types.hpp.

7.5.2.186 typedef soci::session stdair::DBSession_T

Database session handler.

Definition at line 20 of file stdair_db.hpp.

7.5.2.187 typedef soci::statement stdair::DBRequestStatement_T

Database request statement handler.

Definition at line 23 of file stdair_db.hpp.

7.5.2.188 typedef std::string stdair::DBConnectionName_T

Define the name of an database connection.

Definition at line 26 of file stdair_db.hpp.

7.5.2.189 typedef bool stdair::ChangeFees_T

Define the availability option allowing the ticket change.

Definition at line 29 of file stdair_demand_types.hpp.

7.5.2.190 typedef bool stdair::NonRefundable_T

Define the refundable availability of a tickets.

Definition at line 32 of file stdair_demand_types.hpp.

7.5.2.191 typedef bool stdair::SaturdayStay_T

Define the saturday stay of a tickets.

Definition at line 35 of file stdair_demand_types.hpp.

7.5.2.192 typedef double stdair::SaturdayStayRatio_T

Define the average ratio (between 0 and 100 percent) of demand with a saturday stay status equal to TRUE.

Definition at line 39 of file stdair_demand_types.hpp.

7.5.2.193 typedef double stdair::ChangeFeesRatio_T

Define the average ratio of demand with change fee availability.

Definition at line 43 of file stdair_demand_types.hpp.

7.5.2.194 typedef double stdair::NonRefundableRatio_T

Define the average ratio of demand with non-refundable availability.

Definition at line 47 of file stdair_demand_types.hpp.

7.5.2.195 typedef double stdair::Disutility_T

Define the disutility of restriction.

Definition at line 50 of file stdair_demand_types.hpp.

7.5.2.196 typedef std::string stdair::PassengerType_T

Define the passenger characteristics, leisure or business for instance (1-letter-code, e.g., L or B).

Definition at line 54 of file stdair_demand_types.hpp.

7.5.2.197 typedef std::string stdair::DistributionPatternId_T

Define the identifier of a distribution pattern (e.g., 1).

Definition at line 57 of file stdair_demand_types.hpp.

7.5.2.198 typedef std::string stdair::CancellationRateCurveId_T

Define the identifier of a cancellation rate curve (e.g., C1).

Definition at line 60 of file stdair_demand_types.hpp.

7.5.2.199 typedef std::string stdair::AirlinePreferenceId_T

Define the identifier of an airline preference set list (e.g., AP1).

Definition at line 63 of file stdair_demand_types.hpp.

7.5.2.200 typedef std::pair<Percentage_T, Percentage_T> stdair::CancellationNoShowRatePair_T

Define a cancellation & and no-show rate pair.

Definition at line 66 of file stdair_demand_types.hpp.

7.5.2.201 typedef std::string stdair::CharacteristicsPatternId_T

Define the identifier of a demand characteristics pattern (e.g. Ch12); for a customer choice model

Definition at line 70 of file stdair_demand_types.hpp.

7.5.2.202 typedef std::string stdair::CharacteristicsIndex_T

Define characteristics component index (e.g. W for WTP)

Definition at line 73 of file stdair_demand_types.hpp.

7.5.2.203 typedef double stdair::WTP_T

Define a Willingness-To-Pay (WTP) (e.g., 1000.0 Euros).

Definition at line 76 of file stdair_demand_types.hpp.

7.5.2.204 typedef boost::tuples::tuple<double, WTP_T> stdair::CharacteristicsWTP_tuple_T

Define the name of a WTP-component of characteristics pattern.

Definition at line 79 of file stdair_demand_types.hpp.

7.5.2.205 typedef std::pair<WTP_T, MeanStdDevPair_T> stdair::WTPDemandPair_T

Define the <WTP, demand> pair type.

Definition at line 82 of file stdair_demand_types.hpp.

7.5.2.206 typedef NbOfRequests_T stdair::NbOfCancellations_T

Define a number of cancellations (travellers).

Definition at line 85 of file stdair_demand_types.hpp.

7.5.2.207 typedef NbOfRequests_T stdair::NbOfNoShows_T

Define a number of no-shows.

Definition at line 88 of file stdair_demand_types.hpp.

7.5.2.208 typedef double stdair::MatchingIndicator_T

Define a indicator of demand to class matching.

Definition at line 91 of file stdair_demand_types.hpp.

7.5.2.209 typedef std::string stdair::DemandStreamKeyStr_T

Type definition for the hashed key of the DemandStreamKey object.

Definition at line 94 of file stdair_demand_types.hpp.

7.5.2.210 typedef std::string stdair::ChannelLabel_T

Type of booking channel (D=direct, I=indirect, N=oNline, F=oFfline).

Definition at line 97 of file stdair_demand_types.hpp.

7.5.2.211 typedef std::string stdair::FrequentFlyer_T

Type of frequent flyer (P=Platinum, G=Gold, S=Silver, M=Member, N=None).

Definition at line 100 of file stdair_demand_types.hpp.

7.5.2.212 typedef std::string stdair::RequestStatus_T

Define the Request status for booking (1-letter-code, e.g., B: booked, C: cancelled, R: Rejected).

Definition at line 104 of file stdair_demand_types.hpp.

7.5.2.213 typedef std::map<Identity_T, Identity_T> stdair::BookingTSIDMap_T

Define a map between a BookingID and a TravelSolutionID.

Definition at line 107 of file stdair_demand_types.hpp.

7.5.2.214 typedef std::pair<CabinCode_T, ClassCode_T> stdair::CabinClassPair_T

Define a pair (cabin code, class code) e.g., (economy, K).

Definition at line 110 of file stdair_demand_types.hpp.

7.5.2.215 typedef std::list<CabinClassPair_T> stdair::CabinClassPairList_T

Define a list of pair (cabin code, class code).

Definition at line 113 of file stdair_demand_types.hpp.

7.5.2.216 typedef double stdair::ProportionFactor_T

Define the forecast booking requests proportion.

Definition at line 116 of file stdair_demand_types.hpp.

7.5.2.217 typedef std::list<ProportionFactor_T> stdair::ProportionFactorList_T

Define the list of forecast booking requests proportions.

Definition at line 119 of file stdair_demand_types.hpp.

7.5.2.218 typedef std::string stdair::OnDString_T

Define the O&D string key (e.g. "SQ;11,2010-Feb-08;SIN,BKK").

Definition at line 122 of file stdair_demand_types.hpp.

7.5.2.219 typedef std::list<OnDString_T> stdair::OnDStringList_T

Define the list of O&D string key.

Definition at line 125 of file stdair_demand_types.hpp.

7.5.2.220 typedef std::string stdair::EventName_T

Define the name of an event.

Definition at line 14 of file stdair_event_types.hpp.

7.5.2.221 typedef double stdair::NbOfEvents_T

Define a number of events.

Definition at line 17 of file stdair_event_types.hpp.

7.5.2.222 typedef std::string stdair::EventGeneratorKey_T

Define a key string of an event generator.

Definition at line 20 of file stdair_event_types.hpp.

7.5.2.223 typedef double stdair::NbOfFareRules_T

Define a number of fare rules.

Definition at line 12 of file stdair_fare_types.hpp.

7.5.2.224 typedef std::string stdair::NetworkID_T

Define the type for network ID.

Definition at line 23 of file stdair_inventory_types.hpp.

7.5.2.225 typedef std::vector<AirlineCode_T> stdair::AirlineCodeList_T

Define a list of airline code.

Definition at line 26 of file stdair_inventory_types.hpp.

7.5.2.226 typedef std::vector<ClassList_String_T> stdair::ClassList_StringList_T

Define the list of list of class codes as a string.

Definition at line 29 of file stdair_inventory_types.hpp.

7.5.2.227 typedef std::vector<ClassCode_T> stdair::ClassCodeList_T

Define a list of class code.

Definition at line 32 of file stdair_inventory_types.hpp.

7.5.2.228 typedef unsigned short stdair::SubclassCode_T

Define the sub-class code (e.g., 0, 1, 2, etc.). The subclass is a sub-structure for the booking class, allowing to have specific rules for some criteria like POS.

Definition at line 37 of file stdair_inventory_types.hpp.

7.5.2.229 typedef std::string stdair::FlightPathCode_T

Define the flight path code (code made by a suite of flight numbers).

Definition at line 40 of file stdair_inventory_types.hpp.

7.5.2.230 typedef std::map<CabinCode_T, ClassList_String_T> stdair::CabinBookingClassMap_T

Map between the cabin codes and the booking class codes within each cabin.

Definition at line 44 of file stdair_inventory_types.hpp.

7.5.2.231 typedef std::string stdair::CurveKey_T

Curve key for FRAT5 or FF Disutility.

Definition at line 47 of file stdair_inventory_types.hpp.

7.5.2.232 typedef double stdair::CabinCapacity_T

Define the cabin capacity (resource, e.g., 200 seats).

The capacity is expressed as a double to cope with overbooking.

Definition at line 51 of file stdair_inventory_types.hpp.

7.5.2.233 typedef double stdair::NbOfFlightDates_T

Define a number of flight dates.

Definition at line 54 of file stdair_inventory_types.hpp.

7.5.2.234 typedef double stdair::CommittedSpace_T

Define the commited space of a cabin.

Definition at line 57 of file stdair_inventory_types.hpp.

7.5.2.235 typedef double stdair::UPR_T

Define the unsold protection (UPR).

Definition at line 60 of file stdair_inventory_types.hpp.

7.5.2.236 typedef double stdair::BookingLimit_T

Define the value of the booking limit.

Definition at line 63 of file stdair_inventory_types.hpp.

7.5.2.237 typedef double stdair::AuthorizationLevel_T

Define the value of the authorization level.

Definition at line 66 of file stdair_inventory_types.hpp.

7.5.2.238 typedef double stdair::CapacityAdjustment_T

Define the value of the adjustment for cabin capacity.

Definition at line 69 of file stdair_inventory_types.hpp.

7.5.2.239 typedef double stdair::BlockSpace_T

Define the number of seat which could not be used for the booking.

Definition at line 72 of file stdair_inventory_types.hpp.

7.5.2.240 typedef bool stdair::AvailabilityStatus_T

Define an availability status (AVS).

Definition at line 75 of file stdair_inventory_types.hpp.

7.5.2.241 typedef std::vector<Availability_T> stdair::BucketAvailabilities_T

Define a list of availabilities.

Definition at line 78 of file stdair_inventory_types.hpp.

7.5.2.242 typedef double stdair::NbOfYields_T

Define a number of yields.

Definition at line 81 of file stdair_inventory_types.hpp.

7.5.2.243 typedef double stdair::NbOfInventoryControlRules_T

Define a number of InventoryControlRules.

Definition at line 84 of file stdair_inventory_types.hpp.

7.5.2.244 typedef bool stdair::CensorshipFlag_T

Define availibility of booking limit.

Definition at line 87 of file stdair_inventory_types.hpp.

7.5.2.245 typedef short stdair::DTD_T

Define the type of day-to-departure.

Definition at line 90 of file stdair_inventory_types.hpp.

7.5.2.246 typedef short stdair::DCP_T

Define the type of data collection point.

Definition at line 93 of file stdair_inventory_types.hpp.

7.5.2.247 typedef std::list<DCP_T> stdair::DCPList_T

Define the type of data collection point list.

Definition at line 96 of file stdair_inventory_types.hpp.

7.5.2.248 typedef std::map<DTD_T, RealNumber_T> stdair::DTDFratMap_T

Define the DTD (days to departure) frat5 coef map.

Definition at line 99 of file stdair_inventory_types.hpp.

7.5.2.249 typedef std::map<FloatDuration_T, float> stdair::DTDProbMap_T

Define the DTD (days to departure) probability map.

Definition at line 102 of file stdair_inventory_types.hpp.

7.5.2.250 typedef std::vector<CensorshipFlag_T> stdair::CensorshipFlagList_T

Define the list of censorship flags (une list per booking class, one censorship flag per DCP).

Definition at line 106 of file stdair_inventory_types.hpp.

7.5.2.251 typedef double stdair::BookingRatio_T

Define the bookingRatio (for instance OnD bookings over whole class bookings).

Definition at line 110 of file stdair_inventory_types.hpp.

7.5.2.252 typedef double stdair::Yield_T

Define the yield of a virtual class.

Definition at line 113 of file stdair_inventory_types.hpp.

7.5.2.253 typedef unsigned int stdair::YieldLevel_T

Define the yield level (yield as an integer).

Definition at line 116 of file stdair_inventory_types.hpp.

7.5.2.254 typedef std::map<YieldLevel_T, MeanStdDevPair_T> stdair::YieldLevelDemandMap_T

Define the <YieldLevel, demand> demand map.

Definition at line 119 of file stdair_inventory_types.hpp.

7.5.2.255 typedef std::pair<Yield_T, MeanStdDevPair_T> stdair::YieldDemandPair_T

Define the <Yield, demand> pair type.

Definition at line 122 of file stdair_inventory_types.hpp.

7.5.2.256 typedef double stdair::BidPrice_T

Define the Bid-Price.

Definition at line 125 of file stdair_inventory_types.hpp.

7.5.2.257 typedef std::vector<BidPrice_T> stdair::BidPriceVector_T

Define a Bid-Price Vector.

Definition at line 128 of file stdair_inventory_types.hpp.

7.5.2.258 typedef unsigned int stdair::SeatIndex_T

Define the current index of a Bid-Price Vector (for a given LegCabin).

Definition at line 131 of file stdair_inventory_types.hpp.

7.5.2.259 typedef std::string stdair::ControlMode_T

Mode of inventory control.

Definition at line 134 of file stdair_inventory_types.hpp.

7.5.2.260 typedef double stdair::OverbookingRate_T

Define the rate of overbooking

Definition at line 137 of file stdair_inventory_types.hpp.

7.5.2.261 typedef double stdair::BookingLimit_T

Define the Booking Limit.

It is a double, as it allows for overbooking.

Definition at line 141 of file stdair_inventory_types.hpp.

7.5.2.262 typedef double stdair::ProtectionLevel_T

Define the Protection Level.

It is a double, as it allows for overbooking.

Definition at line 145 of file stdair_inventory_types.hpp.

7.5.2.263 typedef std::vector<double> stdair::EmsrValueList_T

Define the list of EMSR values for the EMSR algorithm.

Definition at line 148 of file stdair_inventory_types.hpp.

7.5.2.264 typedef std::vector<double> stdair::BookingLimitVector_T

Define the vector of booking limits.

It is a vector of double.

Definition at line 152 of file stdair_inventory_types.hpp.

7.5.2.265 typedef std::vector<double> stdair::ProtectionLevelVector_T

Define the vector of protection levels.

It is a vector of double.

Definition at line 156 of file stdair_inventory_types.hpp.

7.5.2.266 typedef boost::multi_array<double, 2> stdair::SnapshotBlock_T

Define a snapshot block.

Definition at line 159 of file stdair_inventory_types.hpp.

7.5.2.267 typedef SnapshotBlock_T::index_range stdair::SnapshotBlockRange_T

Define a range for array view.

Definition at line 162 of file stdair_inventory_types.hpp.

7.5.2.268 typedef SnapshotBlock_T::array_view<1>::type stdair::SegmentCabinDTDSnapshotView_T

Define a view for a given DTD.

Definition at line 165 of file stdair_inventory_types.hpp.

7.5.2.269 typedef SnapshotBlock_T::array_view<2>::type stdair::SegmentCabinDTDRangeSnapshotView_T

Define a view for a given range of DTD.

Definition at line 168 of file stdair_inventory_types.hpp.

7.5.2.270 typedef SnapshotBlock_T::const_array_view<1>::type stdair::ConstSegmentCabinDTDSnapshotView_T

Define a const view for a given DTD.

Definition at line 171 of file stdair_inventory_types.hpp.

7.5.2.271 typedef SnapshotBlock_T::const_array_view<2>::type stdair::ConstSegmentCabinDTDRangeSnapshotView_T

Define a const view for a given range of DTD.

Definition at line 174 of file stdair_inventory_types.hpp.

7.5.2.272 typedef unsigned short stdair::SegmentDataID_T

Define the segment ID within a snapshot data table.

Definition at line 177 of file stdair_inventory_types.hpp.

7.5.2.273 typedef unsigned short stdair::LegDataID_T

Define the leg ID within a snapshot data table.

Definition at line 180 of file stdair_inventory_types.hpp.

7.5.2.274 typedef unsigned short stdair::ClassIndex_T

Define the index type of a class within a snapshot block of a leg/segment.

Definition at line 184 of file stdair_inventory_types.hpp.

7.5.2.275 typedef unsigned int stdair::ReplicationNumber_T

Define the replication number.

Definition at line 24 of file stdair_maths_types.hpp.

7.5.2.276 typedef unsigned long int stdair::ExponentialSeed_T

Define the seed type of an Exponential function.

Definition at line 29 of file stdair_maths_types.hpp.

7.5.2.277 typedef unsigned long int stdair::UniformSeed_T

Define the seed type of an Uniform function.

Definition at line 34 of file stdair_maths_types.hpp.

7.5.2.278 typedef unsigned long int stdair::RandomSeed_T

Seed for the random generation, so that it can be reproductible.

Definition at line 39 of file stdair_maths_types.hpp.

7.5.2.279 typedef boost::minstd_rand stdair::BaseGenerator_T

Random number generator.

Definition at line 44 of file stdair_maths_types.hpp.

7.5.2.280 typedef boost::uniform_real stdair::UniformDistribution_T

Uniform distribution of real numbers (by default, double).

Definition at line 49 of file stdair_maths_types.hpp.

7.5.2.281 typedef boost::variate_generator<BaseGenerator_T&, UniformDistribution_T> stdair::UniformGenerator_T

Uniform random generator.

Definition at line 55 of file stdair_maths_types.hpp.

7.5.2.282 typedef boost::normal_distribution stdair::NormalDistribution_T

Normal distribution of real numbers (by default, double).

Definition at line 60 of file stdair_maths_types.hpp.

7.5.2.283 typedef boost::variate_generator<BaseGenerator_T&, NormalDistribution_T> stdair::NormalGenerator_T

Nornal random generator.

Definition at line 66 of file stdair_maths_types.hpp.

7.5.2.284 typedef boost::exponential_distribution stdair::ExponentialDistribution_T

Type definiton for the exponential distribution (characteristics).

Definition at line 69 of file stdair_maths_types.hpp.

7.5.2.285 typedef boost::variate_generator<BaseGenerator_T&, ExponentialDistribution_T> stdair::ExponentialGenerator_T

Type definition for the exponential distribution random generator.

Definition at line 74 of file stdair_maths_types.hpp.

7.5.2.286 typedef double stdair::MeanValue_T

Define a mean value (e.g., 20.2).

Definition at line 79 of file stdair_maths_types.hpp.

7.5.2.287 typedef double stdair::StdDevValue_T

Define a standard deviation value (e.g., 1.5).

Definition at line 84 of file stdair_maths_types.hpp.

7.5.2.288 typedef std::pair<MeanValue_T, StdDevValue_T> stdair::MeanStdDevPair_T

Define a couple (mean, standart deviation) (e.g., (20.2,1.5)).

Definition at line 89 of file stdair_maths_types.hpp.

7.5.2.289 typedef std::vector<MeanStdDevPair_T> stdair::MeanStdDevPairVector_T

Define a vector of couple (mean, standart deviation)

Definition at line 94 of file stdair_maths_types.hpp.

7.5.2.290 typedef float stdair::Probability_T

Probability.

Definition at line 99 of file stdair_maths_types.hpp.

7.5.2.291 typedef std::string stdair::ForecasterMode_T

Mode of the forecaster.

Definition at line 17 of file stdair_rm_types.hpp.

7.5.2.292 typedef short stdair::HistoricalDataLimit_T

Limit of similar flight-dates used in the forecaster.

Definition at line 24 of file stdair_rm_types.hpp.

7.5.2.293 typedef std::string stdair::OptimizerMode_T

Mode of the forecaster.

Definition at line 27 of file stdair_rm_types.hpp.

7.5.2.294 typedef NbOfBookings_T stdair::PolicyDemand_T

Define the demand for a policy.

Definition at line 30 of file stdair_rm_types.hpp.

7.5.2.295 typedef std::vector<double> stdair::GeneratedDemandVector_T

Define the vector of generated demand (for MC integration use).

It is a vector of double.

Definition at line 34 of file stdair_rm_types.hpp.

7.5.2.296 typedef std::vector<GeneratedDemandVector_T> stdair::GeneratedDemandVector-Holder_T

Define the holder of the generated demand vectors.

Definition at line 37 of file stdair_rm_types.hpp.

7.5.2.297 typedef double stdair::SellupProbability_T

Define the sellup probability.

Definition at line 40 of file stdair_rm_types.hpp.

7.5.2.298 typedef std::vector<NbOfRequests_T> stdair::UncDemVector_T

Define the vector of historical unconstrained demand.

Definition at line 43 of file stdair_rm_types.hpp.

7.5.2.299 typedef std::vector<NbOfBookings_T> stdair::BookingVector_T

Define the vector of historical bookings.

Definition at line 46 of file stdair_rm_types.hpp.

7.5.2.300 typedef double stdair::FRAT5_T

Define the FRAT5 coefficient.

Definition at line 49 of file stdair_rm_types.hpp.

7.5.2.301 typedef std::map<const DTD_T, FRAT5_T> stdair::FRAT5Curve_T

Define the FRAT5 curve.

Definition at line 52 of file stdair_rm_types.hpp.

7.5.2.302 typedef std::map<const DTD_T, double> stdair::FFDisutilityCurve_T

Define the fare family disutility curve.

Definition at line 55 of file stdair_rm_types.hpp.

7.5.2.303 typedef std::map<const DTD_T, double> stdair::SellUpCurve_T

Define the sell-up factor curve.

Definition at line 58 of file stdair_rm_types.hpp.

7.5.2.304 typedef std::map<const DTD_T, double> stdair::DispatchingCurve_T

Define the dispatching factor curve.

Definition at line 61 of file stdair_rm_types.hpp.

7.5.2.305 typedef std::map<BookingClass*, SellUpCurve_T> stdair::BookingClassSellUpCurveMap_T

Define the map between class and sell-up factor curve.

Definition at line 64 of file stdair_rm_types.hpp.

7.5.2.306 typedef std::map<BookingClass*, DispatchingCurve_T> stdair::BookingClassDispatchingCurveMap_T

Define the map between class and dispatching factor curve.

Definition at line 67 of file stdair_rm_types.hpp.

7.5.2.307 typedef std::map<const Yield_T, double> stdair::YieldDemandMap_T

Define the map between the yield of a class and the demand forecast of this class within a policy.

Definition at line 71 of file stdair_rm_types.hpp.

7.5.2.308 typedef double stdair::Revenue_T

Define the revenue of a policy

Definition at line 74 of file stdair_rm_types.hpp.

7.5.2.309 typedef unsigned int stdair::NbOfSamples_T

Define the number of samples for the generated demand of booking class

Definition at line 77 of file stdair_rm_types.hpp.

7.5.2.310 typedef boost::shared_ptr<STDAIR_Service> stdair::STDAIR_ServicePtr_T

Pointer on the STDAIR Service handler.

Definition at line 13 of file stdair_service_types.hpp.

7.5.3 Function Documentation

7.5.3.1 const std::string stdair::DEFAULT_BOM_ROOT_KEY (" – ROOT – ")

Default value for the BOM tree root key (" – ROOT – ").

7.5.3.2 const double stdair::DEFAULT_EPSILON_VALUE (0. *0001*)

Default very small value.

7.5.3.3 const unsigned int stdair::DEFAULT_FLIGHT_SPEED (900)

Default flight speed (number of kilometers per hour).

7.5.3.4 const NbOfFlightDates_T stdair::DEFAULT_NB_OF_FLIGHTDATES (0. *0*)

Default number of generated flight dates.

7.5.3.5 const Duration_T stdair::NULL_BOOST_TIME_DURATION (- *1*, - *1*, - *1*)

Null time duration (in boost::time_duration unit).

7.5.3.6 const Duration_T stdair::DEFAULT_NULL_DURATION (0, 0, 0)

Default null duration (in boost::time_duration unit).

7.5.3.7 const unsigned int stdair::DEFAULT_NB_OF_DAYS_IN_A_YEAR (365)

Default number of days in a year.

7.5.3.8 const unsigned int stdair::DEFAULT_NUMBER_OF_SUBDIVISIONS (1000)

Higher value per thousand

7.5.3.9 const DayDuration_T stdair::DEFAULT_DAY_DURATION (0)

Default number of duration days.

7.5.3.10 const DatePeriod_T stdair::BOOST_DEFAULT_DATE_PERIOD (Date_T(2007, 1, 1), Date_T(2007, 1, 1))

Default date period (0-length, i.e., it lasts one day).

7.5.3.11 const DOW_String_T stdair::DEFAULT_DOW_STRING ("0000000")

Default DOW String (e.g., "0000000").

7.5.3.12 const DateOffset_T stdair::DEFAULT_DATE_OFFSET (0)

Default Date Offset (e.g., 0).

7.5.3.13 const Date_T stdair::DEFAULT_DATE (2010, boost::gregorian::Jan, 1)

Default date for the General.

7.5.3.14 const DateTime_T stdair::DEFAULT_DATETIME (DEFAULT_DATE, NULL_BOOST_TIME_DURATION)

Default date-time.

7.5.3.15 const Duration_T stdair::DEFAULT_EPSILON_DURATION (0, 0, 0, 1)

Default epsilon duration (1 nanosecond).

7.5.3.16 const Count_T stdair::SECONDS_IN_ONE_DAY (86400)

Number of seconds in one day.

7.5.3.17 const Count_T stdair::MILLISECONDS_IN_ONE_SECOND (1000)

Number of milliseconds in one second

7.5.3.18 const RandomSeed_T stdair::DEFAULT_RANDOM_SEED (120765987)

Default random seed.

7.5.3.19 const AirportCode_T stdair::AIRPORT_LHR ("LHR")

Default origin airport (e.g., "LHR").

7.5.3.20 const AirportCode_T stdair::AIRPORT_SYD ("SYD")

Default destination airport (e.g., "SYD").

7.5.3.21 const CityCode_T stdair::POS_LHR ("LHR")

London city code (e.g., "LHR").

7.5.3.22 const Date_T stdair::DATE_20110115 (2011, boost::gregorian::Jan, 15)

Date.

7.5.3.23 const Date_T stdair::DATE_20111231 (2011, boost::gregorian::Dec, 31)

7.5.3.24 const DayDuration_T stdair::NO_ADVANCE_PURCHASE (0)

Advance purchase 0 day.

7.5.3.25 const SaturdayStay_T stdair::SATURDAY_STAY (true)

Default saturdayStay value (true).

7.5.3.26 const SaturdayStay_T stdair::NO_SATURDAY_STAY (false)

Default saturdayStay value (false).

7.5.3.27 const ChangeFees_T stdair::CHANGE_FEES (true)

Default change fees value (true).

7.5.3.28 const ChangeFees_T stdair::NO_CHANGE_FEES (false)

Default change fees value (false).

7.5.3.29 const NonRefundable_T stdair::NON_REFUNDABLE (true)

Default non refundable value (true).

7.5.3.30 const NonRefundable_T stdair::NO_NON_REFUNDABLE (false)

Default refundable value (false).

7.5.3.31 const SaturdayStay_T stdair::DEFAULT_BOM_TREE_SATURDAY_STAY (true)

Default saturdayStay value (true).

7.5.3.32 const ChangeFees_T stdair::DEFAULT_BOM_TREE_CHANGE_FEES (true)

Default change fees value (true).

7.5.3.33 const NonRefundable_T stdair::DEFAULT_BOM_TREE_NON_REFUNDABLE (true)

Default non refundable value (true).

7.5.3.34 const DayDuration_T stdair::NO_STAY_DURATION (0)

Stay duration 0 day.

7.5.3.35 const AirlineCode_T stdair::AIRLINE_CODE_BA ("BA")

Airline code "BA".

7.5.3.36 const CabinCode_T stdair::CABIN_Y ("Y")

Cabin 'Y'.

7.5.3.37 const ClassCode_T stdair::CLASS_CODE_Y ("Y")

Class code 'Y'.

7.5.3.38 const ClassCode_T stdair::CLASS_CODE_Q ("Q")

Class code 'Q'.

7.5.3.39 const AirportCode_T stdair::AIRPORT_SIN ("SIN")

Singapour airport (e.g., "SIN").

7.5.3.40 const AirportCode_T stdair::AIRPORT_BKK ("BKK")

Bangkok airport (e.g., "BKK").

7.5.3.41 const CityCode_T stdair::POS_SIN ("SIN")

Singapour city code (e.g., "SIN").

7.5.3.42 const CabinCode_T stdair::CABIN_ECO ("Eco")

Economic cabin (e.g., "Eco").

7.5.3.43 const FrequentFlyer_T stdair::FREQUENT_FLYER_MEMBER ("M")

Frequent flyer tier (e.g., "M" meaning member).

7.5.3.44 const FamilyCode_T stdair::DEFAULT_FAMILY_CODE ("0")

Default family code value ("0").

7.5.3.45 const PolicyCode_T stdair::DEFAULT_POLICY_CODE ("0")

Default policy code value ("0").

7.5.3.46 const NestingStructureCode_T stdair::DEFAULT_NESTING_STRUCTURE_CODE ("DEFAULT")

Default Nesting Structure Code ("DEFAULT").

7.5.3.47 const NestingStructureCode_T stdair::DISPLAY_NESTING_STRUCTURE_CODE ("Display Nesting")

Display Nesting Structure Code ("Display Nesting").

7.5.3.48 const NestingStructureCode_T stdair::YIELD_BASED_NESTING_STRUCTURE_CODE ("Yield-Based Nesting")

Display Nesting Structure Code ("Yield-Based Nesting").

7.5.3.49 const NestingNodeCode_T stdair::DEFAULT_NESTING_NODE_CODE ("0")

Default Nesting Node Code ("0").

7.5.3.50 const NbOfAirlines_T stdair::DEFAULT_NBOFAIRLINES (0)

Default number of airlines.

7.5.3.51 const FlightPathCode_T stdair::DEFAULT_FLIGHTPATH_CODE ("")

Default flight-path code value ("").

7.5.3.52 const Distance_T stdair::DEFAULT_DISTANCE_VALUE (0)

Default distance value (kilometers).

7.5.3.53 const ClassCode_T stdair::DEFAULT_CLOSED_CLASS_CODE ("CC")

Default closed class code.

7.5.3.54 const NbOfBookings_T stdair::DEFAULT_CLASS_NB_OF_BOOKINGS (0)

Default number of bookings (with counted cancellation) for BookingClass.

7.5.3.55 const NbOfBookings_T stdair::DEFAULT_CLASS_TOTAL_NB_OF_BOOKINGS (0)

Default number of booking (without cancellation) demands for BookingClass.

7.5.3.56 const NbOfBookings_T stdair::DEFAULT_CLASS_UNCONSTRAINED_DEMAND (0)

Default unconstrained demand for BookingClass.

7.5.3.57 const NbOfBookings_T stdair::DEFAULT_CLASS_REMAINING_DEMAND_MEAN (0)

Default remaining future demand mean for BookingClass.

7.5.3.58 const NbOfBookings_T stdair::DEFAULT_CLASS_REMAINING_DEMAND_STANDARD_DEVIATION (0)

Default remaining futre demand standard deviation for BookingClass.

7.5.3.59 const NbOfCancellations_T stdair::DEFAULT_CLASS_NB_OF_CANCELLATIONS (0)

Default number of cancellations for BookingClass.

7.5.3.60 const NbOfNoShows_T stdair::DEFAULT_CLASS_NB_OF_NOSHOWS (0)

Default number of no-shows for BookingClass.

7.5.3.61 const CabinCapacity_T stdair::DEFAULT_CABIN_CAPACITY (100. *0*)

Default cabin capacity for Leg cabins.

7.5.3.62 const CommittedSpace_T stdair::DEFAULT_COMMITTED_SPACE (0. *0*)

Default committed space value for Leg cabins.

7.5.3.63 const BlockSpace_T stdair::DEFAULT_BLOCK_SPACE (0. *0*)

Default commited space value for Leg cabins.

7.5.3.64 const Availability_T stdair::DEFAULT_NULL_AVAILABILITY (0. *0*)

Default null availability (0.0).

7.5.3.65 const Availability_T stdair::DEFAULT_AVAILABILITY (9. *0*)

Default availability (9.0).

7.5.3.66 const Availability_T stdair::MAXIMAL_AVAILABILITY (9999. *0*)

Maximal offered capacity in a cabin.

7.5.3.67 const CensorshipFlag_T stdair::DEFAULT_CLASS_CENSORSHIPFLAG (false)

Default boolean for censorship flag given the status of availability for BookingClass.

7.5.3.68 const BookingLimit_T stdair::DEFAULT_CLASS_BOOKING_LIMIT (9999. *0*)

Default booking limit value for BookingClass.

7.5.3.69 const AuthorizationLevel_T stdair::DEFAULT_CLASS_AUTHORIZATION_LEVEL (9999. *0*)

Default authorization level for BookingClass.

7.5.3.70 const AuthorizationLevel_T stdair::DEFAULT_CLASS_MAX_AUTHORIZATION_LEVEL (9999. *0*)

Default MAX value of authorization level for BookingClass.

7.5.3.71 const AuthorizationLevel_T stdair::DEFAULT_CLASS_MIN_AUTHORIZATION_LEVEL (0. *0*)

Default MIN value of authorization level for BookingClass.

7.5.3.72 const OverbookingRate_T stdair::DEFAULT_CLASS_OVERBOOKING_RATE (0. *0*)

Default over-booking rate for BookingClass.

7.5.3.73 const BookingRatio_T stdair::DEFAULT_OND_BOOKING_RATE (0. *0*)

Default booking rate for OnD bookings over overall class bookings.

7.5.3.74 const Fare_T stdair::DEFAULT_FARE_VALUE (0. *0*)

Default Fare value.

7.5.3.75 const Yield_T stdair::DEFAULT_CLASS_YIELD_VALUE (0. *0*)

Default yield value for a virtual class.

7.5.3.76 const Revenue_T stdair::DEFAULT_REVENUE_VALUE (0. *0*)

Default Revenue value.

7.5.3.77 const Percentage_T stdair::DEFAULT_LOAD_FACTOR_VALUE (100. *0*)

Default load factor value (100%).

7.5.3.78 const Yield_T stdair::DEFAULT_YIELD_VALUE (0. *0*)

Default yield value.

7.5.3.79 const Yield_T stdair::DEFAULT_YIELD_MAX_VALUE (std::numeric_limits< double >:: *max*())

Default yield max value.

7.5.3.80 const NbOfBookings_T stdair::DEFAULT_YIELD_NB_OF_BOOKINGS (0. *0*)

Default number of bookings for YieldRangeStruct_T.

7.5.3.81 const Identity_T stdair::DEFAULT_BOOKING_NUMBER (0)

Default booking number.

7.5.3.82 const NbOfCancellations_T stdair::DEFAULT_YIELD_NB_OF_CANCELLATIONS (0. *0*)

Default cancellation number for YieldRangeStruct_T.

7.5.3.83 const NbOfNoShows_T stdair::DEFAULT_YIELD_NB_OF_NOSHOWS (0. *0*)

Default no-shows number for YieldRangeStruct_T.

7.5.3.84 const Availability_T stdair::DEFAULT_YIELD_AVAILABILITY (0. *0*)

Default availability for YieldRangeStruct_T.

7.5.3.85 const CensorshipFlag_T stdair::DEFAULT_YIELD_CENSORSHIPFLAG (false)

Default boolean for booking limit availability for YieldRangeStruct_T.

7.5.3.86 const BookingLimit_T stdair::DEFAULT_YIELD_BOOKING_LIMIT (0. *0*)

Default booking limit value for YieldRangeStruct_T.

7.5.3.87 const OverbookingRate_T stdair::DEFAULT_YIELD_OVERBOOKING_RATE (0. *0*)

Default over-booking rate for YieldRangeStruct_T.

7.5.3.88 const Fare_T stdair::DEFAULT_OND_FARE_VALUE (0. *0*)

Default value of Fare.

7.5.3.89 const Count_T stdair::DEFAULT_PROGRESS_STATUS (0)

Default progress status.

7.5.3.90 const Percentage_T stdair::MAXIMUM_PROGRESS_STATUS (100)

Maximum progress status.

7.5.3.91 const Date_T stdair::DEFAULT_EVENT_OLDEST_DATE (2008, boost::gregorian::Jan, 1)

Default reference (oldest) date for the events. No event can occur before that date.

7.5.3.92 const DateTime_T stdair::DEFAULT_EVENT_OLDEST_DATETIME (DEFAULT_-EVENT_OLDEST_DATE, NULL_BOOST_TIME_DURATION)

Default reference (oldest) date-time for the events. No event can occur before that date-time.

7.5.3.93 const PartySize_T stdair::DEFAULT_PARTY_SIZE (1)

Default party size in a request.

7.5.3.94 const DayDuration_T stdair::DEFAULT_STAY_DURATION (7)

Default duration for a stay.

7.5.3.95 const WTP_T stdair::DEFAULT_WTP (1000. *0*)

Default Willingness-to-Pay (WTP, as expressed as a monetary unit).

7.5.3.96 const Date_T stdair::DEFAULT_PREFERRED_DEPARTURE_DATE (DEFAULT_-DEPARTURE_DATE)

Default departure date.

7.5.3.97 const Duration_T stdair::DEFAULT_PREFERRED_DEPARTURE_TIME (8, 0, 0)

Default preferred departure time (08:00).

7.5.3.98 const DateOffset_T stdair::DEFAULT_ADVANCE_PURCHASE (22)

Default advance purchase.

7.5.3.99 const Date_T stdair::DEFAULT_REQUEST_DATE (DEFAULT_PREFERRED_-DEPARTURE_DATE- *DEFAULT_ADVANCE_PURCHASE*)

Default request date.

7.5.3.100 const Duration_T stdair::DEFAULT_REQUEST_TIME (8, 0, 0)

Default preferred departure time (08:00).

7.5.3.101 const DateTime_T stdair::DEFAULT_REQUEST_DATE_TIME (DEFAULT_-REQUEST_DATE, DEFAULT_REQUEST_TIME)

Default request date-time.

7.5.3.102 const CabinCode_T stdair::DEFAULT_PREFERRED_CABIN ("M")

Default preferred cabin.

7.5.3.103 const CityCode_T stdair::DEFAULT_POS ("ALL")

Default point-of-sale.

7.5.3.104 const ChannelLabel_T stdair::DEFAULT_CHANNEL ("DC")

Default channel (e.g., "DC" meaning Different Channels).

7.5.3.105 const ChannelLabel_T stdair::CHANNEL_DN ("DN")

DN channel (e.g., direct on-line).

7.5.3.106 const ChannelLabel_T stdair::CHANNEL_IN ("IN")

IN channel (e.g., indirect on-line).

7.5.3.107 const TripType_T stdair::TRIP_TYPE_ONE_WAY ("OW")

Trip type one-way (e.g., "OW").

7.5.3.108 const TripType_T stdair::TRIP_TYPE_ROUND_TRIP ("RT")

Trip type round-trip (e.g., "RT").

7.5.3.109 const TripType_T stdair::TRIP_TYPE_INBOUND ("RI")

Trip type inbound (e.g., "RI").

7.5.3.110 const TripType_T stdair::TRIP_TYPE_OUTBOUND ("RO")

Trip type outbound (e.g., "RO").

7.5.3.111 const FrequentFlyer_T stdair::DEFAULT_FF_TIER ("N")

Default frequent flyer tier (non member).

7.5.3.112 const PriceValue_T stdair::DEFAULT_VALUE_OF_TIME (100. *0*)

Default value of time (expressed as a monetary unit per hour).

7.5.3.113 const IntDuration_T stdair::HOUR_CONVERTED_IN_SECONDS (3600)

Number of second in one hour

7.5.3.114 const Duration_T stdair::DEFAULT_MINIMAL_CONNECTION_TIME (0, 30, 0)

Default Minimal connection time.

7.5.3.115 const Duration_T stdair::DEFAULT_MAXIMAL_CONNECTION_TIME (24, 0, 0)

Default maximal connection time.

7.5.3.116 const MatchingIndicator_T stdair::DEFAULT_MATCHING_INDICATOR (0. *0*)

Default Matching Indicator value.

7.5.3.117 const PriceCurrency_T stdair::DEFAULT_CURRENCY ("EUR")

Default currency (euro).

7.5.3.118 const AvailabilityStatus_T stdair::DEFAULT_AVAILABILITY_STATUS (false)

Default availability status for a travel solution.

7.5.3.119 const AirlineCode_T stdair::DEFAULT_AIRLINE_CODE ("XX")

Default airline code value ("XX").

7.5.3.120 const AirlineCode_T stdair::DEFAULT_NULL_AIRLINE_CODE ("")

Default airline code value ("").

7.5.3.121 const FlightNumber_T stdair::DEFAULT_FLIGHT_NUMBER (9999)

Default flight number (9999).

7.5.3.122 const FlightNumber_T stdair::DEFAULT_FLIGHT_NUMBER_FF (255)

Default flight number for fare families (255).

7.5.3.123 const TableID_T stdair::DEFAULT_TABLE_ID (9999)

Default data table number (9999).

7.5.3.124 const Date_T stdair::DEFAULT_DEPARTURE_DATE (1900, boost::gregorian::Jan, 1)

Default flight departure date (01/01/1900).

7.5.3.125 const AirportCode_T stdair::DEFAULT_AIRPORT_CODE ("XXX")

Default airport code value ("XXX").

7.5.3.126 const AirportCode_T stdair::DEFAULT_NULL_AIRPORT_CODE ("")

Default airport code value (")..

7.5.3.127 const AirportCode_T stdair::DEFAULT_ORIGIN ("XXX")

Default Origin.

7.5.3.128 const AirportCode_T stdair::DEFAULT_DESTINATION ("YYY")

Default destination.

7.5.3.129 const CabinCode_T stdair::DEFAULT_CABIN_CODE ("X")

Default cabin code.

7.5.3.130 const FamilyCode_T stdair::DEFAULT_FARE_FAMILY_CODE ("EcoSaver")

Default fare family Code.

7.5.3.131 const FamilyCode_T stdair::DEFAULT_NULL_FARE_FAMILY_CODE ("NoFF")

Default null fare family Code ("NoFF").

7.5.3.132 const ClassCode_T stdair::DEFAULT_CLASS_CODE ("X")

Default class code value ("X").

7.5.3.133 const ClassCode_T stdair::DEFAULT_NULL_CLASS_CODE ("")

Default null class code value ("").

7.5.3.134 const BidPrice_T stdair::DEFAULT_BID_PRICE (0. *0*)

Default Bid-Price.

7.5.3.135 const unsigned short stdair::MAXIMAL_NUMBER_OF_LEGS_IN_FLIGHT (7)

Maximal number of legs linked to a single flight-date.

Note that the number of derived segments is n∗(n+1)/2 if n is the number of legs.

7.5.3.136 const unsigned short stdair::MAXIMAL_NUMBER_OF_SEGMENTS_IN_OND (3)

Maximal number of segments linked to a single O&D (Origin & Destination).

7.5.3.137 const SeatIndex_T stdair::DEFAULT_SEAT_INDEX (1)

Default seat index (for a bucket and/or Bid-Price Vector slot).

7.5.3.138 const NbOfSeats_T stdair::DEFAULT_NULL_BOOKING_NUMBER (0)

Default number of bookings.

7.5.3.139 const CapacityAdjustment_T stdair::DEFAULT_NULL_CAPACITY_ADJUSTMENT (0)

Default capacity adjustment of the cabin.

7.5.3.140 const UPR_T stdair::DEFAULT_NULL_UPR (0)

Default unsold Protection (UPR).

7.5.3.141 const std::string stdair::DEFAULT_FARE_FAMILY_VALUE_TYPE ("FF")

Default value type (within a guillotine block) for fare family.

7.5.3.142 const std::string stdair::DEFAULT_SEGMENT_CABIN_VALUE_TYPE ("SC")

Default value type (within a guillotine block) for segment-cabin.

7.5.3.143 const std::string stdair::DEFAULT_KEY_FLD_DELIMITER (";")

Default delimiter for string display (e.g delimiter for inventory key and flight-date key).

7.5.3.144 const std::string stdair::DEFAULT_KEY_SUB_FLD_DELIMITER (", ")

Default sub delimiter for string display (e.g delimiter for flight number and departure date of a flight-date key).

7.5.3.145 const boost::char_separator<char> stdair::DEFAULT_KEY_TOKEN_DELIMITER (";, ")

Default token for decoding a full string display.

7.5.3.146 template<int MIN, int MAX> date_time_element<MIN, MAX> stdair::operator * (const date_time_element< MIN, MAX > & *o1*, const date_time_element< MIN, MAX > & *o2*) [inline]

Operator* overload.

Definition at line 47 of file BasParserHelperTypes.hpp.

References stdair::date_time_element< MIN, MAX >::_value.

7.5.3.147 template<int MIN, int MAX> date_time_element<MIN, MAX> stdair::operator+ (const date_time_element< MIN, MAX > & *o1*, const date_time_element< MIN, MAX > & *o2*) [inline]

Operator+ overload.

Definition at line 55 of file BasParserHelperTypes.hpp.

References stdair::date_time_element< MIN, MAX >::_value.

7.5.3.148 template void stdair::AirlineClassListKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned *int*)

7.5.3.149 template void stdair::AirlineClassListKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned *int*)

7.5.3.150 template void stdair::BomRootKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned *int*)

7.5.3.151 template void stdair::BomRootKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned *int*)

7.5.3.152 void stdair::intDisplay (std::ostream & *oStream*, const int & *iInt*)

Definition at line 159 of file BookingRequestStruct.cpp.

7.5.3.153 template void stdair::BucketKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned *int*)

7.5.3.154 template void stdair::BucketKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned *int*)

7.5.3.155 template void stdair::FareFamilyKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned *int*)

7.5.3.156 template void stdair::FareFamilyKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned *int*)

7.5.3.157 template void stdair::FlightDateKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned *int*)

7.5.3.158 template void stdair::FlightDateKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned *int*)

7.5.3.159 template void stdair::InventoryKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned *int*)

7.5.3.160 template void stdair::InventoryKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned *int*)

7.5.3.161 template void stdair::NestingNodeKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned *int*)

7.5.3.162 template void stdair::NestingNodeKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned *int*)

7.5.3.163 template void stdair::NestingStructureKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned *int*)

7.5.3.164 template void stdair::NestingStructureKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned *int*)

7.5.3.165 template void stdair::OnDDateKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned *int*)

7.5.3.166 template void stdair::OnDDateKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned *int*)

7.5.3.167 const boost::char_separator<char> stdair::TokeniserDashSeparator ("-")

Dash delimitor for the tokenisation process.

Referenced by stdair::ParsedKey::getFlightDateKey().

7.5.3.168 const boost::char_separator<char> stdair::TokeniserTimeSeparator (":")

Time delimitor for the tokenisation process.

Referenced by stdair::ParsedKey::getBoardingTime().

7.5.3.169 template void stdair::PolicyKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned *int*)

7.5.3.170 template void stdair::PolicyKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned *int*)

7.5.3.171 template void stdair::SegmentCabinKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned *int*)

7.5.3.172 template void stdair::SegmentCabinKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned *int*)

7.5.3.173 template void stdair::SegmentDateKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned *int*)

7.5.3.174 template void stdair::SegmentDateKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned *int*)

7.5.3.175 template void stdair::SegmentSnapshotTableKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned *int*)

7.5.3.176 template void stdair::SegmentSnapshotTableKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned *int*)

7.5.3.177 template<class Archive, class BOM_OBJECT1, class BOM_OBJECT2> void stdair::serialiseHelper (BOM_OBJECT1 & *ioObject1*, Archive & *ioArchive*, const unsigned int *iFileVersion*)

Definition at line 34 of file CmdBomSerialiser.cpp.

References stdair::BomHolder< BOM >::_bomList, stdair::BomHolder< BOM >::_bomMap, and stdair::FacBomManager::linkWithParent().

7.5.3.178 template void stdair::BomRoot::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned *int*)

7.5.3.179 template void stdair::BomRoot::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned *int*)

7.5.3.180 template void stdair::Inventory::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned *int*)

7.5.3.181 template void stdair::Inventory::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned *int*)

7.5.3.182 template void stdair::FlightDate::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned *int*)

7.5.3.183 template void stdair::FlightDate::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned *int*)

7.5.3.184 template void stdair::SegmentDate::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned *int*)

7.5.3.185 template void stdair::SegmentDate::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned *int*)

7.5.3.186 template void stdair::SegmentCabin::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned *int*)

7.5.3.187 template void stdair::SegmentCabin::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned *int*)

7.5.4 Variable Documentation

7.5.4.1 const std::string stdair::DOW_STR[ ]

Initial value:

```
{"Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"}
```

Day names (in English).

Definition at line 53 of file BasConst.cpp.

Referenced by stdair::DoWStruct::describe().

7.5.4.2 const UnconstrainingMethod stdair::DEFAULT_UNCONSTRAINING_METHOD('E')

Default Unconstraining Method (By Expectation-Maximisation).

7.5.4.3 const PartnershipTechnique stdair::DEFAULT_PARTNERSHIP_TECHNIQUE('N')

Default Partnership Technique (None).

7.5.4.4 const ForecastingMethod stdair::DEFAULT_FORECASTING_METHOD('Q')

Default Forecasting Method (Q Forecasting).

7.5.4.5 const PreOptimisationMethod stdair::DEFAULT_PREOPTIMISATION_METHOD('N')

Default Pre-Optimisation Method (NONE).

7.5.4.6 const OptimisationMethod stdair::DEFAULT_OPTIMISATION_METHOD('M')

Default Optimisation Method (Leg Based Monte Carlo).

7.5.4.7 const CensorshipFlagList_T stdair::DEFAULT_CLASS_CENSORSHIPFLAG_LIST

Initial value:

```
std::vector<CensorshipFlag_T>()
```

Default list of censorship flag given the status of availability for BookingClass.

Definition at line 253 of file BasConst.cpp.

7.5.4.8 const Date_T stdair::DEFAULT_DICO_STUDIED_DATE

Default DICO studied date.

Definition at line 426 of file BasConst.cpp.

7.5.4.9 const AirlineCodeList_T stdair::DEFAULT_AIRLINE_CODE_LIST

Default airline code list value (empty vector).

Definition at line 436 of file BasConst.cpp.

7.5.4.10 const ClassList_StringList_T stdair::DEFAULT_CLASS_CODE_LIST

Default class code list value (empty vector).

Definition at line 478 of file BasConst.cpp.

7.5.4.11 const BidPriceVector_T stdair::DEFAULT_BID_PRICE_VECTOR = std::vector<BidPrice_T>()

Default Bid-Price Vector.

Definition at line 484 of file BasConst.cpp.

7.5.4.12 const int stdair::DEFAULT_MAX_DTD = 365

Default value for max day-to-departure (365).

Definition at line 514 of file BasConst.cpp.

Referenced by stdair::SegmentSnapshotTable::initSnapshotBlocks().

7.5.4.13 const DCPList_T stdair::DEFAULT_DCP_LIST = DefaultDCPList::init()

Defaut data collection point list.

Definition at line 517 of file BasConst.cpp.

7.5.4.14 const FRAT5Curve_T stdair::FRAT5_CURVE_A

Initial value:

```
DefaultMap::createFRAT5CurveA()
```

FRAT5 curve A for forecasting and optimisation.

Definition at line 531 of file BasConst.cpp.

7.5.4.15 const FRAT5Curve_T stdair::FRAT5_CURVE_B

Initial value:

```
DefaultMap::createFRAT5CurveB()
```

FRAT5 curve B for forecasting and optimisation.

Definition at line 545 of file BasConst.cpp.

7.5.4.16 const FRAT5Curve_T stdair::FRAT5_CURVE_C

Initial value:

```
DefaultMap::createFRAT5CurveC()
```

FRAT5 curve C for forecasting and optimisation.

Definition at line 559 of file BasConst.cpp.

7.5.4.17 const FRAT5Curve_T stdair::FRAT5_CURVE_D

Initial value:

```
DefaultMap::createFRAT5CurveD()
```

FRAT5 curve D for forecasting and optimisation.

Definition at line 573 of file BasConst.cpp.

7.5.4.18 const FFDisutilityCurve_T stdair::FF_DISUTILITY_CURVE_A

Initial value:

```
DefaultMap::createFFDisutilityCurveA()
```

Disutility curve A for forecasting and optimisation. The lower the value (disutility), the higher the demand sells up to higher fare families.

Definition at line 591 of file BasConst.cpp.

7.5.4.19 const FFDisutilityCurve_T stdair::FF_DISUTILITY_CURVE_B

Initial value:

```
DefaultMap::createFFDisutilityCurveB()
```

Disutility curve B for forecasting and optimisation. The lower the value (disutility), the higher the demand sells up to higher fare families.

Definition at line 609 of file BasConst.cpp.

7.5.4.20 const FFDisutilityCurve_T stdair::FF_DISUTILITY_CURVE_C

Initial value:

```
DefaultMap::createFFDisutilityCurveC()
```

Disutility curve C for forecasting and optimisation. The lower the value (disutility), the higher the demand sells up to higher fare families.

Definition at line 627 of file BasConst.cpp.

7.5.4.21 const FFDisutilityCurve_T stdair::FF_DISUTILITY_CURVE_D

Initial value:

```
DefaultMap::createFFDisutilityCurveD()
```

Disutility curve D for forecasting and optimisation. The lower the value (disutility), the higher the demand sells up to higher fare families.

Definition at line 645 of file BasConst.cpp.

7.5.4.22 const FFDisutilityCurve_T stdair::FF_DISUTILITY_CURVE_E

Initial value:

```
DefaultMap::createFFDisutilityCurveE()
```

Disutility curve E for forecasting and optimisation. The lower the value (disutility), the higher the demand sells up to higher fare families.

Definition at line 663 of file BasConst.cpp.

7.5.4.23 const FFDisutilityCurve_T stdair::FF_DISUTILITY_CURVE_F

Initial value:

```
DefaultMap::createFFDisutilityCurveF()
```

Disutility curve F for forecasting and optimisation. The lower the value (disutility), the higher the demand sells up to higher fare families.

Definition at line 681 of file BasConst.cpp.

7.5.4.24 const DTDFratMap_T stdair::DEFAULT_DTD_FRAT5COEF_MAP

Initial value:

```
DefaultDtdFratMap::init()
```

Default frat5 coef map for demand to come forecaster.

Definition at line 695 of file BasConst.cpp.

7.5.4.25 const DTDProbMap_T stdair::DEFAULT_DTD_PROB_MAP

Initial value:

```
DefaultDtdProbMap::init()
```

Default arrival pattern map.

Definition at line 712 of file BasConst.cpp.

7.5.4.26 const OnDStringList_T stdair::DEFAULT_OND_STRING_LIST

Default list of full keys.

Definition at line 736 of file BasConst.cpp.

7.5.4.27 const std::string stdair::DISPLAY_LEVEL_STRING_ARRAY[51]

Array with the indentation spaces needed for all the BOM hierachical levels.

Definition at line 742 of file BasConst.cpp.

7.5.4.28 const std::string stdair::DISPLAY_LEVEL_STRING_ARRAY[51]

Array with the indentation spaces needed for all the BOM hierachical levels.

Definition at line 742 of file BasConst.cpp.

7.5.4.29 const std::string stdair::DEFAULT_KEY_FLD_DELIMITER

Default delimiter for string display (e.g delimiter for inventory key and flight-date key). Typically set to ';'.

Referenced by stdair::LegDate::describeRoutingKey(), stdair::SegmentCabin::getFullerKey(), stdair::LegCabin::getFullerKey(), and stdair::ParsedKey::toString().

7.5.4.30 const std::string stdair::DEFAULT_KEY_SUB_FLD_DELIMITER

Default sub delimiter for string display (e.g delimiter for flight number and departure date of a flight-date key). Typically set to ','.

Referenced by stdair::BomRetriever::retrieveFullKeyFromSegmentDate(), stdair::SegmentDateKey::toString(), stdair::PosChannelKey::toString(), stdair::ParsedKey::toString(), stdair::FlightDateKey::toString(), stdair::AirportPairKey::toString(), and stdair::AirlineClassListKey::toString().

7.5.4.31 const boost::char_separator<char> stdair::DEFAULT_KEY_TOKEN_DELIMITER

Default token for decoding a full string display.

Referenced by stdair::BomKeyManager::extractKeys().

7.5.4.32 const Distance_T stdair::DEFAULT_DISTANCE_VALUE

Default distance value, in kilometers (0).

7.5.4.33 const ClassCode_T stdair::DEFAULT_CLOSED_CLASS_CODE

Default closed class code ("CC").

7.5.4.34 const NbOfBookings_T stdair::DEFAULT_CLASS_NB_OF_BOOKINGS

Default number of bookings (with counted cancellation) for BookingClass (0).

7.5.4.35 const NbOfBookings_T stdair::DEFAULT_CLASS_TOTAL_NB_OF_BOOKINGS

Default number of bookings (without cancellation) for BookingClass (0).

7.5.4.36 const NbOfBookings_T stdair::DEFAULT_CLASS_UNCONSTRAINED_DEMAND

Default unconstrained demand for BookingClass (0).

7.5.4.37 const NbOfBookings_T stdair::DEFAULT_CLASS_REMAINING_DEMAND_MEAN

Default remaining future demand mean for BookingClass (0).

7.5.4.38 const NbOfBookings_T stdair::DEFAULT_CLASS_REMAINING_DEMAND_STANDARD_DEVIATION

Default remaining futre demand standard deviation for BookingClass (0).

7.5.4.39 const NbOfCancellations_T stdair::DEFAULT_CLASS_NB_OF_CANCELLATIONS

Default number of cancellations for BookingClass (0).

7.5.4.40 const NbOfNoShows_T stdair::DEFAULT_CLASS_NB_OF_NOSHOWS

Default number of no-shows for BookingClass (0).

7.5.4.41 const CabinCapacity_T stdair::DEFAULT_CABIN_CAPACITY

Default cabin capacity for Leg cabins (0.0).

7.5.4.42 const CommittedSpace_T stdair::DEFAULT_COMMITTED_SPACE

Default commited space value for Leg cabins (0.0).

7.5.4.43 const BlockSpace_T stdair::DEFAULT_BLOCK_SPACE

Default commited space value for Leg cabins (0.0).

7.5.4.44 const Availability_T stdair::DEFAULT_NULL_AVAILABILITY

Default null availability (0.0).

7.5.4.45 const Availability_T stdair::DEFAULT_AVAILABILITY

Default availability (9.0).

7.5.4.46 const CensorshipFlag_T stdair::DEFAULT_CLASS_CENSORSHIPFLAG

Default boolean for censorship flag given the status of availability for BookingClass.

7.5.4.47 const CensorshipFlagList_T stdair::DEFAULT_CLASS_CENSORSHIPFLAG_LIST

Default list of censorship flag given the status of availability for BookingClass.

Definition at line 253 of file BasConst.cpp.

7.5.4.48 const BookingLimit_T stdair::DEFAULT_CLASS_BOOKING_LIMIT

Default booking limit value for BookingClass.

7.5.4.49 const AuthorizationLevel_T stdair::DEFAULT_CLASS_AUTHORIZATION_LEVEL

Default authorization level for BookingClass.

7.5.4.50 const AuthorizationLevel_T stdair::DEFAULT_CLASS_MAX_AUTHORIZATION_LEVEL

Default MAX value of authorization level for BookingClass.

7.5.4.51 const AuthorizationLevel_T stdair::DEFAULT_CLASS_MIN_AUTHORIZATION_LEVEL

Default MIN value of authorization level for BookingClass.

7.5.4.52 const OverbookingRate_T stdair::DEFAULT_CLASS_OVERBOOKING_RATE

Default over-booking rate for BookingClass.

7.5.4.53 const Fare_T stdair::DEFAULT_FARE_VALUE

Default fare.

7.5.4.54 const Revenue_T stdair::DEFAULT_REVENUE_VALUE

Default revenue value for BookingClass.

7.5.4.55 const PriceCurrency_T stdair::DEFAULT_CURRENCY

Default currency (euro).

7.5.4.56 const Percentage_T stdair::DEFAULT_LOAD_FACTOR_VALUE

Default load factor value (100%).

7.5.4.57 const DayDuration_T stdair::DEFAULT_DAY_DURATION

Default number of duration days (0).

7.5.4.58 const double stdair::DEFAULT_EPSILON_VALUE

Default epsilon value between customer requirements and a fare rule.

7.5.4.59 const AirportCode_T stdair::AIRPORT_LHR

London Heathrow airport (e.g., "LHR").

7.5.4.60 const AirportCode_T stdair::AIRPORT_SYD

Sydney airport (e.g., "SYD").

7.5.4.61 const CityCode_T stdair::POS_LHR

London city code (e.g., "LHR").

7.5.4.62 const DayDuration_T stdair::NO_ADVANCE_PURCHASE

Advance purchase 0 day.

7.5.4.63 const SaturdayStay_T stdair::SATURDAY_STAY

Default saturdayStay value (true).

7.5.4.64 const SaturdayStay_T stdair::NO_SATURDAY_STAY

Default saturdayStay value (false).

7.5.4.65 const ChangeFees_T stdair::CHANGE_FEES

Default change fees value (true).

7.5.4.66 const ChangeFees_T stdair::NO_CHANGE_FEES

Default change fees value (false).

7.5.4.67 const NonRefundable_T stdair::NON_REFUNDABLE

Default non refundable value (true).

7.5.4.68 const NonRefundable_T stdair::NO_NON_REFUNDABLE

Default refundable value (false).

7.5.4.69 const DayDuration_T stdair::NO_STAY_DURATION

Stay duration 0 day.

7.5.4.70 const CabinCode_T stdair::CABIN_Y

Cabin 'Y'.

7.5.4.71 const AirlineCode_T stdair::AIRLINE_CODE_BA

Airline code "BA".

7.5.4.72 const ClassCode_T stdair::CLASS_CODE_Y

Class code 'Y'.

7.5.4.73 const ClassCode_T stdair::CLASS_CODE_Q

Class code 'Q'.

7.5.4.74 const AirportCode_T stdair::AIRPORT_SIN

Singapour airport (e.g., "SIN").

7.5.4.75 const AirportCode_T stdair::AIRPORT_BKK

Bangkok airport (e.g., "BKK").

7.5.4.76 const CityCode_T stdair::POS_SIN

Singapour city code (e.g., "SIN").

7.5.4.77 const CabinCode_T stdair::CABIN_ECO

Economic cabin (e.g., "Eco").

7.5.4.78 const FrequentFlyer_T stdair::FREQUENT_FLYER_MEMBER

Frequent flyer tier (e.g., "M" meaning member).

7.5.4.79 const Count_T stdair::DEFAULT_PROGRESS_STATUS

Default progress status.

Referenced by stdair::ProgressStatus::reset().

7.5.4.80 const Date_T stdair::DEFAULT_EVENT_OLDEST_DATE

Default reference (oldest) date for the events. No event can occur before that date.

7.5.4.81 const DateTime_T stdair::DEFAULT_EVENT_OLDEST_DATETIME

Default reference (oldest) date-time for the events. No event can occur before that date-time.

Referenced by stdair::EventStruct::describe(), stdair::EventStruct::EventStruct(), and stdair::EventStruct::getEventTime().

7.5.4.82 const Percentage_T stdair::MAXIMUM_PROGRESS_STATUS

Maximum progress status.

Referenced by stdair::ProgressStatus::progress().

7.5.4.83 const std::string stdair::DEFAULT_BOM_ROOT_KEY

Default value for the BOM tree root key (" – ROOT – ").

7.5.4.84 const double stdair::DEFAULT_EPSILON_VALUE

Default epsilon value (1e-4).

7.5.4.85 const CabinCapacity_T stdair::DEFAULT_CABIN_CAPACITY

Default cabin capacity for Leg cabins.

7.5.4.86 const NbOfFlightDates_T stdair::DEFAULT_NB_OF_FLIGHTDATES

Default number of generated flight dates (0).

7.5.4.87 const NbOfBookings_T stdair::DEFAULT_CLASS_NB_OF_BOOKINGS

Default number of bookings for BookingClass.

7.5.4.88 const Distance_T stdair::DEFAULT_DISTANCE_VALUE

Default distance value (kilometers).

7.5.4.89 const unsigned int stdair::DEFAULT_FLIGHT_SPEED

Default flight speed (number of kilometers per hour).

7.5.4.90 const Fare_T stdair::DEFAULT_FARE_VALUE

Default value of Fare.

7.5.4.91 const PriceCurrency_T stdair::DEFAULT_CURRENCY

Default currency (euro).

7.5.4.92 const Revenue_T stdair::DEFAULT_REVENUE_VALUE

Default revenue value.

7.5.4.93 const BookingRatio_T stdair::DEFAULT_OND_BOOKING_RATE

Default booking rate for OnD bookings over overall class bookings.

7.5.4.94 const Count_T stdair::SECONDS_IN_ONE_DAY

Number of seconds in one day (86400).

7.5.4.95 const Count_T stdair::MILLISECONDS_IN_ONE_SECOND

Number of milliseconds in one second (1000).

7.5.4.96 const Date_T stdair::DEFAULT_DATE

Default date for the General (1-Jan-2010).

7.5.4.97 const DateTime_T stdair::DEFAULT_DATETIME

Default date-time (1-Jan-2010).

7.5.4.98 const Duration_T stdair::DEFAULT_EPSILON_DURATION

Default epsilon duration (1 nanosecond).

7.5.4.99 const RandomSeed_T stdair::DEFAULT_RANDOM_SEED

Default random seed (120765987).

Referenced by stdair::BookingClass::generateDemandSamples().

7.5.4.100 const Duration_T stdair::NULL_BOOST_TIME_DURATION

Null time duration (in boost::time_duration unit).

7.5.4.101 const Duration_T stdair::DEFAULT_NULL_DURATION

Default null duration (in boost::time_duration unit).

7.5.4.102 const Fare_T stdair::DEFAULT_CLASS_FARE_VALUE

Default value of Availability.

7.5.4.103 const NbOfAirlines_T stdair::DEFAULT_NBOFAIRLINES

Default number of airlines (0).

7.5.4.104 const unsigned int stdair::DEFAULT_NB_OF_DAYS_IN_A_YEAR

Default number of days in a year (365).

7.5.4.105 const NbOfBookings_T stdair::DEFAULT_CLASS_NB_OF_BOOKINGS

Default number of bookings (0).

7.5.4.106 const ChannelLabel_T stdair::DEFAULT_CHANNEL

Default channel.

7.5.4.107 const OnDStringList_T stdair::DEFAULT_OND_STRING_LIST

Default list of full keys.

Definition at line 736 of file BasConst.cpp.

7.5.4.108 const unsigned int stdair::DEFAULT_NUMBER_OF_SUBDIVISIONS

Higher value per thousand

Referenced by stdair::DictionaryManager::keyToValue(), and stdair::DictionaryManager::valueToKey().

7.5.4.109 const AirlineCode_T stdair::DEFAULT_AIRLINE_CODE

Default airline code value ("XX").

Referenced by stdair::BomRetriever::retrieveDummyLegCabin(), and stdair::BomRetriever::retrieveDummySegmentCabin().

7.5.4.110 const AirlineCode_T stdair::DEFAULT_NULL_AIRLINE_CODE

Default airline code value ("").

7.5.4.111 const AirlineCodeList_T stdair::DEFAULT_AIRLINE_CODE_LIST

Default airline code list value (empty vector).

Definition at line 436 of file BasConst.cpp.

7.5.4.112 const FlightNumber_T stdair::DEFAULT_FLIGHT_NUMBER

Default flight number (9999).

Referenced by stdair::BomRetriever::retrieveDummyLegCabin(), and stdair::BomRetriever::retrieveDummySegmentCabin().

7.5.4.113 const FlightNumber_T stdair::DEFAULT_FLIGHT_NUMBER_FF

Default flight number for fare families (255).

Referenced by stdair::BomRetriever::retrieveDummyLegCabin(), and stdair::BomRetriever::retrieveDummySegmentCabin().

7.5.4.114 const TableID_T stdair::DEFAULT_TABLE_ID

Default data table ID (9999).

7.5.4.115 const Date_T stdair::DEFAULT_DEPARTURE_DATE

Default flight departure date (01/01/1900).

Referenced by stdair::BomRetriever::retrieveDummyLegCabin(), and stdair::BomRetriever::retrieveDummySegmentCabin().

7.5.4.116 const AirportCode_T stdair::DEFAULT_AIRPORT_CODE

Default airport code value ("XXX").

7.5.4.117 const AirportCode_T stdair::DEFAULT_NULL_AIRPORT_CODE

Default airport code value ("")..

7.5.4.118 const AirportCode_T stdair::DEFAULT_ORIGIN

Default Origin ("XXX").

Referenced by stdair::BomRetriever::retrieveDummyLegCabin(), and stdair::BomRetriever::retrieveDummySegmentCabin().

7.5.4.119 const AirportCode_T stdair::DEFAULT_DESTINATION

Default Destination ("XXX").

Referenced by stdair::BomRetriever::retrieveDummySegmentCabin().

7.5.4.120 const CabinCode_T stdair::DEFAULT_CABIN_CODE

Default Cabin Code ("X").

Referenced by stdair::BomRetriever::retrieveDummyLegCabin(), and stdair::BomRetriever::retrieveDummySegmentCabin().

7.5.4.121 const FamilyCode_T stdair::DEFAULT_FARE_FAMILY_CODE

Default Fare Family Code ("EcoSaver").

7.5.4.122 const FamilyCode_T stdair::DEFAULT_NULL_FARE_FAMILY_CODE

Default null fare family Code ("NoFF").

7.5.4.123 const PolicyCode_T stdair::DEFAULT_POLICY_CODE

Default Policy Code ("0").

7.5.4.124 const NestingStructureCode_T stdair::DEFAULT_NESTING_STRUCTURE_CODE

Default Nesting Structure Code ("DEFAULT").

7.5.4.125 const NestingStructureCode_T stdair::DISPLAY_NESTING_STRUCTURE_CODE

Display Nesting Structure Code ("Display Nesting").

7.5.4.126 const NestingStructureCode_T stdair::YIELD_BASED_NESTING_STRUCTURE_CODE

Display Nesting Structure Code ("Yield-Based Nesting").

Referenced by stdair::FacBomManager::resetYieldBasedNestingStructure().

7.5.4.127 const NestingNodeCode_T stdair::DEFAULT_NESTING_NODE_CODE

Default Nesting Node Code ("0").

7.5.4.128 const ClassCode_T stdair::DEFAULT_CLASS_CODE

Default class code value ("X").

7.5.4.129 const ClassCode_T stdair::DEFAULT_NULL_CLASS_CODE

Default null class code value ("").

7.5.4.130 const ClassList_StringList_T stdair::DEFAULT_CLASS_CODE_LIST

Default class code list value (empty vector).

Definition at line 478 of file BasConst.cpp.

7.5.4.131 const BidPrice_T stdair::DEFAULT_BID_PRICE

Default Bid-Price (0.0).

7.5.4.132 const BidPriceVector_T stdair::DEFAULT_BID_PRICE_VECTOR

Default Bid-Price Vector.

Definition at line 484 of file BasConst.cpp.

7.5.4.133 const unsigned short stdair::MAXIMAL_NUMBER_OF_LEGS_IN_FLIGHT

Maximal number of legs linked to a single flight-date (e.g., 7).

Note that the number of derived segments is n∗(n+1)/2 if n is the number of legs.

7.5.4.134 const unsigned short stdair::MAXIMAL_NUMBER_OF_SEGMENTS_IN_OND

Maximal number of segments linked to a single O&D (Origin & Destination)(e.g., 3).

7.5.4.135 const Availability_T stdair::MAXIMAL_AVAILABILITY

Maximal offered capacity in a cabin.

7.5.4.136 const SeatIndex_T stdair::DEFAULT_SEAT_INDEX

Default seat index (for a bucket and/or Bid-Price Vector slot)(e.g., 1).

7.5.4.137 const NbOfSeats_T stdair::DEFAULT_NULL_BOOKING_NUMBER

Default number of bookings.

7.5.4.138 const CapacityAdjustment_T stdair::DEFAULT_NULL_CAPACITY_ADJUSTMENT

Default capacity adjustment of the cabin.

7.5.4.139 const UPR_T stdair::DEFAULT_NULL_UPR

Default unsold Protection (UPR).

7.5.4.140 const std::string stdair::DEFAULT_FARE_FAMILY_VALUE_TYPE

Default value type (within a guillotine block) for fare family.

7.5.4.141 const std::string stdair::DEFAULT_SEGMENT_CABIN_VALUE_TYPE

Default value type (within a guillotine block) for segment-cabin.

7.5.4.142 const int stdair::DEFAULT_MAX_DTD

Default value for max day-to-departure (365).

Definition at line 514 of file BasConst.cpp.

Referenced by stdair::SegmentSnapshotTable::initSnapshotBlocks().

7.5.4.143 const DCPList_T stdair::DEFAULT_DCP_LIST

Defaut data collection point list.

Definition at line 517 of file BasConst.cpp.

7.5.4.144 const DTDFratMap_T stdair::DEFAULT_DTD_FRAT5COEF_MAP

Default frat5 coef map for demand to come forecaster.

Definition at line 695 of file BasConst.cpp.

7.5.4.145 const DTDProbMap_T stdair::DEFAULT_DTD_PROB_MAP

Default arrival pattern map.

Definition at line 712 of file BasConst.cpp.

7.5.4.146 const ForecastingMethod stdair::DEFAULT_FORECASTING_METHOD

Default Forecasting Method (Q Forecasting).

7.5.4.147 const UnconstrainingMethod stdair::DEFAULT_UNCONSTRAINING_METHOD

Default Unconstraining Method (By Expectation-Maximisation).

7.5.4.148 const PreOptimisationMethod stdair::DEFAULT_PREOPTIMISATION_METHOD

Default Pre-Optimisation Method (NONE).

7.5.4.149 const OptimisationMethod stdair::DEFAULT_OPTIMISATION_METHOD

Default Optimisation Method (Leg Based Monte Carlo).

7.5.4.150 const PartnershipTechnique stdair::DEFAULT_PARTNERSHIP_TECHNIQUE

Default Partnership Technique (None).

7.5.4.151 const DatePeriod_T stdair::BOOST_DEFAULT_DATE_PERIOD

Default date period (0-length, i.e., it lasts one day).

7.5.4.152 const std::string stdair::DOW_STR[ ]

Day names (in English).

Definition at line 53 of file BasConst.cpp.

Referenced by stdair::DoWStruct::describe().

7.5.4.153 const DOW_String_T stdair::DEFAULT_DOW_STRING

Default DOW String (e.g., "1111100").

Referenced by stdair::DoWStruct::intersection(), and stdair::DoWStruct::shift().

7.5.4.154 const DateOffset_T stdair::DEFAULT_DATE_OFFSET

Default Date Offset (e.g., 0).

7.5.4.155 const DayDuration_T stdair::DEFAULT_DAY_DURATION

Default Duration in days (e.g., 0).

7.5.4.156 const PartySize_T stdair::DEFAULT_PARTY_SIZE

Default party size in a request (e.g., 1).

7.5.4.157 const DayDuration_T stdair::DEFAULT_STAY_DURATION

Default duration for a stay (e.g., 7 days).

7.5.4.158 const WTP_T stdair::DEFAULT_WTP

Default Willingness-to-Pay (WTP, as expressed as a monetary unit).

7.5.4.159 const CityCode_T stdair::DEFAULT_POS

Default Point-Of-Sale (POS, e.g., "WORLD").

7.5.4.160 const Date_T stdair::DEFAULT_PREFERRED_DEPARTURE_DATE

Default departure date (e.g., 01-Jan-2011).

7.5.4.161 const Duration_T stdair::DEFAULT_PREFERRED_DEPARTURE_TIME

Default preferred departure time (e.g., 08:00).

7.5.4.162 const DateOffset_T stdair::DEFAULT_ADVANCE_PURCHASE

Default advance purchase (e.g., 22 days).

7.5.4.163 const Date_T stdair::DEFAULT_REQUEST_DATE

Default request date (e.g., 10-Jan-2011).

7.5.4.164 const Duration_T stdair::DEFAULT_REQUEST_TIME

Default preferred departure time (e.g., 08:00).

7.5.4.165 const DateTime_T stdair::DEFAULT_REQUEST_DATE_TIME

Default request date-time (e.g., 08:00).

7.5.4.166 const CabinCode_T stdair::DEFAULT_PREFERRED_CABIN

Default preferred cabin (e.g., 'M').

7.5.4.167 const ChannelLabel_T stdair::DEFAULT_CHANNEL

Default channel (e.g., direct on-line).

7.5.4.168 const ChannelLabel_T stdair::CHANNEL_DN

DN channel (e.g., direct on-line).

7.5.4.169 const ChannelLabel_T stdair::CHANNEL_IN

IN channel (e.g., indirect on-line).

7.5.4.170 const TripType_T stdair::TRIP_TYPE_ONE_WAY

Trip type one-way (e.g., "OW").

Referenced by stdair::BookingRequestStruct::display().

7.5.4.171 const TripType_T stdair::TRIP_TYPE_ROUND_TRIP

Trip type round-trip (e.g., "RT").

Referenced by stdair::YieldFeatures::isTripTypeValid(), and stdair::FareFeatures::isTripTypeValid().

7.5.4.172 const TripType_T stdair::TRIP_TYPE_INBOUND

Trip type inbound (e.g., "RI").

Referenced by stdair::YieldFeatures::isTripTypeValid(), and stdair::FareFeatures::isTripTypeValid().

7.5.4.173 const TripType_T stdair::TRIP_TYPE_OUTBOUND

Trip type outbound (e.g., "RO").

Referenced by stdair::YieldFeatures::isTripTypeValid(), and stdair::FareFeatures::isTripTypeValid().

7.5.4.174 const FrequentFlyer_T stdair::DEFAULT_FF_TIER

Default frequent flyer tier (e.g., non member).

7.5.4.175 const PriceValue_T stdair::DEFAULT_VALUE_OF_TIME

Default value of time (expressed as a monetary unit per hour).

7.5.4.176 const IntDuration_T stdair::HOUR_CONVERTED_IN_SECONDS

Number of second in one hour

7.5.4.177 const FRAT5Curve_T stdair::FRAT5_CURVE_A

FRAT5 curve A for forecasting and optimisation.

Definition at line 531 of file BasConst.cpp.

7.5.4.178 const FRAT5Curve_T stdair::FRAT5_CURVE_B

FRAT5 curve B for forecasting and optimisation.

Definition at line 545 of file BasConst.cpp.

7.5.4.179 const FRAT5Curve_T stdair::FRAT5_CURVE_C

FRAT5 curve C for forecasting and optimisation.

Definition at line 559 of file BasConst.cpp.

7.5.4.180 const FRAT5Curve_T stdair::FRAT5_CURVE_D

FRAT5 curve D for forecasting and optimisation.

Definition at line 573 of file BasConst.cpp.

7.5.4.181 const FFDisutilityCurve_T stdair::FF_DISUTILITY_CURVE_A

Disutility curve A for forecasting and optimisation. The lower the value (disutility), the higher the demand sells up to higher fare families.

Definition at line 591 of file BasConst.cpp.

7.5.4.182 const FFDisutilityCurve_T stdair::FF_DISUTILITY_CURVE_B

Disutility curve B for forecasting and optimisation. The lower the value (disutility), the higher the demand sells up to higher fare families.

Definition at line 609 of file BasConst.cpp.

7.5.4.183 const FFDisutilityCurve_T stdair::FF_DISUTILITY_CURVE_C

Disutility curve C for forecasting and optimisation. The lower the value (disutility), the higher the demand sells up to higher fare families.

Definition at line 627 of file BasConst.cpp.

7.5.4.184 const FFDisutilityCurve_T stdair::FF_DISUTILITY_CURVE_D

Disutility curve D for forecasting and optimisation. The lower the value (disutility), the higher the demand sells up to higher fare families.

Definition at line 645 of file BasConst.cpp.

7.5.4.185 const FFDisutilityCurve_T stdair::FF_DISUTILITY_CURVE_E

Disutility curve E for forecasting and optimisation. The lower the value (disutility), the higher the demand sells up to higher fare families.

Definition at line 663 of file BasConst.cpp.

7.5.4.186 const FFDisutilityCurve_T stdair::FF_DISUTILITY_CURVE_F

Disutility curve F for forecasting and optimisation. The lower the value (disutility), the higher the demand sells up to higher fare families.

Definition at line 681 of file BasConst.cpp.

7.5.4.187 const Distance_T stdair::DEFAULT_DISTANCE_VALUE

Default distance value (kilometers).

7.5.4.188 const Duration_T stdair::DEFAULT_MINIMAL_CONNECTION_TIME

Default Minimal connection time.

7.5.4.189 const Duration_T stdair::DEFAULT_MAXIMAL_CONNECTION_TIME

Default maximal connection time.

7.5.4.190 const Duration_T stdair::NULL_BOOST_TIME_DURATION

Null time duration (in boost::time_duration unit).

7.5.4.191 const FlightPathCode_T stdair::DEFAULT_FLIGHTPATH_CODE

Default flightPathCode value (”).

7.5.4.192 const Availability_T stdair::DEFAULT_CLASS_AVAILABILITY

Default value of Availability.

7.5.4.193 const AvailabilityStatus_T stdair::DEFAULT_AVAILABILITY_STATUS

Default availability status for a travel solution.

7.5.4.194 const unsigned short stdair::DEFAULT_NUMBER_OF_REQUIRED_SEATS

Default nember of required seats by the demand.

7.5.4.195 const MatchingIndicator_T stdair::DEFAULT_MATCHING_INDICATOR

Default Matching Indicator value between customer requirements and a fare rule.

7.5.4.196 const Revenue_T stdair::DEFAULT_REVENUE_VALUE

Default revenue value.

7.5.4.197 const AirlineCode_T stdair::DEFAULT_DICO_STUDIED_AIRLINE

Default DICO studied airline.

7.5.4.198 const Date_T stdair::DEFAULT_DICO_STUDIED_DATE

Default DICO studied date.

Definition at line 426 of file BasConst.cpp.

7.5.4.199 const Yield_T stdair::DEFAULT_YIELD_VALUE

Default yield value.

7.5.4.200 const Yield_T stdair::DEFAULT_YIELD_MAX_VALUE

Default yield max value.

7.6 stdair::LOG Namespace Reference

Enumerations

- enum EN_LogLevel {
  CRITICAL = 0, ERROR, NOTIFICATION, WARNING,
  DEBUG, VERBOSE, LAST_VALUE }

Variables

- static const std::string _logLevels [LAST_VALUE]

7.6.1 Detailed Description

Level of logs.

7.6.2 Enumeration Type Documentation

7.6.2.1 enum stdair::LOG::EN_LogLevel

Enumerator:

CRITICAL
ERROR
NOTIFICATION
WARNING
DEBUG
VERBOSE
LAST_VALUE

Definition at line 18 of file stdair_log.hpp.

7.6.3 Variable Documentation

7.6.3.1 const std::string stdair::LOG::_logLevels[LAST_VALUE] [static]

Initial value:

```
{"C", "E", "N", "W", "D", "V"}
```

Definition at line 28 of file stdair_log.hpp.

Referenced by stdair::Logger::log(), stdair::BasLogParams::toShortString(), and stdair::BasLogParams::toString().

7.7 stdair_test Namespace Reference

Classes

- struct BookingClass
- struct Cabin

7.7.1 Detailed Description

Namespace gathering classes and structures for test purposes

7.8 swift Namespace Reference

The wrapper namespace.

Classes

- class SKeymap

  The readline keymap wrapper.

- class SReadline

  The readline library wrapper.

7.8.1 Detailed Description

The wrapper namespace.

The namespace is also used for other library elements.

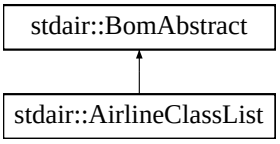
8 StdAir Class Documentation

8.1 stdair::AirlineClassList Class Reference

Class representing the actual attributes for a segment-features.

`#include <stdair/bom/AirlineClassList.hpp>`

Inheritance diagram for stdair::AirlineClassList::



Public Types

- typedef AirlineClassListKey Key_T

Public Member Functions

- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const AirlineCodeList_T & getAirlineCodeList () const
- const ClassList_StringList_T & getClassCodeList () const
- const HolderMap_T & getHolderMap () const
- const stdair::Yield_T & getYield () const
- const stdair::Fare_T & getFare () const
- void setYield (const Yield_T &iYield)
- void setFare (const Fare_T &iFare)
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- template<class Archive> void serialize (Archive &ar, const unsigned int iFileVersion)

Protected Member Functions

- AirlineClassList (const Key_T &)
- virtual ~AirlineClassList ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent
- HolderMap_T _holderMap
- Yield_T _yield
- Fare_T _fare

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager
- class boost::serialization::access

8.1.1 Detailed Description

Class representing the actual attributes for a segment-features.

Definition at line 27 of file AirlineClassList.hpp.

8.1.2 Member Typedef Documentation

8.1.2.1 typedef AirlineClassListKey stdair::AirlineClassList::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 38 of file AirlineClassList.hpp.

8.1.3 Constructor & Destructor Documentation

8.1.3.1 stdair::AirlineClassList::AirlineClassList (const Key_T &) [protected]

Main constructor.

Definition at line 34 of file AirlineClassList.cpp.

8.1.3.2 stdair::AirlineClassList::~AirlineClassList () [protected, virtual]

Destructor.

Definition at line 39 of file AirlineClassList.cpp.

8.1.4 Member Function Documentation

8.1.4.1 const Key_T& stdair::AirlineClassList::getKey () const [inline]

Get the airline class list key.

Definition at line 44 of file AirlineClassList.hpp.

References _key.

8.1.4.2 BomAbstract* const stdair::AirlineClassList::getParent () const [inline]

Get the parent object.

Definition at line 49 of file AirlineClassList.hpp.

References _parent.

8.1.4.3 const AirlineCodeList_T& stdair::AirlineClassList::getAirlineCodeList () const [inline]

Get the airline code list (part of the primary key).

Definition at line 54 of file AirlineClassList.hpp.

References _key, and stdair::AirlineClassListKey::getAirlineCodeList().

8.1.4.4 const ClassList_StringList_T& stdair::AirlineClassList::getClassCodeList () const [inline]

Get the class code list (part of the primary key).

Definition at line 59 of file AirlineClassList.hpp.

References _key, and stdair::AirlineClassListKey::getClassCodeList().

8.1.4.5 const HolderMap_T& stdair::AirlineClassList::getHolderMap () const [inline]

Get the map of children holders.

Definition at line 64 of file AirlineClassList.hpp.

References _holderMap.

8.1.4.6 const stdair::Yield_T& stdair::AirlineClassList::getYield () const [inline]

Get the yield.

Definition at line 69 of file AirlineClassList.hpp.

References _yield.

8.1.4.7 const stdair::Fare_T& stdair::AirlineClassList::getFare () const [inline]

Get the fare.

Definition at line 74 of file AirlineClassList.hpp.

References _fare.

8.1.4.8 void stdair::AirlineClassList::setYield (const Yield_T & *iYield*) [inline]

Definition at line 80 of file AirlineClassList.hpp.

References _yield.

8.1.4.9 void stdair::AirlineClassList::setFare (const Fare_T & *iFare*) [inline]

Definition at line 84 of file AirlineClassList.hpp.

References _fare.

8.1.4.10 void stdair::AirlineClassList::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 95 of file AirlineClassList.hpp.

References toString().

8.1.4.11 void stdair::AirlineClassList::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 104 of file AirlineClassList.hpp.

8.1.4.12 std::string stdair::AirlineClassList::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 43 of file AirlineClassList.cpp.

References _fare, _yield, and describeKey().

Referenced by toStream().

8.1.4.13 const std::string stdair::AirlineClassList::describeKey () const [inline]

Get a string describing the key.

Definition at line 115 of file AirlineClassList.hpp.

References _key, and stdair::AirlineClassListKey::toString().

Referenced by toString().

8.1.4.14 template<class Archive> void stdair::AirlineClassList::serialize (Archive & *ar*, const unsigned int *iFileVersion*)

Serialisation.

Definition at line 65 of file AirlineClassList.cpp.

References _fare, _key, and _yield.

8.1.5 Friends And Related Function Documentation

8.1.5.1 friend class FacBom [friend]

Definition at line 28 of file AirlineClassList.hpp.

8.1.5.2 friend class FacCloneBom [friend]

Definition at line 29 of file AirlineClassList.hpp.

8.1.5.3 friend class FacBomManager [friend]

Definition at line 30 of file AirlineClassList.hpp.

8.1.5.4 friend class boost::serialization::access [friend]

Definition at line 31 of file AirlineClassList.hpp.

8.1.6 Member Data Documentation

8.1.6.1 Key_T stdair::AirlineClassList::_key [protected]

Primary key (flight number and departure date).

Definition at line 165 of file AirlineClassList.hpp.

Referenced by describeKey(), getAirlineCodeList(), getClassCodeList(), getKey(), and serialize().

8.1.6.2 BomAbstract* stdair::AirlineClassList::_parent [protected]

Pointer on the parent class (Inventory).

Definition at line 170 of file AirlineClassList.hpp.

Referenced by getParent().

8.1.6.3 HolderMap_T stdair::AirlineClassList::_holderMap [protected]

Map holding the children (SegmentDate and LegDate objects).

Definition at line 175 of file AirlineClassList.hpp.

Referenced by getHolderMap().

8.1.6.4 Yield_T stdair::AirlineClassList::_yield [protected]

Definition at line 180 of file AirlineClassList.hpp.

Referenced by getYield(), serialize(), setYield(), and toString().

8.1.6.5 Fare_T stdair::AirlineClassList::_fare [protected]

Definition at line 185 of file AirlineClassList.hpp.

Referenced by getFare(), serialize(), setFare(), and toString().

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

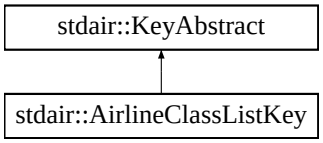
- stdair/bom/AirlineClassList.hpp
- stdair/bom/AirlineClassList.cpp

8.2 stdair::AirlineClassListKey Struct Reference

Key of airport-pair.

`#include <stdair/bom/AirlineClassListKey.hpp>`

Inheritance diagram for stdair::AirlineClassListKey::



Public Member Functions

- AirlineClassListKey (const AirlineCodeList_T &, const ClassList_StringList_T &)
- AirlineClassListKey (const AirlineClassListKey &)
- ~AirlineClassListKey ()
- const AirlineCodeList_T & getAirlineCodeList () const
- const ClassList_StringList_T & getClassCodeList () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)

- const std::string toString () const
- template<class Archive> void serialize (Archive &ar, const unsigned int iFileVersion)

Friends

- class boost::serialization::access

8.2.1 Detailed Description

Key of airport-pair.

Definition at line 25 of file AirlineClassListKey.hpp.

8.2.2 Constructor & Destructor Documentation

8.2.2.1 stdair::AirlineClassListKey::AirlineClassListKey (const AirlineCodeList_T &, const ClassList_StringList_T &)

Constructor.

Definition at line 24 of file AirlineClassListKey.cpp.

8.2.2.2 stdair::AirlineClassListKey::AirlineClassListKey (const AirlineClassListKey &)

Copy constructor.

Definition at line 30 of file AirlineClassListKey.cpp.

8.2.2.3 stdair::AirlineClassListKey::~AirlineClassListKey ()

Destructor.

Definition at line 36 of file AirlineClassListKey.cpp.

8.2.3 Member Function Documentation

8.2.3.1 const AirlineCodeList_T& stdair::AirlineClassListKey::getAirlineCodeList () const [inline]

Get the airline code list.

Definition at line 56 of file AirlineClassListKey.hpp.

Referenced by stdair::AirlineClassList::getAirlineCodeList().

8.2.3.2 const ClassList_StringList_T& stdair::AirlineClassListKey::getClassCodeList () const [inline]

Get the class code list.

Definition at line 61 of file AirlineClassListKey.hpp.

Referenced by stdair::AirlineClassList::getClassCodeList().

8.2.3.3 void stdair::AirlineClassListKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 40 of file AirlineClassListKey.cpp.

References toString().

8.2.3.4 void stdair::AirlineClassListKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 45 of file AirlineClassListKey.cpp.

8.2.3.5 const std::string stdair::AirlineClassListKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from stdair::KeyAbstract.

Definition at line 49 of file AirlineClassListKey.cpp.

References stdair::DEFAULT_KEY_SUB_FLD_DELIMITER.

Referenced by stdair::AirlineClassList::describeKey(), and toStream().

8.2.3.6 template<class Archive> void stdair::AirlineClassListKey::serialize (Archive & *ar*, const unsigned int *iFileVersion*)

Serialisation.

Definition at line 86 of file AirlineClassListKey.cpp.

8.2.4 Friends And Related Function Documentation

8.2.4.1 friend class boost::serialization::access [friend]

Definition at line 26 of file AirlineClassListKey.hpp.

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

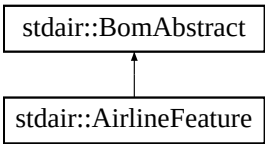
- stdair/bom/AirlineClassListKey.hpp
- stdair/bom/AirlineClassListKey.cpp

8.3 stdair::AirlineFeature Class Reference

Class representing various configuration parameters (e.g., revenue management methods such EMSRb or Monte-Carlo) for a given airline for the simulation.

`#include <stdair/bom/AirlineFeature.hpp>`

Inheritance diagram for stdair::AirlineFeature::



Public Types

- typedef AirlineFeatureKey Key_T

Public Member Functions

- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const HolderMap_T & getHolderMap () const
- ForecastingMethod::EN_ForecastingMethod getForecastingMethod () const
- UnconstrainingMethod::EN_UnconstrainingMethod getUnconstrainingMethod () const
- PartnershipTechnique::EN_PartnershipTechnique getPartnershipTechnique () const
- PreOptimisationMethod::EN_PreOptimisationMethod getPreOptimisationMethod () const
- OptimisationMethod::EN_OptimisationMethod getOptimisationMethod () const
- void init (const ForecastingMethod &, const UnconstrainingMethod &, const PreOptimisationMethod &, const OptimisationMethod &, const HistoricalDataLimit_T &, const ControlMode_T &, const PartnershipTechnique &)
- void setForecastingMethod (const ForecastingMethod &iForecastingMethod)
- void setUnconstrainingMethod (const UnconstrainingMethod &iUnconstrainingMethod)
- void setPartnershipTechnique (const PartnershipTechnique &iPartnershipTechnique)
- void setPreOptimisationMethod (const PreOptimisationMethod &iPreOptimisationMethod)
- void setOptimisationMethod (const OptimisationMethod &iOptimisationMethod)

Protected Member Functions

- AirlineFeature (const Key_T &)
- virtual ~AirlineFeature ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent
- HolderMap_T _holderMap
- ForecastingMethod _forecastingMethod
- HistoricalDataLimit_T _historicalDataLimit
- ControlMode_T _controlMode
- UnconstrainingMethod _unconstrainingMethod
- PreOptimisationMethod _preOptimisationMethod
- OptimisationMethod _optimisationMethod
- PartnershipTechnique _partnershipTechnique

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager

8.3.1 Detailed Description

Class representing various configuration parameters (e.g., revenue management methods such EMSRb or Monte-Carlo) for a given airline for the simulation.

Definition at line 25 of file AirlineFeature.hpp.

8.3.2 Member Typedef Documentation

8.3.2.1 typedef AirlineFeatureKey stdair::AirlineFeature::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 35 of file AirlineFeature.hpp.

8.3.3 Constructor & Destructor Documentation

8.3.3.1 stdair::AirlineFeature::AirlineFeature (const Key_T &) [protected]

Main constructor.

Definition at line 14 of file AirlineFeature.cpp.

8.3.3.2 stdair::AirlineFeature::~AirlineFeature () [protected, virtual]

Destructor.

Definition at line 34 of file AirlineFeature.cpp.

8.3.4 Member Function Documentation

8.3.4.1 void stdair::AirlineFeature::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 44 of file AirlineFeature.hpp.

References toString().

8.3.4.2 void stdair::AirlineFeature::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 53 of file AirlineFeature.hpp.

8.3.4.3 std::string stdair::AirlineFeature::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 55 of file AirlineFeature.cpp.

References _forecastingMethod, _historicalDataLimit, _optimisationMethod, _partnershipTechnique, _preOptimisationMethod, _unconstrainingMethod, and describeKey().

Referenced by toStream().

8.3.4.4 const std::string stdair::AirlineFeature::describeKey () const [inline]

Get a string describing the key.

Definition at line 64 of file AirlineFeature.hpp.

References _key, and stdair::AirlineFeatureKey::toString().

Referenced by toString().

8.3.4.5 const Key_T& stdair::AirlineFeature::getKey () const [inline]

Get the airline feature primary key (airline code).

Definition at line 73 of file AirlineFeature.hpp.

References _key.

8.3.4.6 BomAbstract* const stdair::AirlineFeature::getParent () const [inline]

Get a reference on the parent object instance.

Definition at line 80 of file AirlineFeature.hpp.

References _parent.

8.3.4.7 const HolderMap_T& stdair::AirlineFeature::getHolderMap () const [inline]

Get a reference on the children holder.

Definition at line 87 of file AirlineFeature.hpp.

References _holderMap.

8.3.4.8 ForecastingMethod::EN_ForecastingMethod stdair::AirlineFeature::getForecastingMethod () const [inline]

Get the forecasting method.

Definition at line 94 of file AirlineFeature.hpp.

References _forecastingMethod, and stdair::ForecastingMethod::getMethod().

Referenced by stdair::Inventory::getForecastingMethod().

8.3.4.9 UnconstrainingMethod::EN_UnconstrainingMethod stdair::AirlineFeature::getUnconstrainingMethod () const [inline]

Get the unconstraining method.

Definition at line 101 of file AirlineFeature.hpp.

References _unconstrainingMethod, and stdair::UnconstrainingMethod::getMethod().

Referenced by stdair::Inventory::getUnconstrainingMethod().

8.3.4.10 PartnershipTechnique::EN_PartnershipTechnique stdair::AirlineFeature::getPartnershipTechnique () const [inline]

Get the partnership technique.

Definition at line 108 of file AirlineFeature.hpp.

References _partnershipTechnique, and stdair::PartnershipTechnique::getTechnique().

Referenced by stdair::Inventory::getPartnershipTechnique().

8.3.4.11 PreOptimisationMethod::EN_PreOptimisationMethod stdair::AirlineFeature::getPreOptimisationMethod () const [inline]

Get the pre-optimisation method.

Definition at line 115 of file AirlineFeature.hpp.

References _preOptimisationMethod, and stdair::PreOptimisationMethod::getMethod().

Referenced by stdair::Inventory::getPreOptimisationMethod().

8.3.4.12 OptimisationMethod::EN_OptimisationMethod stdair::AirlineFeature::getOptimisationMethod () const [inline]

Get the optimisation method.

Definition at line 122 of file AirlineFeature.hpp.

References _optimisationMethod, and stdair::OptimisationMethod::getMethod().

Referenced by stdair::Inventory::getOptimisationMethod().

8.3.4.13 void stdair::AirlineFeature::init (const ForecastingMethod &, const UnconstrainingMethod &, const PreOptimisationMethod &, const OptimisationMethod &, const HistoricalDataLimit_T &, const ControlMode_T &, const PartnershipTechnique &)

Initialization method.

Parameters:

const ForecastingMethod& Forecasting method.

const UnconstrainingMethod& Unconstraining method.

const PreOptimisationMethod& Pre-optimisation method.

const OptimisationMethodGet& Optimisation method.

const HistoricalDataLimit_T& Historical Data Limit

const ControlMode_T& Control Mode

const PartnershipTechnique& Partnership method.

Definition at line 38 of file AirlineFeature.cpp.

References _controlMode, _forecastingMethod, _historicalDataLimit, _optimisationMethod, _partnershipTechnique, _preOptimisationMethod, and _unconstrainingMethod.

8.3.4.14 void stdair::AirlineFeature::setForecastingMethod (const ForecastingMethod & *iForecastingMethod*) [inline]

Set the forecasting method.

Definition at line 150 of file AirlineFeature.hpp.

References _forecastingMethod.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

8.3.4.15 void stdair::AirlineFeature::setUnconstrainingMethod (const UnconstrainingMethod & *iUnconstrainingMethod*) [inline]

Set the unconstraining method.

Definition at line 157 of file AirlineFeature.hpp.

References _unconstrainingMethod.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

8.3.4.16 void stdair::AirlineFeature::setPartnershipTechnique (const PartnershipTechnique & *iPartnershipTechnique*) [inline]

Set the partnership technique.

Definition at line 164 of file AirlineFeature.hpp.

References _partnershipTechnique.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

8.3.4.17 void stdair::AirlineFeature::setPreOptimisationMethod (const PreOptimisationMethod & *iPreOptimisationMethod*) [inline]

Set the pre-optimisation method.

Definition at line 171 of file AirlineFeature.hpp.

References _preOptimisationMethod.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

8.3.4.18 void stdair::AirlineFeature::setOptimisationMethod (const OptimisationMethod & *iOptimisationMethod*) [inline]

Set the optimisation method.

Definition at line 178 of file AirlineFeature.hpp.

References _optimisationMethod.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

8.3.5 Friends And Related Function Documentation

8.3.5.1 friend class FacBom [friend]

Definition at line 26 of file AirlineFeature.hpp.

8.3.5.2 friend class FacCloneBom [friend]

Definition at line 27 of file AirlineFeature.hpp.

8.3.5.3 friend class FacBomManager [friend]

Definition at line 28 of file AirlineFeature.hpp.

8.3.6 Member Data Documentation

8.3.6.1 Key_T stdair::AirlineFeature::_key [protected]

Primary key (date period).

Definition at line 209 of file AirlineFeature.hpp.

Referenced by describeKey(), and getKey().

8.3.6.2 BomAbstract* stdair::AirlineFeature::_parent [protected]

Pointer on the parent class.

Definition at line 214 of file AirlineFeature.hpp.

Referenced by getParent().

8.3.6.3 HolderMap_T stdair::AirlineFeature::_holderMap [protected]

Map holding the children.

Definition at line 219 of file AirlineFeature.hpp.

Referenced by getHolderMap().

8.3.6.4 ForecastingMethod stdair::AirlineFeature::_forecastingMethod [protected]

The type of forecaster.

Definition at line 224 of file AirlineFeature.hpp.

Referenced by getForecastingMethod(), init(), setForecastingMethod(), and toString().

8.3.6.5 HistoricalDataLimit_T stdair::AirlineFeature::_historicalDataLimit [protected]

The size of the moving average window.

Definition at line 229 of file AirlineFeature.hpp.

Referenced by init(), and toString().

8.3.6.6 ControlMode_T stdair::AirlineFeature::_controlMode [protected]

The type of inventory control.

Definition at line 234 of file AirlineFeature.hpp.

Referenced by init().

8.3.6.7 UnconstrainingMethod stdair::AirlineFeature::_unconstrainingMethod [protected]

The type of unconstraining method.

Definition at line 239 of file AirlineFeature.hpp.

Referenced by getUnconstrainingMethod(), init(), setUnconstrainingMethod(), and toString().

8.3.6.8 PreOptimisationMethod stdair::AirlineFeature::_preOptimisationMethod [protected]

The type of pre-optimisation method.

Definition at line 244 of file AirlineFeature.hpp.

Referenced by getPreOptimisationMethod(), init(), setPreOptimisationMethod(), and toString().

8.3.6.9 OptimisationMethod stdair::AirlineFeature::_optimisationMethod [protected]

The type of optimisation method.

Definition at line 249 of file AirlineFeature.hpp.

Referenced by getOptimisationMethod(), init(), setOptimisationMethod(), and toString().

8.3.6.10 PartnershipTechnique stdair::AirlineFeature::_partnershipTechnique [protected]

The type of partnership technique.

Definition at line 254 of file AirlineFeature.hpp.

Referenced by getPartnershipTechnique(), init(), setPartnershipTechnique(), and toString().

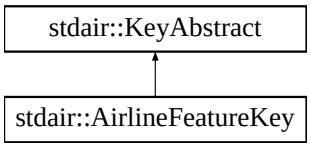
The documentation for this class was generated from the following files:

- stdair/bom/AirlineFeature.hpp
- stdair/bom/AirlineFeature.cpp

8.4 stdair::AirlineFeatureKey Struct Reference

`#include <stdair/bom/AirlineFeatureKey.hpp>`

Inheritance diagram for stdair::AirlineFeatureKey::



Public Member Functions

- AirlineFeatureKey (const AirlineCode_T &iAirlineCode)
- ~AirlineFeatureKey ()
- const AirlineCode_T & getAirlineCode () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const

8.4.1 Detailed Description

Key of AirlineFeature.

Definition at line 15 of file AirlineFeatureKey.hpp.

8.4.2 Constructor & Destructor Documentation

8.4.2.1 stdair::AirlineFeatureKey::AirlineFeatureKey (const AirlineCode_T & *iAirlineCode*)

Constructor.

Definition at line 12 of file AirlineFeatureKey.cpp.

8.4.2.2 stdair::AirlineFeatureKey::~AirlineFeatureKey ()

Destructor.

Definition at line 17 of file AirlineFeatureKey.cpp.

8.4.3 Member Function Documentation

8.4.3.1 const AirlineCode_T& stdair::AirlineFeatureKey::getAirlineCode () const [inline]

Get the airline code.

Definition at line 27 of file AirlineFeatureKey.hpp.

8.4.3.2 void stdair::AirlineFeatureKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 21 of file AirlineFeatureKey.cpp.

References toString().

8.4.3.3 void stdair::AirlineFeatureKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 26 of file AirlineFeatureKey.cpp.

8.4.3.4 const std::string stdair::AirlineFeatureKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from stdair::KeyAbstract.

Definition at line 30 of file AirlineFeatureKey.cpp.

Referenced by stdair::AirlineFeature::describeKey(), and toStream().

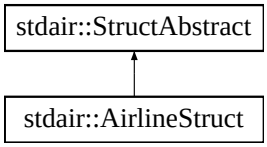
The documentation for this struct was generated from the following files:

- stdair/bom/AirlineFeatureKey.hpp
- stdair/bom/AirlineFeatureKey.cpp

8.5 stdair::AirlineStruct Struct Reference

`#include <stdair/bom/AirlineStruct.hpp>`

Inheritance diagram for stdair::AirlineStruct::



Public Member Functions

- const AirlineCode_T & getAirlineCode () const
- const std::string & getAirlineName () const
- void setAirlineCode (const AirlineCode_T &iAirlineCode)
- void setAirlineName (const std::string &iAirlineName)
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string describe () const
- AirlineStruct (const AirlineCode_T &, const std::string &iAirlineName)
- AirlineStruct ()
- AirlineStruct (const AirlineStruct &)
- ~AirlineStruct ()

8.5.1 Detailed Description

Structure holding parameters describing an airline.

Definition at line 18 of file AirlineStruct.hpp.

8.5.2 Constructor & Destructor Documentation

8.5.2.1 stdair::AirlineStruct::AirlineStruct (const AirlineCode_T &, const std::string & *iAirlineName*)

Main constructor.

Definition at line 24 of file AirlineStruct.cpp.

8.5.2.2 stdair::AirlineStruct::AirlineStruct ()

Default constructor.

Definition at line 15 of file AirlineStruct.cpp.

8.5.2.3 stdair::AirlineStruct::AirlineStruct (const AirlineStruct &)

Default copy constructor.

Definition at line 19 of file AirlineStruct.cpp.

8.5.2.4 stdair::AirlineStruct::~AirlineStruct ()

Destructor.

Definition at line 30 of file AirlineStruct.cpp.

8.5.3 Member Function Documentation

8.5.3.1 const AirlineCode_T& stdair::AirlineStruct::getAirlineCode () const [inline]

Get the airline code.

Definition at line 22 of file AirlineStruct.hpp.

Referenced by soci::type_conversion< stdair::AirlineStruct >::to_base(), and stdair::DBManagerForAirlines::updateAirlineInDB().

8.5.3.2 const std::string& stdair::AirlineStruct::getAirlineName () const [inline]

Get the airline name.

Definition at line 27 of file AirlineStruct.hpp.

Referenced by soci::type_conversion< stdair::AirlineStruct >::to_base(), and stdair::DBManagerForAirlines::updateAirlineInDB().

8.5.3.3 void stdair::AirlineStruct::setAirlineCode (const AirlineCode_T & *iAirlineCode*) [inline]

Set the airline code.

Definition at line 33 of file AirlineStruct.hpp.

Referenced by soci::type_conversion< stdair::AirlineStruct >::from_base().

8.5.3.4 void stdair::AirlineStruct::setAirlineName (const std::string & *iAirlineName*) [inline]

Set the airline name.

Definition at line 38 of file AirlineStruct.hpp.

Referenced by soci::type_conversion< stdair::AirlineStruct >::from_base().

8.5.3.5 void stdair::AirlineStruct::toStream (std::ostream & *ioOut*) const

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::StructAbstract.

Definition at line 34 of file AirlineStruct.cpp.

References describe().

8.5.3.6 void stdair::AirlineStruct::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::StructAbstract.

Definition at line 39 of file AirlineStruct.cpp.

8.5.3.7 const std::string stdair::AirlineStruct::describe () const [virtual]

Display of the structure.

Implements stdair::StructAbstract.

Definition at line 43 of file AirlineStruct.cpp.

Referenced by toStream().

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

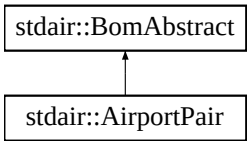
- stdair/bom/AirlineStruct.hpp
- stdair/bom/AirlineStruct.cpp

8.6 stdair::AirportPair Class Reference

Class representing the actual attributes for an airport-pair.

`#include <stdair/bom/AirportPair.hpp>`

Inheritance diagram for stdair::AirportPair::



Public Types

- typedef AirportPairKey Key_T

Public Member Functions

- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- const Key_T & getKey () const
- const AirportCode_T & getBoardingPoint () const
- const AirportCode_T & getOffPoint () const
- BomAbstract *const getParent () const
- const HolderMap_T & getHolderMap () const

Protected Member Functions

- AirportPair (const Key_T &)
- virtual ~AirportPair ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent
- HolderMap_T _holderMap

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager

8.6.1 Detailed Description

Class representing the actual attributes for an airport-pair.

Definition at line 18 of file AirportPair.hpp.

8.6.2 Member Typedef Documentation

8.6.2.1 typedef AirportPairKey stdair::AirportPair::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 28 of file AirportPair.hpp.

8.6.3 Constructor & Destructor Documentation

8.6.3.1 stdair::AirportPair::AirportPair (const Key_T &) [protected]

Main constructor.

Definition at line 27 of file AirportPair.cpp.

8.6.3.2 stdair::AirportPair::~AirportPair () [protected, virtual]

Destructor.

Definition at line 32 of file AirportPair.cpp.

8.6.4 Member Function Documentation

8.6.4.1 void stdair::AirportPair::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 37 of file AirportPair.hpp.

References toString().

8.6.4.2 void stdair::AirportPair::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 46 of file AirportPair.hpp.

8.6.4.3 std::string stdair::AirportPair::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 36 of file AirportPair.cpp.

References describeKey().

Referenced by toStream().

8.6.4.4 const std::string stdair::AirportPair::describeKey () const [inline]

Get a string describing the key.

Definition at line 57 of file AirportPair.hpp.

References _key, and stdair::AirportPairKey::toString().

Referenced by toString().

8.6.4.5 const Key_T& stdair::AirportPair::getKey () const [inline]

Get the primary key (origin airport, destination airport).

Definition at line 66 of file AirportPair.hpp.

References _key.

8.6.4.6 const AirportCode_T& stdair::AirportPair::getBoardingPoint () const [inline]

Get the origin airport.

Definition at line 73 of file AirportPair.hpp.

References _key, and stdair::AirportPairKey::getBoardingPoint().

8.6.4.7 const AirportCode_T& stdair::AirportPair::getOffPoint () const [inline]

Get the destination airport.

Definition at line 80 of file AirportPair.hpp.

References _key, and stdair::AirportPairKey::getOffPoint().

8.6.4.8 BomAbstract* const stdair::AirportPair::getParent () const [inline]

Get a reference on the parent object instance.

Definition at line 87 of file AirportPair.hpp.

References _parent.

8.6.4.9 const HolderMap_T& stdair::AirportPair::getHolderMap () const [inline]

Get a reference on the children holder.

Definition at line 94 of file AirportPair.hpp.

References _holderMap.

8.6.5 Friends And Related Function Documentation

8.6.5.1 friend class FacBom [friend]

Definition at line 19 of file AirportPair.hpp.

8.6.5.2 friend class FacCloneBom [friend]

Definition at line 20 of file AirportPair.hpp.

8.6.5.3 friend class FacBomManager [friend]

Definition at line 21 of file AirportPair.hpp.

8.6.6 Member Data Documentation

8.6.6.1 Key_T stdair::AirportPair::_key [protected]

Primary key (flight number and departure date).

Definition at line 124 of file AirportPair.hpp.

Referenced by describeKey(), getBoardingPoint(), getKey(), and getOffPoint().

8.6.6.2 BomAbstract* stdair::AirportPair::_parent [protected]

Pointer on the parent class (Inventory).

Definition at line 129 of file AirportPair.hpp.

Referenced by getParent().

8.6.6.3 HolderMap_T stdair::AirportPair::_holderMap [protected]

Map holding the children.

Definition at line 134 of file AirportPair.hpp.

Referenced by getHolderMap().

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

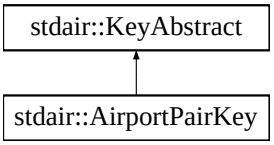
- stdair/bom/AirportPair.hpp
- stdair/bom/AirportPair.cpp

8.7 stdair::AirportPairKey Struct Reference

Key of airport-pair.

`#include <stdair/bom/AirportPairKey.hpp>`

Inheritance diagram for stdair::AirportPairKey::



Public Member Functions

- AirportPairKey (const stdair::AirportCode_T &, const stdair::AirportCode_T &)
- AirportPairKey (const AirportPairKey &)
- ~AirportPairKey ()
- const stdair::AirportCode_T & getBoardingPoint () const
- const stdair::AirportCode_T & getOffPoint () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const

8.7.1 Detailed Description

Key of airport-pair.

Definition at line 16 of file AirportPairKey.hpp.

8.7.2 Constructor & Destructor Documentation

8.7.2.1 stdair::AirportPairKey::AirportPairKey (const stdair::AirportCode_T &, const stdair::AirportCode_T &)

Main constructor.

Definition at line 22 of file AirportPairKey.cpp.

8.7.2.2 stdair::AirportPairKey::AirportPairKey (const AirportPairKey &)

Copy constructor.

Definition at line 28 of file AirportPairKey.cpp.

8.7.2.3 stdair::AirportPairKey::~AirportPairKey ()

Destructor.

Definition at line 34 of file AirportPairKey.cpp.

8.7.3 Member Function Documentation

8.7.3.1 const stdair::AirportCode_T& stdair::AirportPairKey::getBoardingPoint () const [inline]

Get the boarding point.

Definition at line 36 of file AirportPairKey.hpp.

Referenced by stdair::AirportPair::getBoardingPoint().

8.7.3.2 const stdair::AirportCode_T& stdair::AirportPairKey::getOffPoint () const [inline]

Get the arrival point.

Definition at line 43 of file AirportPairKey.hpp.

Referenced by stdair::AirportPair::getOffPoint().

8.7.3.3 void stdair::AirportPairKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 38 of file AirportPairKey.cpp.

References toString().

8.7.3.4 void stdair::AirportPairKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 43 of file AirportPairKey.cpp.

8.7.3.5 const std::string stdair::AirportPairKey::toString () const [virtual]

Get the serialised version of the Business Object Key. That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

Reimplemented from stdair::KeyAbstract.

Definition at line 47 of file AirportPairKey.cpp.

References stdair::DEFAULT_KEY_SUB_FLD_DELIMITER.

Referenced by stdair::AirportPair::describeKey(), stdair::BomRetriever::retrieveAirportPairFromKeySet(), and toStream().

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

- stdair/bom/AirportPairKey.hpp
- stdair/bom/AirportPairKey.cpp

8.8 stdair::BasChronometer Struct Reference

`#include <stdair/basic/BasChronometer.hpp>`

Public Member Functions

- BasChronometer ()
- void start ()
- std::string getStart () const
- double elapsed () const

8.8.1 Detailed Description

Structure allowing measuring the time elapsed between two events.

Definition at line 14 of file BasChronometer.hpp.

8.8.2 Constructor & Destructor Documentation

8.8.2.1 stdair::BasChronometer::BasChronometer ()

Constructor.

Definition at line 12 of file BasChronometer.cpp.

8.8.3 Member Function Documentation

8.8.3.1 void stdair::BasChronometer::start ()

Start the chronometer from the local time

The elapsed time given is the one elapsed since the start is launched.

Definition at line 16 of file BasChronometer.cpp.

8.8.3.2 std::string stdair::BasChronometer::getStart () const [inline]

Get the start time.

Definition at line 24 of file BasChronometer.hpp.

8.8.3.3 double stdair::BasChronometer::elapsed () const

Return the time elapsed since the structure has been instanciated.

That elapsed time is expressed in seconds.

Definition at line 26 of file BasChronometer.cpp.

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

- stdair/basic/BasChronometer.hpp

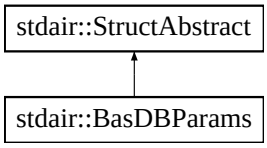
- stdair/basic/BasChronometer.cpp

8.9 stdair::BasDBParams Struct Reference

Structure holding the parameters for connection to a database.

`#include <stdair/basic/BasDBParams.hpp>`

Inheritance diagram for stdair::BasDBParams::



Public Member Functions

- const std::string & getUser () const
- const std::string & getPassword () const
- const std::string & getHost () const
- const std::string & getPort () const
- const std::string & getDBName () const
- void setUser (const std::string &iUser)
- void setPassword (const std::string &iPasswd)
- void setHost (const std::string &iHost)
- void setPort (const std::string &iPort)
- void setDBName (const std::string &iDBName)
- bool check () const
- const std::string describe () const
- std::string toShortString () const
- std::string toString () const
- BasDBParams (const std::string &iDBUser, const std::string &iDBPasswd, const std::string &iDBHost, const std::string &iDBPort, const std::string &iDBName)
- BasDBParams ()
- BasDBParams (const BasDBParams &)
- ~BasDBParams ()
- void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

8.9.1 Detailed Description

Structure holding the parameters for connection to a database.

Definition at line 19 of file BasDBParams.hpp.

8.9.2 Constructor & Destructor Documentation

8.9.2.1 stdair::BasDBParams::BasDBParams (const std::string & *iDBUser*, const std::string & *iDBPasswd*, const std::string & *iDBHost*, const std::string & *iDBPort*, const std::string & *iDBName*)

Main Constructor.

Definition at line 24 of file BasDBParams.cpp.

8.9.2.2 stdair::BasDBParams::BasDBParams ()

Default Constructor.

Definition at line 13 of file BasDBParams.cpp.

8.9.2.3 stdair::BasDBParams::BasDBParams (const BasDBParams &)

Default copy constructor.

Definition at line 17 of file BasDBParams.cpp.

8.9.2.4 stdair::BasDBParams::~BasDBParams ()

Destructor.

Definition at line 34 of file BasDBParams.cpp.

8.9.3 Member Function Documentation

8.9.3.1 const std::string& stdair::BasDBParams::getUser () const [inline]

Get the database user name.

Definition at line 23 of file BasDBParams.hpp.

8.9.3.2 const std::string& stdair::BasDBParams::getPassword () const [inline]

Get the database user password.

Definition at line 28 of file BasDBParams.hpp.

8.9.3.3 const std::string& stdair::BasDBParams::getHost () const [inline]

Get the database host name.

Definition at line 33 of file BasDBParams.hpp.

8.9.3.4 const std::string& stdair::BasDBParams::getPort () const [inline]

Get the database port number.

Definition at line 38 of file BasDBParams.hpp.

8.9.3.5 const std::string& stdair::BasDBParams::getDBName () const [inline]

Get the database name.

Definition at line 43 of file BasDBParams.hpp.

8.9.3.6 void stdair::BasDBParams::setUser (const std::string & *iUser*) [inline]

Set the database user name.

Definition at line 50 of file BasDBParams.hpp.

8.9.3.7 void stdair::BasDBParams::setPassword (const std::string & *iPasswd*) [inline]

Set the database password.

Definition at line 55 of file BasDBParams.hpp.

8.9.3.8 void stdair::BasDBParams::setHost (const std::string & *iHost*) [inline]

Set the database host name.

Definition at line 60 of file BasDBParams.hpp.

8.9.3.9 void stdair::BasDBParams::setPort (const std::string & *iPort*) [inline]

Set the database port number.

Definition at line 65 of file BasDBParams.hpp.

8.9.3.10 void stdair::BasDBParams::setDBName (const std::string & *iDBName*) [inline]

Set the database name.

Definition at line 70 of file BasDBParams.hpp.

8.9.3.11 bool stdair::BasDBParams::check () const

Check that all the parameters are fine.

Definition at line 57 of file BasDBParams.cpp.

8.9.3.12 const std::string stdair::BasDBParams::describe () const [virtual]

Get the serialised version of the DBParams structure.

Implements stdair::StructAbstract.

Definition at line 38 of file BasDBParams.cpp.

References toString().

8.9.3.13 std::string stdair::BasDBParams::toShortString () const

Get a short display of the DBParams structure.

Definition at line 43 of file BasDBParams.cpp.

8.9.3.14 std::string stdair::BasDBParams::toString () const

Get the serialised version of the DBParams structure.

Definition at line 50 of file BasDBParams.cpp.

Referenced by describe().

8.9.3.15 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const [inline, inherited]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References stdair::StructAbstract::describe().

8.9.3.16 virtual void stdair::StructAbstract::fromStream (std::istream & *ioIn*) [inline, virtual, inherited]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented in stdair::ProgressStatusSet, stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::EventStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 38 of file StructAbstract.hpp.

Referenced by operator>>().

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

- stdair/basic/BasDBParams.hpp
- stdair/basic/BasDBParams.cpp

8.10 stdair::BasFileMgr Struct Reference

`#include <stdair/basic/BasFileMgr.hpp>`

Static Public Member Functions

- static bool doesExistAndIsReadable (const std::string &iFilepath)

8.10.1 Detailed Description

Helper class for operations on files and on the file-system.

Definition at line 13 of file BasFileMgr.hpp.

8.10.2 Member Function Documentation

8.10.2.1 bool stdair::BasFileMgr::doesExistAndIsReadable (const std::string & *iFilepath*) [static]

Definition at line 23 of file BasFileMgr.cpp.

Referenced by stdair::BomINIImport::importINIConfig().

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

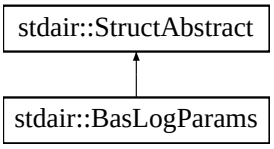
- stdair/basic/BasFileMgr.hpp
- stdair/basic/BasFileMgr.cpp

8.11 stdair::BasLogParams Struct Reference

Structure holding parameters for logging.

`#include <stdair/basic/BasLogParams.hpp>`

Inheritance diagram for stdair::BasLogParams::



Public Member Functions

- const LOG::EN_LogLevel & getLogLevel () const
- std::ostream & getLogStream () const
- const bool getForcedInitialisationFlag () const
- void setForcedInitialisationFlag (const bool iForceMultipleInstance)
- bool check () const
- const std::string describe () const
- std::string toShortString () const
- std::string toString () const
- BasLogParams (const LOG::EN_LogLevel iLogLevel, std::ostream &ioLogOutputStream, const bool iForceMultipleInstance=false)
- BasLogParams (const BasLogParams &)
- ~BasLogParams ()
- void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

Friends

- class Logger

8.11.1 Detailed Description

Structure holding parameters for logging.

Definition at line 19 of file BasLogParams.hpp.

8.11.2 Constructor & Destructor Documentation

8.11.2.1 stdair::BasLogParams::BasLogParams (const LOG::EN_LogLevel *iLogLevel*, std::ostream & *ioLogOutputStream*, const bool *iForceMultipleInstance* = `false`)

Main Constructor.

Parameters:

← ***const*** LOG::EN_LogLevel Level of the log (e.g., DEBUG)

↔ ***std::ostream&*** (STL) Stream to log into.

← ***const*** bool Whether or not multiple initialisation should be forced.

Definition at line 27 of file BasLogParams.cpp.

8.11.2.2 stdair::BasLogParams::BasLogParams (const BasLogParams &)

Copy constructor.

Definition at line 21 of file BasLogParams.cpp.

8.11.2.3 stdair::BasLogParams::~BasLogParams ()

Destructor.

Definition at line 35 of file BasLogParams.cpp.

8.11.3 Member Function Documentation

8.11.3.1 const LOG::EN_LogLevel& stdair::BasLogParams::getLogLevel () const `[inline]`

Get the log level.

Definition at line 26 of file BasLogParams.hpp.

8.11.3.2 std::ostream& stdair::BasLogParams::getLogStream () const `[inline]`

Get the log output stream.

Definition at line 33 of file BasLogParams.hpp.

8.11.3.3 const bool stdair::BasLogParams::getForcedInitialisationFlag () const `[inline]`

State whether or not multiple initialisations are to be forced.

Definition at line 40 of file BasLogParams.hpp.

8.11.3.4 void stdair::BasLogParams::setForcedInitialisationFlag (const bool *iForceMultipleInstance*) [inline]

State whether or not multiple initialisations are to be forced.

Definition at line 49 of file BasLogParams.hpp.

8.11.3.5 bool stdair::BasLogParams::check () const

Check that all the parameters are fine.

8.11.3.6 const std::string stdair::BasLogParams::describe () const [virtual]

Get the serialised version of the DBParams structure.

Implements stdair::StructAbstract.

Definition at line 39 of file BasLogParams.cpp.

References toString().

8.11.3.7 std::string stdair::BasLogParams::toShortString () const

Get a short display of the LOGParams structure.

Definition at line 44 of file BasLogParams.cpp.

References stdair::LOG::_logLevels.

8.11.3.8 std::string stdair::BasLogParams::toString () const

Get the serialised version of the LOGParams structure.

Definition at line 52 of file BasLogParams.cpp.

References stdair::LOG::_logLevels.

Referenced by describe().

8.11.3.9 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const [inline, inherited]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References stdair::StructAbstract::describe().

8.11.3.10 virtual void stdair::StructAbstract::fromStream (std::istream & *ioIn*) [inline, virtual, inherited]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented in stdair::ProgressStatusSet, stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::EventStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 38 of file StructAbstract.hpp.

Referenced by operator>>().

8.11.4 Friends And Related Function Documentation

8.11.4.1 friend class Logger [friend]

Definition at line 20 of file BasLogParams.hpp.

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

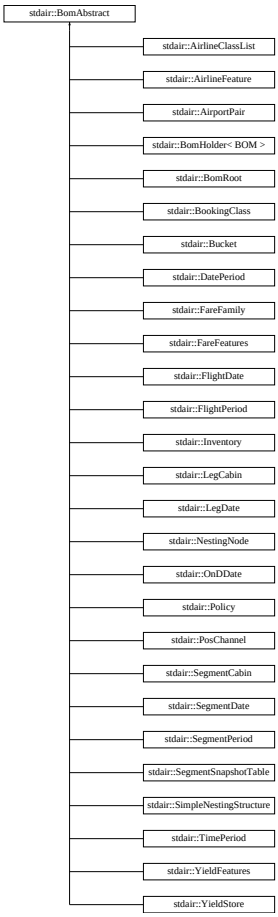
- stdair/basic/BasLogParams.hpp
- stdair/basic/BasLogParams.cpp

8.12 stdair::BomAbstract Class Reference

Base class for the Business Object Model (BOM) layer.

`#include <stdair/bom/BomAbstract.hpp>`

Inheritance diagram for stdair::BomAbstract::



Public Member Functions

- virtual void toStream (std::ostream &ioOut) const =0
- virtual void fromStream (std::istream &ioIn)=0
- virtual std::string toString () const =0
- virtual ~BomAbstract ()

Protected Member Functions

- BomAbstract ()
- BomAbstract (const BomAbstract &)

8.12.1 Detailed Description

Base class for the Business Object Model (BOM) layer.

Definition at line 24 of file BomAbstract.hpp.

8.12.2 Constructor & Destructor Documentation

8.12.2.1 stdair::BomAbstract::BomAbstract () [inline, protected]

Protected Default Constructor to ensure this class is abtract.

Definition at line 53 of file BomAbstract.hpp.

8.12.2.2 stdair::BomAbstract::BomAbstract (const BomAbstract &) [inline, protected]

Definition at line 54 of file BomAbstract.hpp.

8.12.2.3 virtual stdair::BomAbstract::~BomAbstract () [inline, virtual]

Destructor.

Definition at line 59 of file BomAbstract.hpp.

8.12.3 Member Function Documentation

8.12.3.1 virtual void stdair::BomAbstract::toStream (std::ostream & *ioOut*) const [pure virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& The input/output stream.

Implemented in stdair::AirlineClassList, stdair::AirlineFeature, stdair::AirportPair, stdair::BomHolder< BOM >, stdair::BomRoot, stdair::BookingClass, stdair::Bucket, stdair::DatePeriod, stdair::FareFamily, stdair::FareFeatures, stdair::FlightDate, stdair::FlightPeriod, stdair::Inventory, stdair::LegCabin, stdair::LegDate, stdair::NestingNode, stdair::OnDDate, stdair::Policy, stdair::PosChannel, stdair::SegmentCabin, stdair::SegmentDate, stdair::SegmentPeriod, stdair::SegmentSnapshotTable, stdair::SimpleNestingStructure, stdair::TimePeriod, stdair::YieldFeatures, and stdair::YieldStore.

8.12.3.2 virtual void stdair::BomAbstract::fromStream (std::istream & *ioIn*) [pure virtual]

Read a Business Object from an input stream.

Parameters:

istream& The input stream.

Implemented in stdair::AirlineClassList, stdair::AirlineFeature, stdair::AirportPair, stdair::BomHolder< BOM >, stdair::BomRoot, stdair::BookingClass, stdair::Bucket, stdair::DatePeriod, stdair::FareFamily, stdair::FareFeatures, stdair::FlightDate, stdair::FlightPeriod, stdair::Inventory, stdair::LegCabin, stdair::LegDate, stdair::NestingNode, stdair::OnDDate, stdair::Policy, stdair::PosChannel, stdair::SegmentCabin, stdair::SegmentDate, stdair::SegmentPeriod, stdair::SegmentSnapshotTable, stdair::SimpleNestingStructure, stdair::TimePeriod, stdair::YieldFeatures, and stdair::YieldStore.

Referenced by operator>>().

8.12.3.3 virtual std::string stdair::BomAbstract::toString () const [pure virtual]

Get the serialised version of the Business Object.

Returns:

std::string The output string

Implemented in stdair::AirlineClassList, stdair::AirlineFeature, stdair::AirportPair, stdair::BomHolder< BOM >, stdair::BomRoot, stdair::BookingClass, stdair::Bucket, stdair::DatePeriod, stdair::FareFamily, stdair::FareFeatures, stdair::FlightDate, stdair::FlightPeriod, stdair::Inventory, stdair::LegCabin, stdair::LegDate, stdair::NestingNode, stdair::OnDDate, stdair::Policy, stdair::PosChannel, stdair::SegmentCabin, stdair::SegmentDate, stdair::SegmentPeriod, stdair::SegmentSnapshotTable, stdair::SimpleNestingStructure, stdair::TimePeriod, stdair::YieldFeatures, and stdair::YieldStore.

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

- stdair/bom/BomAbstract.hpp

8.13 stdair::BomArchive Class Reference

Utility class to archive/restore BOM objects with Boost serialisation.

`#include <stdair/bom/BomArchive.hpp>`

Static Public Member Functions

- static void archive (const BomRoot &)
- static std::string archive (const Inventory &)
- static void restore (const std::string &iArchive, Inventory &)
- static void archive (const FlightDate &)

8.13.1 Detailed Description

Utility class to archive/restore BOM objects with Boost serialisation.

Definition at line 28 of file BomArchive.hpp.

8.13.2 Member Function Documentation

8.13.2.1 void stdair::BomArchive::archive (const BomRoot &) [static]

Recursively archive (dump in the underlying STL string) the objects of the BOM tree.

Parameters:

const BomRoot& Root of the BOM tree to be archived.

Definition at line 32 of file BomArchive.cpp.

8.13.2.2 std::string stdair::BomArchive::archive (const Inventory &) [static]

Recursively archive (dump in the underlying STL string) the objects of the BOM tree.

Parameters:

const Inventory& Root of the BOM tree to be archived.

Definition at line 36 of file BomArchive.cpp.

8.13.2.3 void stdair::BomArchive::restore (const std::string & *iArchive*, Inventory &) [static]

Recursively restore (from the underlying STL string) the objects of the BOM tree.

Parameters:

const std::string& String holding the serialised objects.

Inventory& Root of the BOM tree to be restored.

Definition at line 44 of file BomArchive.cpp.

8.13.2.4 void stdair::BomArchive::archive (const FlightDate &) [static]

Recursively archive (dump in the underlying STL string) the objects of the BOM tree.

Parameters:

const FlightDate& Root of the BOM tree to be archived.

Definition at line 52 of file BomArchive.cpp.

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

- stdair/bom/BomArchive.hpp
- stdair/bom/BomArchive.cpp

8.14 stdair::BomDisplay Class Reference

Utility class to display StdAir objects with a pretty format.

`#include <stdair/bom/BomDisplay.hpp>`

Static Public Member Functions

- static void list (std::ostream &, const BomRoot &, const AirlineCode_T &iAirlineCode="all", const FlightNumber_T &iFlightNumber=0)
- static void list (std::ostream &, const Inventory &, const unsigned short iInventoryIndex=0, const FlightNumber_T &iFlightNumber=0)
- static void listAirportPairDateRange (std::ostream &, const BomRoot &)
- static void csvDisplay (std::ostream &, const BomRoot &)
- static void csvDisplay (std::ostream &, const Inventory &)
- static void csvDisplay (std::ostream &, const OnDDate &)
- static void csvDisplay (std::ostream &, const FlightDate &)
- static void csvLegDateDisplay (std::ostream &, const FlightDate &)
- static void csvSegmentDateDisplay (std::ostream &, const FlightDate &)
- static void csvLegCabinDisplay (std::ostream &, const FlightDate &)
- static void csvSegmentCabinDisplay (std::ostream &, const FlightDate &)
- static void csvFareFamilyDisplay (std::ostream &, const FlightDate &)
- static void csvBucketDisplay (std::ostream &, const FlightDate &)
- static void csvBookingClassDisplay (std::ostream &, const BookingClass &, const std::string &iLeadingString)

- static void csvBookingClassDisplay (std::ostream &, const FlightDate &)
- static void csvDisplay (std::ostream &, const TravelSolutionList_T &)
- static void csvDisplay (std::ostream &, const DatePeriodList_T &)
- static void csvSimFQTAirRACDisplay (std::ostream &, const BomRoot &)
- static void csvAirportPairDisplay (std::ostream &, const AirportPair &)
- static void csvDateDisplay (std::ostream &, const DatePeriod &)
- static void csvPosChannelDisplay (std::ostream &, const PosChannel &)
- static void csvTimeDisplay (std::ostream &, const TimePeriod &)
- template<typename FEATURE_TYPE> static void csvFeatureListDisplay (std::ostream &oStream, const TimePeriod &)
- template<typename FEATURE_TYPE> static void csvFeaturesDisplay (std::ostream &oStream, const FEATURE_TYPE &)
- static void csvAirlineClassDisplay (std::ostream &, const AirlineClassList &)

8.14.1 Detailed Description

Utility class to display StdAir objects with a pretty format.

Definition at line 38 of file BomDisplay.hpp.

8.14.2 Member Function Documentation

8.14.2.1 static void stdair::BomDisplay::list (std::ostream &, const BomRoot &, const AirlineCode_T & *iAirlineCode* = "all", const FlightNumber_T & *iFlightNumber* = 0) [static]

Display (dump in the underlying output log stream) the list of flight-dates contained within the given BOM tree.

Parameters:

std::ostream& Output stream in which the flight-date keys should be logged/dumped.

const BomRoot& Root of the BOM tree to be displayed.

const AirlineCode& Airline for which the flight-dates should be displayed. If set to "all" (default), all the inventories will be displayed.

const FlightNumber_T& Flight number for which all the departure dates should be displayed. If set to 0 (the default), all the flight numbers will be displayed.

Referenced by stdair::STDAIR_Service::list().

8.14.2.2 static void stdair::BomDisplay::list (std::ostream &, const Inventory &, const unsigned short *iInventoryIndex* = 0, const FlightNumber_T & *iFlightNumber* = 0) [static]

Display (dump in the underlying output log stream) the list of flight-dates contained within the given BOM tree.

Parameters:

std::ostream& Output stream in which the flight-date keys should be logged/dumped.

const Inventory& Root of the BOM tree to be displayed.

const unsigned short Index, within the list, of the inventory. It is 0 when that inventory is displayed alone.

const FlightNumber_T& Flight number for which all the departure dates should be displayed. If set to 0 (the default), all the flight numbers will be displayed.

8.14.2.3 static void stdair::BomDisplay::listAirportPairDateRange (std::ostream &, const BomRoot &) [static]

Display the list of aiports pairs and date ranges (contained within the BOM tree)

Parameters:

std::ostream& Output stream in which the airport pairs and date ranges are logged/dumped.

const BomRoot& Root of the BOM tree to be displayed.

Referenced by stdair::STDAIR_Service::listAirportPairDateRange().

8.14.2.4 static void stdair::BomDisplay::csvDisplay (std::ostream &, const BomRoot &) [static]

Recursively display (dump in the underlying output log stream) the objects of the BOM tree.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const BomRoot& Root of the BOM tree to be displayed.

Referenced by stdair::STDAIR_Service::csvDisplay().

8.14.2.5 static void stdair::BomDisplay::csvDisplay (std::ostream &, const Inventory &) [static]

Recursively display (dump in the underlying output log stream) the objects of the BOM tree from the level of the given Inventory.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const Inventory& Root of the BOM tree to be displayed.

8.14.2.6 static void stdair::BomDisplay::csvDisplay (std::ostream &, const OnDDate &) [static]

Display the O&D date object information.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const OnDDate& the BOM to be displayed.

8.14.2.7 static void stdair::BomDisplay::csvDisplay (std::ostream &, const FlightDate &) [static]

Recursively display (dump in the underlying output log stream) the objects of the BOM tree from the level of the given FlightDate.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const FlightDate& Root of the BOM tree to be displayed.

8.14.2.8 static void stdair::BomDisplay::csvLegDateDisplay (std::ostream &, const FlightDate &) [static]

Recursively display (dump in the underlying output log stream) the leg-date level objects of the BOM tree.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const FlightDate& Root of the BOM tree to be displayed.

8.14.2.9 static void stdair::BomDisplay::csvSegmentDateDisplay (std::ostream &, const FlightDate &) [static]

Recursively display (dump in the underlying output log stream) the segment-date level objects of the BOM tree.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const FlightDate& Root of the BOM tree to be displayed.

8.14.2.10 static void stdair::BomDisplay::csvLegCabinDisplay (std::ostream &, const FlightDate &) [static]

Recursively display (dump in the underlying output log stream) the leg-cabin level objects of the BOM tree.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const FlightDate& Root of the BOM tree to be displayed.

8.14.2.11 static void stdair::BomDisplay::csvSegmentCabinDisplay (std::ostream &, const FlightDate &) [static]

Recursively display (dump in the underlying output log stream) the segment-cabin level objects of the BOM tree.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const FlightDate& Root of the BOM tree to be displayed.

8.14.2.12 static void stdair::BomDisplay::csvFareFamilyDisplay (std::ostream &, const FlightDate &) [static]

Recursively display (dump in the underlying output log stream) the fare families level objects of the BOM tree.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const FlightDate& Root of the BOM tree to be displayed.

8.14.2.13 static void stdair::BomDisplay::csvBucketDisplay (std::ostream &, const FlightDate &) [static]

Recursively display (dump in the underlying output log stream) the bucket holder level objects of the BOM tree.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const FlightDate& Root of the BOM tree to be displayed.

8.14.2.14 static void stdair::BomDisplay::csvBookingClassDisplay (std::ostream &, const BookingClass &, const std::string & *iLeadingString*) [static]

Display (dump in the underlying output log stream) the segment-class, without going recursively deeper in the BOM tree.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const BookingClass& Root of the BOM tree to be displayed.

const std::string& Leading string to be displayed.

8.14.2.15 static void stdair::BomDisplay::csvBookingClassDisplay (std::ostream &, const FlightDate &) [static]

Recursively display (dump in the underlying output log stream) the segment-class level objects of the BOM tree.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const FlightDate& Root of the BOM tree to be displayed.

8.14.2.16 static void stdair::BomDisplay::csvDisplay (std::ostream &, const TravelSolutionList_T &) [static]

Display (dump in the underlying output log stream) the full list of travel solution structures.

Parameters:

std::ostream& Output stream in which the list of travel solutions is logged/dumped.

TravelSolutionList_T& List of travel solutions to display.

8.14.2.17 static void stdair::BomDisplay::csvDisplay (std::ostream &, const DatePeriodList_T &) [static]

Display (dump in the underlying output log stream) the full list of date period fare rule sub bom tree.

Parameters:

std::ostream& Output stream in which the list of travel solutions is logged/dumped.

DatePeriodList_T& List of date period to display.

8.14.2.18 static void stdair::BomDisplay::csvSimFQTAirRACDisplay (std::ostream &, const BomRoot &) [static]

Recursively display (dump in the underlying output log stream) the objects of the BOM tree.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const BomRoot& Root of the BOM tree to be displayed.

8.14.2.19 static void stdair::BomDisplay::csvAirportPairDisplay (std::ostream &, const AirportPair &) [static]

Recursively display (dump in the underlying output log stream) the objects of the BOM tree from the level of the given airport pair.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const AirportPair& Root of the BOM tree to be displayed.

8.14.2.20 static void stdair::BomDisplay::csvDateDisplay (std::ostream &, const DatePeriod &) [static]

Recursively display (dump in the underlying output log stream) the objects of the BOM tree from the level of the given date range.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const DatePeriod& Root of the BOM tree to be displayed.

8.14.2.21 static void stdair::BomDisplay::csvPosChannelDisplay (std::ostream &, const PosChannel &) [static]

Recursively display (dump in the underlying output log stream) the objects of the BOM tree from the level of the given point of sale channel.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const PosChannel& Root of the BOM tree to be displayed.

8.14.2.22 static void stdair::BomDisplay::csvTimeDisplay (std::ostream &, const TimePeriod &) [static]

Recursively display (dump in the underlying output log stream) the objects of the BOM tree from the level of the given time range.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const TimePeriod& Root of the BOM tree to be displayed.

8.14.2.23 template<typename FEATURE_TYPE> static void stdair::BomDisplay::csvFeatureListDisplay (std::ostream & *oStream*, const TimePeriod &) [static]

Recursively display (dump in the underlying output log stream) the list of fare/yield features objects of the BOM tree.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const TimePeriod& Root of the BOM tree to be displayed.

8.14.2.24 template<typename FEATURE_TYPE> static void stdair::BomDisplay::csvFeaturesDisplay (std::ostream & *oStream*, const FEATURE_TYPE &) [static]

Recursively display (dump in the underlying output log stream) the fare/yield features objects of the BOM tree.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const FEATURE_TYPE& Root of the BOM tree to be displayed.

8.14.2.25 static void stdair::BomDisplay::csvAirlineClassDisplay (std::ostream &, const AirlineClassList &) [static]

Recursively display (dump in the underlying output log stream) the airline class objects of the BOM tree.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const AirlineClassList& Root of the BOM tree to be displayed.

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

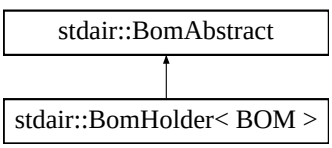
- stdair/bom/BomDisplay.hpp

8.15 stdair::BomHolder< BOM > Class Template Reference

Class representing the holder of BOM object containers (list and map).

`#include <stdair/bom/BomHolder.hpp>`

Inheritance diagram for stdair::BomHolder< BOM >::



Public Types

- typedef stdair::BomHolderKey Key_T
- typedef std::list< BOM * > BomList_T
- typedef std::map< const MapKey_T, BOM * > BomMap_T

Public Member Functions

- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const

Public Attributes

- Key_T _key
- BomList_T _bomList
- BomMap_T _bomMap

Protected Member Functions

- BomHolder ()
- BomHolder (const BomHolder &)
- BomHolder (const Key_T &iKey)
- ~BomHolder ()

Friends

- class FacBom
- class FacBomManager

8.15.1 Detailed Description

template<typename BOM> class stdair::BomHolder< BOM >

Class representing the holder of BOM object containers (list and map).

Definition at line 24 of file BomHolder.hpp.

8.15.2 Member Typedef Documentation

8.15.2.1 template<typename BOM> typedef stdair::BomHolderKey stdair::BomHolder< BOM >::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 34 of file BomHolder.hpp.

8.15.2.2 template<typename BOM> typedef std::list<BOM*> stdair::BomHolder< BOM >::BomList_T

(STL) list of children.

Definition at line 39 of file BomHolder.hpp.

8.15.2.3 template<typename BOM> typedef std::map<const MapKey_T, BOM*> stdair::BomHolder< BOM >::BomMap_T

(STL) map of children.

Definition at line 44 of file BomHolder.hpp.

8.15.3 Constructor & Destructor Documentation

8.15.3.1 template<typename BOM> stdair::BomHolder< BOM >::BomHolder () [protected]

Constructor.

8.15.3.2 template<typename BOM> stdair::BomHolder< BOM >::BomHolder (const BomHolder< BOM > &) [protected]

Copy constructor.

8.15.3.3 template<typename BOM> stdair::BomHolder< BOM >::BomHolder (const Key_T & *iKey*) [inline, protected]

Main constructor.

Definition at line 94 of file BomHolder.hpp.

8.15.3.4 template<typename BOM> stdair::BomHolder< BOM >::~BomHolder () [inline, protected]

Destructor.

Definition at line 99 of file BomHolder.hpp.

8.15.4 Member Function Documentation

8.15.4.1 template<typename BOM> void stdair::BomHolder< BOM >::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 54 of file BomHolder.hpp.

References stdair::BomHolder< BOM >::toString().

8.15.4.2 template<typename BOM> void stdair::BomHolder< BOM >::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 63 of file BomHolder.hpp.

8.15.4.3 template<typename BOM> std::string stdair::BomHolder< BOM >::toString () const [inline, virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 69 of file BomHolder.hpp.

Referenced by stdair::BomHolder< BOM >::toStream().

8.15.4.4 template<typename BOM> const std::string stdair::BomHolder< BOM >::describeKey () const [inline]

Get a string describing the key.

Definition at line 76 of file BomHolder.hpp.

8.15.5 Friends And Related Function Documentation

8.15.5.1 template<typename BOM> friend class FacBom [friend]

Friend classes.

Definition at line 26 of file BomHolder.hpp.

8.15.5.2 template<typename BOM> friend class FacBomManager [friend]

Definition at line 27 of file BomHolder.hpp.

8.15.6 Member Data Documentation

8.15.6.1 template<typename BOM> Key_T stdair::BomHolder< BOM >::_key

Key.

Definition at line 99 of file BomHolder.hpp.

8.15.6.2 template<typename BOM> BomList_T stdair::BomHolder< BOM >::_bomList

(STL) list of children.

Definition at line 111 of file BomHolder.hpp.

Referenced by stdair::FacBomManager::cloneHolder(), stdair::BomManager::getList(), stdair::BomManager::hasList(), and stdair::serialiseHelper().

8.15.6.3 template<typename BOM> BomMap_T stdair::BomHolder< BOM >::_bomMap

(STL) map of children.

Definition at line 116 of file BomHolder.hpp.

Referenced by stdair::FacBomManager::cloneHolder(), stdair::BomManager::getMap(), stdair::BomManager::hasMap(), and stdair::serialiseHelper().

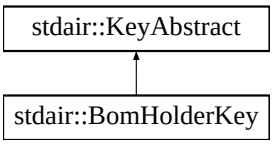
The documentation for this class was generated from the following file:

- stdair/bom/BomHolder.hpp

8.16 stdair::BomHolderKey Struct Reference

`#include <stdair/bom/BomHolderKey.hpp>`

Inheritance diagram for stdair::BomHolderKey::



Public Member Functions

- BomHolderKey ()
- ~BomHolderKey ()
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const
- const std::string describe () const

8.16.1 Detailed Description

Key of the BOM structure holder.

Definition at line 12 of file BomHolderKey.hpp.

8.16.2 Constructor & Destructor Documentation

8.16.2.1 stdair::BomHolderKey::BomHolderKey ()

Constructor.

Definition at line 13 of file BomHolderKey.cpp.

8.16.2.2 stdair::BomHolderKey::~BomHolderKey ()

Destructor.

Definition at line 17 of file BomHolderKey.cpp.

8.16.3 Member Function Documentation

8.16.3.1 void stdair::BomHolderKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 21 of file BomHolderKey.cpp.

References toString().

8.16.3.2 void stdair::BomHolderKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 26 of file BomHolderKey.cpp.

8.16.3.3 const std::string stdair::BomHolderKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from stdair::KeyAbstract.

Definition at line 30 of file BomHolderKey.cpp.

Referenced by toStream().

8.16.3.4 const std::string stdair::BomHolderKey::describe () const

Display of the key.

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

- stdair/bom/BomHolderKey.hpp
- stdair/bom/BomHolderKey.cpp

8.17 stdair::BomID< BOM > Struct Template Reference

Class wrapper of bom ID (e.g. pointer to object).

`#include <stdair/bom/BomID.hpp>`

Public Member Functions

- BOM & getObject () const
- BomID (BOM &iBOM)
- BomID (const BomID &)
- ~BomID ()

8.17.1 Detailed Description

template<typename BOM> struct stdair::BomID< BOM >

Class wrapper of bom ID (e.g. pointer to object).

Definition at line 17 of file BomID.hpp.

8.17.2 Constructor & Destructor Documentation

8.17.2.1 template<typename BOM> stdair::BomID< BOM >::BomID (BOM & *iBOM*)

Main constructor.

Definition at line 58 of file BomID.hpp.

8.17.2.2 template<typename BOM> stdair::BomID< BOM >::BomID (const BomID< BOM > &)

Copy constructor.

Definition at line 61 of file BomID.hpp.

8.17.2.3 template<typename BOM> stdair::BomID< BOM >::~BomID ()

Destructor.

Definition at line 65 of file BomID.hpp.

8.17.3 Member Function Documentation

8.17.3.1 template<typename BOM> BOM & stdair::BomID< BOM >::getObject () const

Retrieve the object.

Definition at line 68 of file BomID.hpp.

Referenced by stdair::CancellationStruct::describe(), and stdair::CancellationStruct::display().

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

- stdair/bom/BomID.hpp

8.18 stdair::BomINIImport Class Reference

Utility class to import StdAir objects in a INI format.

`#include <stdair/bom/BomINIImport.hpp>`

Static Public Member Functions

- static void importINIConfig (ConfigHolderStruct &, const ConfigINIFile &)

8.18.1 Detailed Description

Utility class to import StdAir objects in a INI format.

Definition at line 21 of file BomINIImport.hpp.

8.18.2 Member Function Documentation

8.18.2.1 void stdair::BomINIImport::importINIConfig (ConfigHolderStruct &, const ConfigINIFile &) [static]

Extract a boost property tree from an INI config file.

Parameters:

ConfigHolderStruct& Holder of the configuration tree.

const ConfigINIFile& INI config file.

Definition at line 29 of file BomINIImport.cpp.

References stdair::ConfigHolderStruct::add(), stdair::BasFileMgr::doesExistAndIsReadable(), stdair::RootFilePath::name(), and STDAIR_LOG_DEBUG.

Referenced by stdair::STDAIR_Service::importINIConfig().

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

- stdair/bom/BomINIImport.hpp
- stdair/bom/BomINIImport.cpp

8.19 stdair::BomJSONExport Class Reference

Utility class to export StdAir objects in a JSON format.

`#include <stdair/bom/BomJSONExport.hpp>`

Static Public Member Functions

- static void jsonExportFlightDateList (std::ostream &, const BomRoot &, const AirlineCode_T &iAirlineCode="all", const FlightNumber_T &iFlightNumber=0)
- static void jsonExportFlightDateObjects (std::ostream &, const FlightDate &)
- static void jsonExportBookingRequestObject (std::ostream &, const EventStruct &)
- static void jsonExportBreakPointObject (std::ostream &, const EventStruct &)

8.19.1 Detailed Description

Utility class to export StdAir objects in a JSON format.

Definition at line 42 of file BomJSONExport.hpp.

8.19.2 Member Function Documentation

8.19.2.1 void stdair::BomJSONExport::jsonExportFlightDateList (std::ostream &, const BomRoot &, const AirlineCode_T & *iAirlineCode* = "all", const FlightNumber_T & *iFlightNumber* = 0) [static]

Export (dump in the underlying output log stream and in JSON format) a list of flight date objects.

Parameters:

std::ostream& Output stream in which the flight date objects should be logged/dumped.

const BomRoot& Root of the BOM tree containing flight-dates to be exported.

const AirlineCode& Airline for which the flight-dates should be displayed. If set to "all" (default), all the inventories will be displayed.

const FlightNumber_T& Flight number for which all the departure dates should be displayed. If set to 0 (the default), all the flight numbers will be displayed.

Definition at line 35 of file BomJSONExport.cpp.

Referenced by stdair::STDAIR_Service::jsonExportFlightDateList().

8.19.2.2 void stdair::BomJSONExport::jsonExportFlightDateObjects (std::ostream &, const FlightDate &) [static]

Recursively export (dump in the underlying output log stream and in JSON format) the objects of the BOM tree using the given FlightDate as root.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const FlightDate& Root of the BOM tree to be exported.

Definition at line 163 of file BomJSONExport.cpp.

References stdair::FlightDate::getAirlineCode(), stdair::FlightDate::getDepartureDate(), and stdair::FlightDate::getFlightNumber().

Referenced by stdair::STDAIR_Service::jsonExportFlightDateObjects().

8.19.2.3 void stdair::BomJSONExport::jsonExportBookingRequestObject (std::ostream &, const EventStruct &) [static]

Export (dump in the underlying output log stream and in JSON format) the booking request object contained in the event structure.

Parameters:

std::ostream& Output stream in which the events should be logged/dumped.

const EventStruct& Booking request to be stored in JSON-ified format.

Definition at line 660 of file BomJSONExport.cpp.

References stdair::EventType::BKG_REQ, stdair::BookingRequestStruct::getBookingChannel(), stdair::EventStruct::getBookingRequest(), stdair::BookingRequestStruct::getDestination(), stdair::EventStruct::getEventType(), stdair::EventType::getLabel(), stdair::BookingRequestStruct::getOrigin(),

stdair::BookingRequestStruct::getPartySize(), stdair::BookingRequestStruct::getPOS(), stdair::BookingRequestStruct::getPreferedDepartureDate(), stdair::BookingRequestStruct::getPreferredCabin(), stdair::BookingRequestStruct::getPreferredDepartureTime(), stdair::BookingRequestStruct::getRequestDateTime(), stdair::BookingRequestStruct::getStayDuration(), and stdair::BookingRequestStruct::getWTP().

Referenced by stdair::STDAIR_Service::jsonExportEventObject().

8.19.2.4 void stdair::BomJSONExport::jsonExportBreakPointObject (std::ostream &, const EventStruct &) [static]

Export (dump in the underlying output log stream and in JSON format) the break point object contained in the event structure.

Parameters:

std::ostream& Output stream in which the events should be logged/dumped.

const EventStruct& Break point to be stored in JSON-ified format.

Definition at line 749 of file BomJSONExport.cpp.

References stdair::EventType::BRK_PT, stdair::EventStruct::getBreakPoint(), stdair::BreakPointStruct::getBreakPointTime(), stdair::EventStruct::getEventType(), and stdair::EventType::getLabel().

Referenced by stdair::STDAIR_Service::jsonExportEventObject().

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

- stdair/bom/BomJSONExport.hpp
- stdair/bom/BomJSONExport.cpp

8.20 stdair::BomJSONImport Class Reference

Utility class to import StdAir objects in a JSON format.

`#include <stdair/bom/BomJSONImport.hpp>`

Static Public Member Functions

- static bool jsonImportCommand (const JSONString &, JSonCommand::EN_JSonCommand &)
- static bool jsonImportInventoryKey (const JSONString &, AirlineCode_T &)
- static bool jsonImportFlightDate (const JSONString &, Date_T &)
- static bool jsonImportFlightNumber (const JSONString &, FlightNumber_T &)
- static bool jsonImportBreakPoints (const JSONString &, BreakPointList_T &)
- static bool jsonImportEventType (const JSONString &, EventType::EN_EventType &)
- static bool jsonImportConfig (const JSONString &, ConfigHolderStruct &)

8.20.1 Detailed Description

Utility class to import StdAir objects in a JSON format.

Definition at line 26 of file BomJSONImport.hpp.

8.20.2 Member Function Documentation

8.20.2.1 bool stdair::BomJSONImport::jsonImportCommand (const JSONString &, JSonCommand::EN_JSonCommand &) [static]

Extract the JSON command from a given JSON-formatted string.

Parameters:

const JSONString& JSON-formatted string.

JSonCommand::EN_JSonCommand& JSOM command extracted from the given string.

Returns:

bool State whether the extracting has been successful.

Definition at line 32 of file BomJSONImport.cpp.

References stdair::JSonCommand::getCommand(), and stdair::JSONString::getString().

8.20.2.2 bool stdair::BomJSONImport::jsonImportInventoryKey (const JSONString &, AirlineCode_T &) [static]

Extract the airline code from a given JSON-formatted string.

Parameters:

const JSONString& JSON-formatted string.

AirlineCode_T& Airline code extracted from the given string.

Returns:

bool State whether the extracting has been successful.

Definition at line 98 of file BomJSONImport.cpp.

References stdair::JSONString::getString().

8.20.2.3 bool stdair::BomJSONImport::jsonImportFlightDate (const JSONString &, Date_T &) [static]

Extract the FlightDate from a given JSON-formatted string.

Parameters:

const JSONString& JSON-formatted string.

Date_T& Departure date extracted from the given string.

Returns:

bool State whether the extracting has been successful.

Definition at line 133 of file BomJSONImport.cpp.

References stdair::JSONString::getString().

8.20.2.4 bool stdair::BomJSONImport::jsonImportFlightNumber (const JSONString &, FlightNumber_T &) [static]

Extract the FlightNumber from a given JSON-formatted string.

Parameters:

const JSONString& JSON-formatted string.

FlightNumber_T& Flight number extracted from the given string.

Returns:

bool State whether the extracting has been successful.

Definition at line 167 of file BomJSONImport.cpp.

References stdair::JSONString::getString().

8.20.2.5 bool stdair::BomJSONImport::jsonImportBreakPoints (const JSONString &, BreakPointList_T &) [static]

Extract the break points from a given JSON-formatted string.

Parameters:

const JSONString& JSON-formatted string.

BreakPointList_T& List of breaking points extracted from the given string.

Returns:

bool State whether the extracting has been successful.

Definition at line 203 of file BomJSONImport.cpp.

References stdair::JSONString::getString().

8.20.2.6 bool stdair::BomJSONImport::jsonImportEventType (const JSONString &, EventType::EN_EventType &) [static]

Extract the event type from a given JSON-formatted string.

Parameters:

const JSONString& JSON-formatted string.

EventType::EN_EventType& Event type extracted from the given string.

Returns:

bool State whether the extracting has been successful.

Definition at line 253 of file BomJSONImport.cpp.

References stdair::JSONString::getString().

8.20.2.7 bool stdair::BomJSONImport::jsonImportConfig (const JSONString &, ConfigHolderStruct &) [static]

Extract the configuration ptree from the given JSON-formatted string and add it to the configuration holder

Parameters:

const JSONString& JSON-formatted string.

ConfigHolderStruct& Configuration holder.

Returns:

bool State whether the extracting has been successful.

Definition at line 296 of file BomJSONImport.cpp.

References stdair::ConfigHolderStruct::add(), and stdair::JSONString::getString().

Referenced by stdair::STDAIR_Service::jsonImportConfiguration().

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

- stdair/bom/BomJSONImport.hpp
- stdair/bom/BomJSONImport.cpp

8.21 stdair::BomKeyManager Class Reference

Utility class to extract key structures from strings.

`#include <stdair/bom/BomKeyManager.hpp>`

Static Public Member Functions

- static ParsedKey extractKeys (const std::string &iFullKeyStr)
- static InventoryKey extractInventoryKey (const std::string &iFullKeyStr)
- static FlightDateKey extractFlightDateKey (const std::string &iFullKeyStr)
- static SegmentDateKey extractSegmentDateKey (const std::string &iFullKeyStr)
- static LegDateKey extractLegDateKey (const std::string &iFullKeyStr)

8.21.1 Detailed Description

Utility class to extract key structures from strings.

Definition at line 29 of file BomKeyManager.hpp.

8.21.2 Member Function Documentation

8.21.2.1 ParsedKey stdair::BomKeyManager::extractKeys (const std::string & *iFullKeyStr*) [static]

Build a ParsedKey structure from a full key string which includes an inventory key, flight-date key elements, segment-date key elements.

Definition at line 31 of file BomKeyManager.cpp.

References stdair::ParsedKey::_airlineCode, stdair::ParsedKey::_boardingPoint, stdair::ParsedKey::_boardingTime, stdair::ParsedKey::_departureDate, stdair::ParsedKey::_flightNumber, stdair::ParsedKey::_fullKey, stdair::ParsedKey::_offPoint, and stdair::DEFAULT_KEY_TOKEN_DELIMITER.

Referenced by stdair::TravelSolutionStruct::describe(), stdair::TravelSolutionStruct::describeSegmentPath(), stdair::TravelSolutionStruct::display(), extractFlightDateKey(), extractInventoryKey(), extractLegDateKey(), extractSegmentDateKey(), and stdair::BomRetriever::retrieveSegmentDateFromLongKey().

8.21.2.2 InventoryKey stdair::BomKeyManager::extractInventoryKey (const std::string & *iFullKeyStr*) [static]

Build a InventoryKey structure from a (full) key string.

The full key string gathers airline code, flight number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

Parameters:

const std::string& The full key string.

Returns:

InventoryKey The just built InventoryKey structure.

Definition at line 79 of file BomKeyManager.cpp.

References extractKeys(), and stdair::ParsedKey::getInventoryKey().

Referenced by stdair::BomRetriever::retrieveInventoryFromLongKey(), and stdair::BomRetriever::retrievePartnerSegmentDateFromLongKey().

8.21.2.3 FlightDateKey stdair::BomKeyManager::extractFlightDateKey (const std::string & *iFullKeyStr*) [static]

Build a FlightDateKey structure from a (full) key string.

The full key string gathers airline code, flight number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

Parameters:

const std::string& The full key string.

Returns:

FlightDateKey The just built FlightDateKey structure.

Definition at line 87 of file BomKeyManager.cpp.

References extractKeys(), and stdair::ParsedKey::getFlightDateKey().

Referenced by stdair::OnDDateKey::getDate(), and stdair::BomRetriever::retrieveFlightDateFromLongKey().

8.21.2.4 SegmentDateKey stdair::BomKeyManager::extractSegmentDateKey (const std::string & *iFullKeyStr*) [static]

Build a SegmentDateKey structure from a (full) key string.

The full key string gathers airline code, flight number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

Parameters:

const std::string& The full key string.

Returns:

SegmentDateKey The just built SegmentDateKey structure.

Definition at line 95 of file BomKeyManager.cpp.

References extractKeys(), and stdair::ParsedKey::getSegmentKey().

Referenced by stdair::OnDDateKey::getDestination(), stdair::OnDDateKey::getOrigin(), and stdair::BomRetriever::retrieveSegmentDateFromLongKey().

8.21.2.5 LegDateKey stdair::BomKeyManager::extractLegDateKey (const std::string & *iFullKeyStr*) [static]

Build a LegDateKey structure from a (full) key string.

The full key string gathers airline code, flight number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

Parameters:

const std::string& The full key string.

Returns:

LegDateKey The just built LegDateKey structure.

Definition at line 103 of file BomKeyManager.cpp.

References extractKeys(), and stdair::ParsedKey::getLegKey().

Referenced by stdair::BomRetriever::retrieveOperatingLegDateFromLongKey().

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

- stdair/bom/BomKeyManager.hpp
- stdair/bom/BomKeyManager.cpp

8.22 stdair::BomManager Class Reference

Utility class for StdAir-based objects.

`#include <stdair/bom/BomManager.hpp>`

Public Member Functions

- template<> bool hasList (const SegmentDate &ioSegmentDate)
- template<> const BomHolder< SegmentDate >::BomList_T & getList (const SegmentDate &ioSegmentDate)

- template<> bool hasMap (const SegmentDate &ioSegmentDate)
- template<> bool hasList (const Inventory &ioInventory)
- template<> bool hasMap (const Inventory &ioInventory)
- template<> AirlineFeature * getObjectPtr (const Inventory &iInventory, const MapKey_T &iKey)
- template<> AirlineFeature & getObject (const Inventory &iInventory, const MapKey_T &iKey)

Static Public Member Functions

- template<typename OBJECT2, typename OBJECT1> static const BomHolder< OBJECT2 >::BomList_T & getList (const OBJECT1 &)
- template<typename OBJECT2, typename OBJECT1> static const BomHolder< OBJECT2 >::BomMap_T & getMap (const OBJECT1 &)
- template<typename OBJECT2, typename OBJECT1> static bool hasList (const OBJECT1 &)
- template<typename OBJECT2, typename OBJECT1> static bool hasMap (const OBJECT1 &)
- template<typename PARENT, typename CHILD> static PARENT * getParentPtr (const CHILD &)
- template<typename PARENT, typename CHILD> static PARENT & getParent (const CHILD &)
- template<typename OBJECT2, typename OBJECT1> static OBJECT2 * getObjectPtr (const OBJECT1 &, const MapKey_T &)
- template<typename OBJECT2, typename OBJECT1> static OBJECT2 & getObject (const OBJECT1 &, const MapKey_T &)

Friends

- class FacBomManager

8.22.1 Detailed Description

Utility class for StdAir-based objects.

Most of those methods work for objects specified and instantianted outside StdAir, as long as those objects inherit from StdAir objects.

Definition at line 34 of file BomManager.hpp.

8.22.2 Member Function Documentation

8.22.2.1 template<typename OBJECT2, typename OBJECT1> const BomHolder< OBJECT2 >::BomList_T & stdair::BomManager::getList (const OBJECT1 &) [static]

Get the container (STL list) of OBJECT2 objects within the OBJECT1 object.

Definition at line 140 of file BomManager.hpp.

References stdair::BomHolder< BOM >::_bomList.

8.22.2.2 template<typename OBJECT2, typename OBJECT1> const BomHolder< OBJECT2 >::BomMap_T & stdair::BomManager::getMap (const OBJECT1 &) [static]

Get the container (STL map) of OBJECT2 objects within the OBJECT1 object.

Definition at line 159 of file BomManager.hpp.

References stdair::BomHolder< BOM >::_bomMap.

8.22.2.3 template<typename OBJECT2, typename OBJECT1> bool stdair::BomManager::hasList (const OBJECT1 &) [static]

Check if the list of object2 has been initialised.

Definition at line 181 of file BomManager.hpp.

References stdair::BomHolder< BOM >::_bomList.

8.22.2.4 template<typename OBJECT2, typename OBJECT1> bool stdair::BomManager::hasMap (const OBJECT1 &) [static]

Check if the map of object2 has been initialised.

Definition at line 201 of file BomManager.hpp.

References stdair::BomHolder< BOM >::_bomMap.

8.22.2.5 template<typename PARENT, typename CHILD> PARENT * stdair::BomManager::getParentPtr (const CHILD &) [static]

Get the PARENT of the given CHILD.

If the types do not match, NULL is returned.

Definition at line 220 of file BomManager.hpp.

8.22.2.6 template<typename PARENT, typename CHILD> PARENT & stdair::BomManager::getParent (const CHILD &) [static]

Get the PARENT of the given CHILD.

Definition at line 230 of file BomManager.hpp.

8.22.2.7 template<typename OBJECT2, typename OBJECT1> OBJECT2 * stdair::BomManager::getObjectPtr (const OBJECT1 &, const MapKey_T &) [static]

Get the OBJECT2 pointer corresponding to the given string key.

If such a OBJECT2 does not exist, return NULL.

Definition at line 241 of file BomManager.hpp.

8.22.2.8 template<typename OBJECT2, typename OBJECT1> OBJECT2 & stdair::BomManager::getObject (const OBJECT1 &, const MapKey_T &) [static]

Get the OBJECT2 corresponding to the given string key.

Definition at line 283 of file BomManager.hpp.

References STDAIR_LOG_ERROR.

8.22.2.9 template<> bool stdair::BomManager::hasList (const SegmentDate & *ioSegmentDate*) [inline]

8.22.2.10 template<> const BomHolder<SegmentDate>::BomList_T& stdair::BomManager::getList (const SegmentDate & *ioSegmentDate*) [inline]

8.22.2.11 template<> bool stdair::BomManager::hasMap (const SegmentDate & *ioSegmentDate*) [inline]

8.22.2.12 template<> bool stdair::BomManager::hasList (const Inventory & *ioInventory*) [inline]

8.22.2.13 template<> bool stdair::BomManager::hasMap (const Inventory & *ioInventory*) [inline]

8.22.2.14 template<> AirlineFeature* stdair::BomManager::getObjectPtr (const Inventory & *iInventory*, const MapKey_T & *iKey*) [inline]

8.22.2.15 template<> AirlineFeature& stdair::BomManager::getObject (const Inventory & *iInventory*, const MapKey_T & *iKey*) [inline]

8.22.3 Friends And Related Function Documentation

8.22.3.1 friend class FacBomManager [friend]

Definition at line 35 of file BomManager.hpp.

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

- stdair/bom/BomManager.hpp

8.23 stdair::BomRetriever Class Reference

Utility class to retrieve StdAir objects.

`#include <stdair/bom/BomRetriever.hpp>`

Static Public Member Functions

- static Inventory * retrieveInventoryFromLongKey (const BomRoot &, const std::string &iFullKeyStr)
- static Inventory * retrieveInventoryFromLongKey (const Inventory &, const std::string &iFullKeyStr)
- static Inventory * retrieveInventoryFromKey (const BomRoot &, const InventoryKey &)
- static Inventory * retrieveInventoryFromKey (const BomRoot &, const AirlineCode_T &)
- static AirlineFeature * retrieveAirlineFeatureFromKey (const BomRoot &, const AirlineCode_T &)
- static FlightDate * retrieveFlightDateFromLongKey (const BomRoot &, const std::string &iFullKeyStr)
- static FlightDate * retrieveFlightDateFromKeySet (const BomRoot &, const AirlineCode_T &, const FlightNumber_T &, const Date_T &iFlightDateDate)
- static FlightDate * retrieveFlightDateFromLongKey (const Inventory &, const std::string &iFullKeyStr)
- static FlightDate * retrieveFlightDateFromKey (const Inventory &, const FlightDateKey &)
- static FlightDate * retrieveFlightDateFromKey (const Inventory &, const FlightNumber_T &, const Date_T &iFlightDateDate)

- static LegDate * retrieveOperatingLegDateFromLongKey (const FlightDate &, const std::string &iFullKeyStr)
- static SegmentDate * retrievePartnerSegmentDateFromLongKey (const Inventory &, const std::string &iFullKeyStr)
- static SegmentDate * retrieveSegmentDateFromLongKey (const BomRoot &, const std::string &iFullKeyStr)
- static SegmentDate * retrieveSegmentDateFromLongKey (const Inventory &, const std::string &iFullKeyStr)
- static SegmentDate * retrieveSegmentDateFromLongKey (const FlightDate &, const std::string &iFullKeyStr)
- static SegmentDate * retrieveSegmentDateFromKey (const FlightDate &, const SegmentDateKey &)
- static SegmentDate * retrieveSegmentDateFromKey (const FlightDate &, const AirportCode_T &iOrigin, const AirportCode_T &iDestination)
- static BookingClass * retrieveBookingClassFromLongKey (const Inventory &, const std::string &iFullKeyStr, const ClassCode_T &)
- static AirportPair * retrieveAirportPairFromKeySet (const BomRoot &, const stdair::AirportCode_T &, const stdair::AirportCode_T &)
- static void retrieveDatePeriodListFromKey (const AirportPair &, const stdair::Date_T &, stdair::DatePeriodList_T &)
- static void retrieveDatePeriodListFromKeySet (const BomRoot &, const stdair::AirportCode_T &, const stdair::AirportCode_T &, const stdair::Date_T &, stdair::DatePeriodList_T &)
- static stdair::LegCabin & retrieveDummyLegCabin (stdair::BomRoot &, const bool isForFareFamilies=false)
- static stdair::SegmentCabin & retrieveDummySegmentCabin (stdair::BomRoot &, const bool isForFareFamilies=false)
- static std::string retrieveFullKeyFromSegmentDate (const SegmentDate &)

8.23.1 Detailed Description

Utility class to retrieve StdAir objects.

Definition at line 36 of file BomRetriever.hpp.

8.23.2 Member Function Documentation

8.23.2.1 Inventory * stdair::BomRetriever::retrieveInventoryFromLongKey (const BomRoot &, const std::string & *iFullKeyStr*) [static]

Retrieve an Inventory object from a (full) key string.

The full key string gathers airline code, flight number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

Parameters:

const BomRoot& The root of the BOM tree.

const std::string& The full key string.

Returns:

Inventory* The just retrieved Inventory object.

Definition at line 31 of file BomRetriever.cpp.

References stdair::BomKeyManager::extractInventoryKey(), and stdair::BomRoot::getInventory().

Referenced by retrieveFlightDateFromLongKey(), and retrievePartnerSegmentDateFromLongKey().

8.23.2.2 Inventory * stdair::BomRetriever::retrieveInventoryFromLongKey (const Inventory &, const std::string & *iFullKeyStr*) [static]

Retrieve an Inventory object from a (full) key string.

The full key string gathers airline code, flight number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

Parameters:

const Inventory& The root of the BOM tree.

const std::string& The full key string.

Returns:

Inventory* The just retrieved Inventory object.

Definition at line 46 of file BomRetriever.cpp.

References stdair::BomKeyManager::extractInventoryKey(), and stdair::InventoryKey::getAirlineCode().

8.23.2.3 Inventory * stdair::BomRetriever::retrieveInventoryFromKey (const BomRoot &, const InventoryKey &) [static]

Retrieve an Inventory object from an InventoryKey structure.

Parameters:

const BomRoot& The root of the BOM tree.

const InventoryKey& The key.

Returns:

Inventory* The just retrieved Inventory object.

Definition at line 63 of file BomRetriever.cpp.

References stdair::BomRoot::getInventory().

Referenced by retrieveAirlineFeatureFromKey(), retrieveDummyLegCabin(), retrieveDummySegmentCabin(), and retrieveFlightDateFromKeySet().

8.23.2.4 Inventory * stdair::BomRetriever::retrieveInventoryFromKey (const BomRoot &, const AirlineCode_T &) [static]

Retrieve an Inventory object from an InventoryKey structure.

Parameters:

const BomRoot& The root of the BOM tree.

const AirlineCode_T& The key.

Returns:

Inventory* The just retrieved Inventory object.

Definition at line 75 of file BomRetriever.cpp.

References stdair::BomRoot::getInventory().

8.23.2.5 AirlineFeature * stdair::BomRetriever::retrieveAirlineFeatureFromKey (const BomRoot &, const AirlineCode_T &) [static]

Retrieve an Airline Feature object from an airline code.

Parameters:

const BomRoot& The root of the BOM tree.

const AirlineCode_T& The key.

Returns:

AirlineFeature* The just retrieved Airline Feature object.

Definition at line 88 of file BomRetriever.cpp.

References retrieveInventoryFromKey().

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

8.23.2.6 FlightDate * stdair::BomRetriever::retrieveFlightDateFromLongKey (const BomRoot &, const std::string & *iFullKeyStr*) [static]

Retrieve a FlightDate object from a (full) key string.

The full key string gathers airline code, flight number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

Parameters:

const BomRoot& The root of the BOM tree.

const std::string& The full key string.

Returns:

FlightDate* The just retrieved FlightDate object.

Definition at line 109 of file BomRetriever.cpp.

References stdair::BomKeyManager::extractFlightDateKey(), stdair::Inventory::getFlightDate(), and retrieveInventoryFromLongKey().

Referenced by retrieveSegmentDateFromLongKey().

8.23.2.7 FlightDate * stdair::BomRetriever::retrieveFlightDateFromKeySet (const BomRoot &, const AirlineCode_T &, const FlightNumber_T &, const Date_T & *iFlightDateDate*) [static]

Retrieve a FlightDate object from a set of keys.

Parameters:

const BomRoot& The root of the BOM tree.

const AirlineCode_T& The key.

const FlightNumber_T& Part of the key.

const Date_T& Part of the key.

Returns:

FlightDate* The just retrieved FlightDate object.

Definition at line 132 of file BomRetriever.cpp.

References retrieveFlightDateFromKey(), and retrieveInventoryFromKey().

Referenced by stdair::STDAIR_Service::check(), stdair::STDAIR_Service::csvDisplay(), and stdair::STDAIR_Service::jsonExportFlightDateObjects().

8.23.2.8 FlightDate * stdair::BomRetriever::retrieveFlightDateFromLongKey (const Inventory &, const std::string & *iFullKeyStr*) [static]

Retrieve a FlightDate object from a (full) key string.

The full key string gathers airline code, flight number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

Parameters:

const Inventory& The root of the BOM tree.

const std::string& The full key string.

Returns:

FlightDate* The just retrieved FlightDate object.

Definition at line 155 of file BomRetriever.cpp.

References stdair::BomKeyManager::extractFlightDateKey(), and stdair::Inventory::getFlightDate().

8.23.2.9 FlightDate * stdair::BomRetriever::retrieveFlightDateFromKey (const Inventory &, const FlightDateKey &) [static]

Retrieve a FlightDate object from an FlightDateKey structure.

Parameters:

const Inventory& The root of the BOM tree.

const FlightDateKey& The key.

Returns:

FlightDate* The just retrieved FlightDate object.

Definition at line 170 of file BomRetriever.cpp.

References stdair::Inventory::getFlightDate().

Referenced by retrieveDummyLegCabin(), retrieveDummySegmentCabin(), retrieveFlightDateFromKeySet(), and retrieveSegmentDateFromLongKey().

8.23.2.10 FlightDate * stdair::BomRetriever::retrieveFlightDateFromKey (const Inventory &, const FlightNumber_T &, const Date_T & *iFlightDateDate*) [static]

Retrieve a FlightDate object from an FlightDateKey structure.

Parameters:

const Inventory& The root of the BOM tree.

const FlightNumber_T& Part of the key.

const Date_T& Part of the key.

Returns:

FlightDate* The just retrieved FlightDate object.

Definition at line 182 of file BomRetriever.cpp.

References stdair::Inventory::getFlightDate().

8.23.2.11 LegDate * stdair::BomRetriever::retrieveOperatingLegDateFromLongKey (const FlightDate &, const std::string & *iFullKeyStr*) [static]

Retrieve a LegDate object from an FlightDate structure.

The full key string gathers airline code, flight number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

Parameters:

const FlightDate& The root of the BOM tree.

const std::string& The full key string.

Returns:

LegDate* The just retrieved LegDate object.

Definition at line 266 of file BomRetriever.cpp.

References stdair::BomKeyManager::extractLegDateKey(), and stdair::FlightDate::getLegDate().

8.23.2.12 SegmentDate * stdair::BomRetriever::retrievePartnerSegmentDateFromLongKey (const Inventory &, const std::string & *iFullKeyStr*) [static]

Retrieve a partner SegmentDate object from an Inventory structure.

The full key string gathers airline code, flight number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

Parameters:

const Inventory& The root of the BOM tree.

const std::string& The full key string.

Returns:

SegmentDate* The just retrieved SegmentDate object.

Definition at line 281 of file BomRetriever.cpp.

References stdair::BomKeyManager::extractInventoryKey(), stdair::InventoryKey::getAirlineCode(), retrieveInventoryFromLongKey(), and retrieveSegmentDateFromLongKey().

8.23.2.13 SegmentDate * stdair::BomRetriever::retrieveSegmentDateFromLongKey (const BomRoot &, const std::string & *iFullKeyStr*) [static]

Retrieve a SegmentDate object from a (full) key string.

The full key string gathers airline code, segment number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

Parameters:

const BomRoot& The root of the BOM tree.

const std::string& The full key string.

Returns:

SegmentDate* The just retrieved SegmentDate object.

Definition at line 196 of file BomRetriever.cpp.

References stdair::BomKeyManager::extractSegmentDateKey(), stdair::FlightDate::getSegmentDate(), and retrieveFlightDateFromLongKey().

Referenced by retrieveBookingClassFromLongKey(), and retrievePartnerSegmentDateFromLongKey().

8.23.2.14 SegmentDate * stdair::BomRetriever::retrieveSegmentDateFromLongKey (const Inventory &, const std::string & *iFullKeyStr*) [static]

Retrieve a SegmentDate object from a (full) key string.

The full key string gathers airline code, segment number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

Parameters:

const Inventory& The root of the BOM tree.

const std::string& The full key string.

Returns:

SegmentDate* The just retrieved SegmentDate object.

Definition at line 219 of file BomRetriever.cpp.

References stdair::ParsedKey::_airlineCode, stdair::BomKeyManager::extractKeys(), stdair::Inventory::getAirlineCode(), stdair::ParsedKey::getFlightDateKey(), stdair::ParsedKey::getSegmentKey(), retrieveFlightDateFromKey(), retrieveSegmentDateFromKey(), STDAIR_LOG_DEBUG, stdair::SegmentDateKey::toString(), and stdair::FlightDateKey::toString().

8.23.2.15 SegmentDate * stdair::BomRetriever::retrieveSegmentDateFromLongKey (const FlightDate &, const std::string & *iFullKeyStr*) [static]

Retrieve a SegmentDate object from a (full) key string.

The full key string gathers airline code, segment number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

Parameters:

const FlightDate& The root of the BOM tree.

const std::string& The full key string.

Returns:

SegmentDate* The just retrieved SegmentDate object.

Definition at line 251 of file BomRetriever.cpp.

References stdair::BomKeyManager::extractSegmentDateKey(), and stdair::FlightDate::getSegmentDate().

8.23.2.16 SegmentDate * stdair::BomRetriever::retrieveSegmentDateFromKey (const FlightDate &, const SegmentDateKey &) [static]

Retrieve a SegmentDate object from an SegmentDateKey structure.

Parameters:

const FlightDate& The root of the BOM tree.

const SegmentDateKey& The key.

Returns:

SegmentDate* The just retrieved SegmentDate object.

Definition at line 307 of file BomRetriever.cpp.

References stdair::FlightDate::getSegmentDate().

Referenced by retrieveSegmentDateFromLongKey().

8.23.2.17 SegmentDate * stdair::BomRetriever::retrieveSegmentDateFromKey (const FlightDate &, const AirportCode_T & *iOrigin*, const AirportCode_T & *iDestination*) [static]

Retrieve a SegmentDate object from an SegmentDateKey structure.

Parameters:

const FlightDate& The root of the BOM tree.

const AirportCode_T& Origin, part of the key.

const AirportCode_T& Destination, part of the key.

Returns:

SegmentDate* The just retrieved SegmentDate object.

Definition at line 319 of file BomRetriever.cpp.

References stdair::FlightDate::getSegmentDate().

8.23.2.18 BookingClass * stdair::BomRetriever::retrieveBookingClassFromLongKey (const Inventory &, const std::string & *iFullKeyStr*, const ClassCode_T &) [static]

Retrieve a BookingClass object from a (full) key string.

The full key string gathers airline code, segment number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

Besides being attached to segment-cabin objects (and fare family objects, when they exist), the booking-class objects must also be attached directly to the segment-date.

Hence, if an assertion fails within that method call, chances are that the booking-class objects have not been attached to the segment-date objects. Check, for instance, the CmdBomManager::buildSampleBom() to see how that should be properly done.

Parameters:

const Inventory& The root of the BOM tree.

const std::string& Part of the full key string.

const ClassCode_T& Part of the full key string.

Returns:

BookingClass* The just retrieved BookingClass object.

Definition at line 333 of file BomRetriever.cpp.

References retrieveSegmentDateFromLongKey().

8.23.2.19 AirportPair * stdair::BomRetriever::retrieveAirportPairFromKeySet (const BomRoot &, const stdair::AirportCode_T &, const stdair::AirportCode_T &) [static]

Retrieve an AirportPair object from an AirportPair structure.

Parameters:

const BomRoot& The root of the BOM tree.

const AirportCode_T& Origin, part of the key.

const AirportCode_T& Destination, part of the key.

Returns:

AirportPair* The just retrieved AirportPair object.

Definition at line 355 of file BomRetriever.cpp.

References stdair::AirportPairKey::toString().

Referenced by retrieveDatePeriodListFromKeySet().

8.23.2.20 void stdair::BomRetriever::retrieveDatePeriodListFromKey (const AirportPair &, const stdair::Date_T &, stdair::DatePeriodList_T &) [static]

Retrieve a list of date-period corresponding to a flight date.

Parameters:

const AirportPair& The root of the BOM tree.

const Date_T& Departure Date of the flight

stdair::DatePeriodList_T& List of DatePeriod to display.

Definition at line 373 of file BomRetriever.cpp.

Referenced by retrieveDatePeriodListFromKeySet().

8.23.2.21 void stdair::BomRetriever::retrieveDatePeriodListFromKeySet (const BomRoot &, const stdair::AirportCode_T &, const stdair::AirportCode_T &, const stdair::Date_T &, stdair::DatePeriodList_T &) [static]

Retrieve a list of date-period from a set of keys.

Parameters:

const BomRoot& The root of the BOM tree.

const AirportCode_T& Part of the AirportPair key: the origin airport

const AirportCode_T& Part of the AirportPair key: the destination airport.

const Date_T& Departure date of the flight

stdair::DatePeriodList_T& List of DatePeriod to display.

Definition at line 404 of file BomRetriever.cpp.

References retrieveAirportPairFromKeySet(), and retrieveDatePeriodListFromKey().

Referenced by stdair::STDAIR_Service::check(), and stdair::STDAIR_Service::csvDisplay().

8.23.2.22 LegCabin & stdair::BomRetriever::retrieveDummyLegCabin (stdair::BomRoot &, const bool *isForFareFamilies* = false) [static]

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

Parameters:

stdair::BomRoot& The BOM tree.

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

Definition at line 427 of file BomRetriever.cpp.

References stdair::DEFAULT_AIRLINE_CODE, stdair::DEFAULT_CABIN_CODE, stdair::DEFAULT_DEPARTURE_DATE, stdair::DEFAULT_FLIGHT_NUMBER, stdair::DEFAULT_FLIGHT_NUMBER_FF, stdair::DEFAULT_ORIGIN, stdair::LegDate::getLegCabin(), retrieveFlightDateFromKey(), and retrieveInventoryFromKey().

8.23.2.23 SegmentCabin & stdair::BomRetriever::retrieveDummySegmentCabin (stdair::BomRoot &, const bool *isForFareFamilies* = false) [static]

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

Parameters:

stdair::BomRoot& The BOM tree.

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 502 of file BomRetriever.cpp.

References stdair::DEFAULT_AIRLINE_CODE, stdair::DEFAULT_CABIN_CODE, stdair::DEFAULT_DEPARTURE_DATE, stdair::DEFAULT_DESTINATION, stdair::DEFAULT_FLIGHT_NUMBER, stdair::DEFAULT_FLIGHT_NUMBER_FF, stdair::DEFAULT_ORIGIN, retrieveFlightDateFromKey(), retrieveInventoryFromKey(), and stdair::SegmentDate::toString().

8.23.2.24 std::string stdair::BomRetriever::retrieveFullKeyFromSegmentDate (const SegmentDate &) [static]

Retrieve the whole key of the segment date, that is to say a string composed of the inventory key, the flight date key and the segment date key.

Parameters:

const SegmentDate& Segment date to retrieve the whole key for.

Returns:

std::string The just retrieved whole key.

Definition at line 578 of file BomRetriever.cpp.

References stdair::DEFAULT_KEY_SUB_FLD_DELIMITER, stdair::SegmentDate::describeKey(), and stdair::Inventory::describeKey().

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

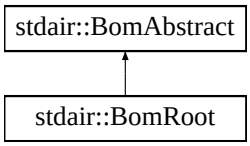
- stdair/bom/BomRetriever.hpp
- stdair/bom/BomRetriever.cpp

8.24 stdair::BomRoot Class Reference

Class representing the actual attributes for the Bom root.

`#include <stdair/bom/BomRoot.hpp>`

Inheritance diagram for stdair::BomRoot::



Public Types

- typedef BomRootKey Key_T

Public Member Functions

- const Key_T & getKey () const
- const HolderMap_T & getHolderMap () const
- const FRAT5Curve_T & getFRAT5Curve (const std::string &iKey) const
- const FFDisutilityCurve_T & getFFDisutilityCurve (const std::string &iKey) const
- Inventory * getInventory (const std::string &iInventoryKeyStr) const
- Inventory * getInventory (const InventoryKey &) const
- void addFRAT5Curve (const std::string &iKey, const FRAT5Curve_T &iCurve)
- void addFFDisutilityCurve (const std::string &iKey, const FFDisutilityCurve_T &iCurve)
- void toStream (std::ostream &ioOut) const

- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- template<class Archive> void serialize (Archive &ar, const unsigned int iFileVersion)

Protected Member Functions

- BomRoot ()
- BomRoot (const BomRoot &)
- BomRoot (const Key_T &iKey)
- ~BomRoot ()

Protected Attributes

- Key_T _key
- HolderMap_T _holderMap
- FRAT5CurveHolderStruct _frat5CurveHolder
- FFDisutilityCurveHolderStruct _ffDisutilityCurveHolder

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager
- class boost::serialization::access

8.24.1 Detailed Description

Class representing the actual attributes for the Bom root.

Definition at line 32 of file BomRoot.hpp.

8.24.2 Member Typedef Documentation

8.24.2.1 typedef BomRootKey stdair::BomRoot::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 42 of file BomRoot.hpp.

8.24.3 Constructor & Destructor Documentation

8.24.3.1 stdair::BomRoot::BomRoot () [protected]

Default constructor.

Definition at line 17 of file BomRoot.cpp.

8.24.3.2 stdair::BomRoot::BomRoot (const BomRoot &) [protected]

Copy constructor.

Definition at line 22 of file BomRoot.cpp.

8.24.3.3 stdair::BomRoot::BomRoot (const Key_T & *iKey*) [protected]

Main constructor.

Definition at line 28 of file BomRoot.cpp.

8.24.3.4 stdair::BomRoot::~BomRoot () [protected]

Destructor.

Definition at line 32 of file BomRoot.cpp.

8.24.4 Member Function Documentation

8.24.4.1 const Key_T& stdair::BomRoot::getKey () const [inline]

Get the inventory key (airline code).

Definition at line 48 of file BomRoot.hpp.

References _key.

8.24.4.2 const HolderMap_T& stdair::BomRoot::getHolderMap () const [inline]

Get the map of children.

Definition at line 53 of file BomRoot.hpp.

References _holderMap.

8.24.4.3 const FRAT5Curve_T& stdair::BomRoot::getFRAT5Curve (const std::string & *iKey*) const [inline]

Get the FRAT5 curve corresponding to the given key.

Definition at line 58 of file BomRoot.hpp.

References _frat5CurveHolder, and stdair::FRAT5CurveHolderStruct::getFRAT5Curve().

8.24.4.4 const FFDisutilityCurve_T& stdair::BomRoot::getFFDisutilityCurve (const std::string & *iKey*) const [inline]

Get the FFDisutility curve corresponding to the given key.

Definition at line 63 of file BomRoot.hpp.

References _ffDisutilityCurveHolder, and stdair::FFDisutilityCurveHolderStruct::getFFDisutilityCurve().

8.24.4.5 Inventory * stdair::BomRoot::getInventory (const std::string & *iInventoryKeyStr*) const

Get a pointer on the Inventory object corresponding to the given key.

Note:

The Inventory object can be inherited from, if needed. In that case, a dynamic_cast<> may be needed.

Parameters:

const std::string& The flight-date key.

Returns:

Inventory* Found Inventory object. NULL if not found.

Definition at line 43 of file BomRoot.cpp.

Referenced by getInventory(), stdair::BomRetriever::retrieveInventoryFromKey(), and stdair::BomRetriever::retrieveInventoryFromLongKey().

8.24.4.6 Inventory * stdair::BomRoot::getInventory (const InventoryKey &) const

Get a pointer on the Inventory object corresponding to the given key.

Note:

The Inventory object can be inherited from, if needed. In that case, a dynamic_cast<> may be needed.

Parameters:

const InventoryKey& The flight-date key

Returns:

Inventory* Found Inventory object. NULL if not found.

Definition at line 50 of file BomRoot.cpp.

References getInventory(), and stdair::InventoryKey::toString().

8.24.4.7 void stdair::BomRoot::addFRAT5Curve (const std::string & *iKey*, const FRAT5Curve_T & *iCurve*) [inline]

Add a new FRAT5 curve to the holder.

Definition at line 93 of file BomRoot.hpp.

References _frat5CurveHolder, and stdair::FRAT5CurveHolderStruct::addCurve().

8.24.4.8 void stdair::BomRoot::addFFDisutilityCurve (const std::string & *iKey*, const FFDisutilityCurve_T & *iCurve*) [inline]

Add a new FF disutility curve to the holder.

Definition at line 98 of file BomRoot.hpp.

References _ffDisutilityCurveHolder, and stdair::FFDisutilityCurveHolderStruct::addCurve().

8.24.4.9 void stdair::BomRoot::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 111 of file BomRoot.hpp.

References toString().

8.24.4.10 void stdair::BomRoot::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 120 of file BomRoot.hpp.

8.24.4.11 std::string stdair::BomRoot::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 36 of file BomRoot.cpp.

References _key, and stdair::BomRootKey::toString().

Referenced by toStream().

8.24.4.12 const std::string stdair::BomRoot::describeKey () const [inline]

Get a string describing the key.

Definition at line 131 of file BomRoot.hpp.

References _key, and stdair::BomRootKey::toString().

8.24.4.13 template<class Archive> void stdair::BomRoot::serialize (Archive & *ar*, const unsigned int *iFileVersion*)

Serialisation.

That method is used both for serialisation a BOM tree (into a backup file/stream), as well as re-instantiating a BOM tree from a back-up file/stream.

Note:

The implementation of that method is to be found in the CmdBomSerialiser command.

Definition at line 133 of file CmdBomSerialiser.cpp.

References _key.

8.24.5 Friends And Related Function Documentation

8.24.5.1 friend class FacBom [friend]

Definition at line 33 of file BomRoot.hpp.

8.24.5.2 friend class FacCloneBom [friend]

Definition at line 34 of file BomRoot.hpp.

8.24.5.3 friend class FacBomManager [friend]

Definition at line 35 of file BomRoot.hpp.

8.24.5.4 friend class boost::serialization::access [friend]

Definition at line 36 of file BomRoot.hpp.

8.24.6 Member Data Documentation

8.24.6.1 Key_T stdair::BomRoot::_key [protected]

Primary key.

Definition at line 191 of file BomRoot.hpp.

Referenced by describeKey(), getKey(), serialize(), and toString().

8.24.6.2 HolderMap_T stdair::BomRoot::_holderMap [protected]

Map holding the children (Inventory objects).

Definition at line 196 of file BomRoot.hpp.

Referenced by getHolderMap().

8.24.6.3 FRAT5CurveHolderStruct stdair::BomRoot::_frat5CurveHolder [protected]

Holder of FRAT5 curves.

Definition at line 201 of file BomRoot.hpp.

Referenced by addFRAT5Curve(), and getFRAT5Curve().

8.24.6.4 FFDisutilityCurveHolderStruct stdair::BomRoot::_ffDisutilityCurveHolder [protected]

Holder of fare family disutility curves.

Definition at line 206 of file BomRoot.hpp.

Referenced by addFFDisutilityCurve(), and getFFDisutilityCurve().

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

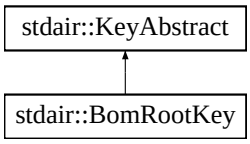
- stdair/bom/BomRoot.hpp
- stdair/bom/BomRoot.cpp
- stdair/command/CmdBomSerialiser.cpp

8.25 stdair::BomRootKey Struct Reference

Key of the BOM structure root.

`#include <stdair/bom/BomRootKey.hpp>`

Inheritance diagram for stdair::BomRootKey::



Public Member Functions

- BomRootKey ()
- BomRootKey (const std::string &iIdentification)
- BomRootKey (const BomRootKey &)
- ~BomRootKey ()
- const std::string & getID () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const
- template<class Archive> void serialize (Archive &ar, const unsigned int iFileVersion)

Friends

- class boost::serialization::access

8.25.1 Detailed Description

Key of the BOM structure root.

Definition at line 25 of file BomRootKey.hpp.

8.25.2 Constructor & Destructor Documentation

8.25.2.1 stdair::BomRootKey::BomRootKey ()

Default constructor.

Definition at line 18 of file BomRootKey.cpp.

8.25.2.2 stdair::BomRootKey::BomRootKey (const std::string & *iIdentification*)

Constructor.

Definition at line 28 of file BomRootKey.cpp.

8.25.2.3 stdair::BomRootKey::BomRootKey (const BomRootKey &)

Copy constructor.

Definition at line 23 of file BomRootKey.cpp.

8.25.2.4 stdair::BomRootKey::~BomRootKey ()

Destructor.

Definition at line 33 of file BomRootKey.cpp.

8.25.3 Member Function Documentation

8.25.3.1 const std::string& stdair::BomRootKey::getID () const [inline]

Get the identification.

Definition at line 56 of file BomRootKey.hpp.

8.25.3.2 void stdair::BomRootKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 37 of file BomRootKey.cpp.

References toString().

8.25.3.3 void stdair::BomRootKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 42 of file BomRootKey.cpp.

8.25.3.4 const std::string stdair::BomRootKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from stdair::KeyAbstract.

Definition at line 46 of file BomRootKey.cpp.

Referenced by stdair::BomRoot::describeKey(), toStream(), and stdair::BomRoot::toString().

8.25.3.5 template<class Archive> void stdair::BomRootKey::serialize (Archive & *ar*, const unsigned int *iFileVersion*)

Serialisation.

Definition at line 68 of file BomRootKey.cpp.

8.25.4 Friends And Related Function Documentation

8.25.4.1 friend class boost::serialization::access [friend]

Definition at line 26 of file BomRootKey.hpp.

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

- stdair/bom/BomRootKey.hpp
- stdair/bom/BomRootKey.cpp

8.26 stdair_test::BookingClass Struct Reference

`#include <test/stdair/StdairTestLib.hpp>`

Public Member Functions

- BookingClass (const std::string &iClassCode)
- std::string toString () const

Public Attributes

- std::string _classCode

8.26.1 Detailed Description

BookingClass

Definition at line 16 of file StdairTestLib.hpp.

8.26.2 Constructor & Destructor Documentation

8.26.2.1 stdair_test::BookingClass::BookingClass (const std::string & *iClassCode*) [inline]

Constructor.

Definition at line 19 of file StdairTestLib.hpp.

8.26.3 Member Function Documentation

8.26.3.1 std::string stdair_test::BookingClass::toString () const [inline]

Display .

Definition at line 24 of file StdairTestLib.hpp.

References _classCode.

8.26.4 Member Data Documentation

8.26.4.1 std::string stdair_test::BookingClass::_classCode

Definition at line 17 of file StdairTestLib.hpp.

Referenced by stdair_test::Cabin::toString(), and toString().

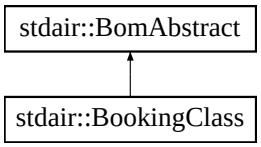
The documentation for this struct was generated from the following file:

- test/stdair/StdairTestLib.hpp

8.27 stdair::BookingClass Class Reference

`#include <stdair/bom/BookingClass.hpp>`

Inheritance diagram for stdair::BookingClass::



Public Types

- typedef BookingClassKey Key_T

Public Member Functions

- const Key_T & getKey () const
- const ClassCode_T & getClassCode () const
- BomAbstract *const getParent () const
- const HolderMap_T & getHolderMap () const
- const SubclassCode_T & getSubclassCode () const
- const AuthorizationLevel_T & getAuthorizationLevel () const
- const ProtectionLevel_T & getProtection () const
- const ProtectionLevel_T & getCumulatedProtection () const
- const BookingLimit_T & getCumulatedBookingLimit () const
- const NbOfSeats_T & getNegotiatedSpace () const
- const OverbookingRate_T & getNoShowPercentage () const
- const OverbookingRate_T & getCancellationPercentage () const
- const NbOfBookings_T & getNbOfBookings () const
- const NbOfBookings_T & getNbOfGroupBookings () const
- const NbOfBookings_T & getNbOfPendingGroupBookings () const
- const NbOfBookings_T & getNbOfStaffBookings () const
- const NbOfBookings_T & getNbOfWLBookings () const
- const NbOfCancellations_T & getNbOfCancellations () const
- const NbOfBookings_T & getETB () const
- const Availability_T & getNetClassAvailability () const
- const Availability_T & getSegmentAvailability () const
- const Availability_T & getNetRevenueAvailability () const
- const Yield_T & getYield () const
- const Yield_T & getAdjustedYield () const
- const MeanValue_T & getMean () const
- const StdDevValue_T & getStdDev () const
- const MeanValue_T & getPriceDemMean () const
- const StdDevValue_T & getPriceDemStdDev () const
- const MeanValue_T & getCumuPriceDemMean () const
- const StdDevValue_T & getCumuPriceDemStdDev () const
- const MeanValue_T & getProductDemMean () const
- const StdDevValue_T & getProductDemStdDev () const

- const GeneratedDemandVector_T & getGeneratedDemandVector () const
- void setCumulatedProtection (const ProtectionLevel_T &iPL)
- void setProtection (const ProtectionLevel_T &iPL)
- void setCumulatedBookingLimit (const BookingLimit_T &iBL)
- void setAuthorizationLevel (const AuthorizationLevel_T &iAU)
- void setSegmentAvailability (const Availability_T &iAvl)
- void setYield (const Yield_T &iYield)
- void setAdjustedYield (const Yield_T &iYield)
- void setMean (const MeanValue_T &iMean)
- void setStdDev (const StdDevValue_T &iStdDev)
- void setPriceDemMean (const MeanValue_T &iMean)
- void setPriceDemStdDev (const StdDevValue_T &iStdDev)
- void setCumuPriceDemMean (const MeanValue_T &iMean)
- void setCumuPriceDemStdDev (const StdDevValue_T &iStdDev)
- void setProductDemMean (const MeanValue_T &iMean)
- void setProductDemStdDev (const StdDevValue_T &iStdDev)
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- void sell (const NbOfBookings_T &)
- void cancel (const NbOfBookings_T &)
- void generateDemandSamples (const NbOfSamples_T &)
- void generateDemandSamples (const NbOfSamples_T &, const RandomSeed_T &)

Protected Member Functions

- BookingClass (const Key_T &)
- virtual ~BookingClass ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent
- HolderMap_T _holderMap
- SubclassCode_T _subclassCode
- ProtectionLevel_T _cumulatedProtection
- ProtectionLevel_T _protection
- BookingLimit_T _cumulatedBookingLimit
- AuthorizationLevel_T _au
- NbOfSeats_T _nego
- OverbookingRate_T _noShowPercentage
- OverbookingRate_T _cancellationPercentage
- NbOfBookings_T _nbOfBookings
- NbOfBookings_T _groupNbOfBookings
- NbOfBookings_T _groupPendingNbOfBookings
- NbOfBookings_T _staffNbOfBookings
- NbOfBookings_T _wlNbOfBookings
- NbOfCancellations_T _nbOfCancellations
- NbOfBookings_T _etb

- Availability_T _netClassAvailability
- Availability_T _segmentAvailability
- Availability_T _netRevenueAvailability
- Yield_T _yield
- Yield_T _adjustedYield
- MeanValue_T _mean
- StdDevValue_T _stdDev
- MeanValue_T _priceDemMean
- StdDevValue_T _priceDemStdDev
- MeanValue_T _cumuPriceDemMean
- StdDevValue_T _cumuPriceDemStdDev
- MeanValue_T _productDemMean
- StdDevValue_T _productDemStdDev
- GeneratedDemandVector_T _generatedDemandVector

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager

8.27.1 Detailed Description

Class representing the actual attributes for an airline booking class.

Definition at line 24 of file BookingClass.hpp.

8.27.2 Member Typedef Documentation

8.27.2.1 typedef BookingClassKey stdair::BookingClass::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 32 of file BookingClass.hpp.

8.27.3 Constructor & Destructor Documentation

8.27.3.1 stdair::BookingClass::BookingClass (const Key_T &) [protected]

Constructor.

Definition at line 49 of file BookingClass.cpp.

8.27.3.2 stdair::BookingClass::~BookingClass () [protected, virtual]

Destructor.

Definition at line 61 of file BookingClass.cpp.

8.27.4 Member Function Documentation

8.27.4.1 const Key_T& stdair::BookingClass::getKey () const [inline]

Get the booking class key.

Definition at line 37 of file BookingClass.hpp.

References _key.

8.27.4.2 const ClassCode_T& stdair::BookingClass::getClassCode () const [inline]

Get the booking code (part of the primary key).

Definition at line 42 of file BookingClass.hpp.

References _key, and stdair::BookingClassKey::getClassCode().

Referenced by stdair::CancellationStruct::describe(), and stdair::CancellationStruct::display().

8.27.4.3 BomAbstract* const stdair::BookingClass::getParent () const [inline]

Get the parent object.

Definition at line 47 of file BookingClass.hpp.

References _parent.

8.27.4.4 const HolderMap_T& stdair::BookingClass::getHolderMap () const [inline]

Get the map of children holders.

Definition at line 52 of file BookingClass.hpp.

References _holderMap.

8.27.4.5 const SubclassCode_T& stdair::BookingClass::getSubclassCode () const [inline]

Get teh sub-class code.

Definition at line 57 of file BookingClass.hpp.

References _subclassCode.

8.27.4.6 const AuthorizationLevel_T& stdair::BookingClass::getAuthorizationLevel () const [inline]

Get the authorisation level (AU, i.e., cumulated protection).

Definition at line 62 of file BookingClass.hpp.

References _au.

8.27.4.7 const ProtectionLevel_T& stdair::BookingClass::getProtection () const [inline]

Get the protection.

Definition at line 67 of file BookingClass.hpp.

References _protection.

8.27.4.8 const ProtectionLevel_T& stdair::BookingClass::getCumulatedProtection () const [inline]

Get the cumulated protection.

Definition at line 72 of file BookingClass.hpp.

References _cumulatedProtection.

8.27.4.9 const BookingLimit_T& stdair::BookingClass::getCumulatedBookingLimit () const [inline]

Get the cumulated booking limit.

Definition at line 77 of file BookingClass.hpp.

References _cumulatedBookingLimit.

8.27.4.10 const NbOfSeats_T& stdair::BookingClass::getNegotiatedSpace () const [inline]

Get the negotiated space.

Definition at line 82 of file BookingClass.hpp.

References _nego.

8.27.4.11 const OverbookingRate_T& stdair::BookingClass::getNoShowPercentage () const [inline]

Get the no-show rate.

Definition at line 87 of file BookingClass.hpp.

References _noShowPercentage.

8.27.4.12 const OverbookingRate_T& stdair::BookingClass::getCancellationPercentage () const [inline]

Get the cancellation rate.

Definition at line 92 of file BookingClass.hpp.

References _cancellationPercentage.

8.27.4.13 const NbOfBookings_T& stdair::BookingClass::getNbOfBookings () const [inline]

Get the number of bookings.

Definition at line 97 of file BookingClass.hpp.

References _nbOfBookings.

8.27.4.14 const NbOfBookings_T& stdair::BookingClass::getNbOfGroupBookings () const [inline]

Get the number of group bookings.

Definition at line 102 of file BookingClass.hpp.

References _groupNbOfBookings.

8.27.4.15 const NbOfBookings_T& stdair::BookingClass::getNbOfPendingGroupBookings () const [inline]

Get the number of pending group bookings.

Definition at line 107 of file BookingClass.hpp.

References _groupPendingNbOfBookings.

8.27.4.16 const NbOfBookings_T& stdair::BookingClass::getNbOfStaffBookings () const [inline]

Get the number of staff bookings.

Definition at line 112 of file BookingClass.hpp.

References _staffNbOfBookings.

8.27.4.17 const NbOfBookings_T& stdair::BookingClass::getNbOfWLBookings () const [inline]

Get the number of wait-list bookings.

Definition at line 117 of file BookingClass.hpp.

References _wlNbOfBookings.

8.27.4.18 const NbOfCancellations_T& stdair::BookingClass::getNbOfCancellations () const [inline]

Get the number of cancellations.

Definition at line 122 of file BookingClass.hpp.

References _nbOfCancellations.

8.27.4.19 const NbOfBookings_T& stdair::BookingClass::getETB () const [inline]

Get the expected number of passengers to board (ETB).

Definition at line 127 of file BookingClass.hpp.

References _etb.

8.27.4.20 const Availability_T& stdair::BookingClass::getNetClassAvailability () const [inline]

Get the net segment class availability.

Definition at line 132 of file BookingClass.hpp.

References _netClassAvailability.

8.27.4.21 const Availability_T& stdair::BookingClass::getSegmentAvailability () const [inline]

Get the segment class availability.

Definition at line 137 of file BookingClass.hpp.

References _segmentAvailability.

8.27.4.22 const Availability_T& stdair::BookingClass::getNetRevenueAvailability () const [inline]

Net revenue availability.

Definition at line 142 of file BookingClass.hpp.

References _netRevenueAvailability.

8.27.4.23 const Yield_T& stdair::BookingClass::getYield () const [inline]

Yield.

Definition at line 147 of file BookingClass.hpp.

References _yield.

8.27.4.24 const Yield_T& stdair::BookingClass::getAdjustedYield () const [inline]

Definition at line 148 of file BookingClass.hpp.

References _adjustedYield.

8.27.4.25 const MeanValue_T& stdair::BookingClass::getMean () const [inline]

Demand distribution.

Definition at line 151 of file BookingClass.hpp.

References _mean.

8.27.4.26 const StdDevValue_T& stdair::BookingClass::getStdDev () const [inline]

Definition at line 152 of file BookingClass.hpp.

References _stdDev.

8.27.4.27 const MeanValue_T& stdair::BookingClass::getPriceDemMean () const [inline]

Definition at line 153 of file BookingClass.hpp.

References _priceDemMean.

8.27.4.28 const StdDevValue_T& stdair::BookingClass::getPriceDemStdDev () const [inline]

Definition at line 154 of file BookingClass.hpp.

References _priceDemStdDev.

8.27.4.29 const MeanValue_T& stdair::BookingClass::getCumuPriceDemMean () const [inline]

Definition at line 155 of file BookingClass.hpp.

References _cumuPriceDemMean.

8.27.4.30 const StdDevValue_T& stdair::BookingClass::getCumuPriceDemStdDev () const [inline]

Definition at line 158 of file BookingClass.hpp.

References _cumuPriceDemStdDev.

8.27.4.31 const MeanValue_T& stdair::BookingClass::getProductDemMean () const [inline]

Definition at line 161 of file BookingClass.hpp.

References _productDemMean.

8.27.4.32 const StdDevValue_T& stdair::BookingClass::getProductDemStdDev () const [inline]

Definition at line 162 of file BookingClass.hpp.

References _productDemStdDev.

8.27.4.33 const GeneratedDemandVector_T& stdair::BookingClass::getGeneratedDemandVector () const [inline]

Generated demand vector.

Definition at line 165 of file BookingClass.hpp.

References _generatedDemandVector.

Referenced by stdair::VirtualClassStruct::getGeneratedDemandVector().

8.27.4.34 void stdair::BookingClass::setCumulatedProtection (const ProtectionLevel_T & *iPL*) [inline]

Cumulated protection.

Definition at line 172 of file BookingClass.hpp.

References _cumulatedProtection.

8.27.4.35 void stdair::BookingClass::setProtection (const ProtectionLevel_T & *iPL*) [inline]

Protection.

Definition at line 177 of file BookingClass.hpp.

References _protection.

8.27.4.36 void stdair::BookingClass::setCumulatedBookingLimit (const BookingLimit_T & *iBL*) [inline]

Cumulated booking limit.

Definition at line 182 of file BookingClass.hpp.

References _cumulatedBookingLimit.

8.27.4.37 void stdair::BookingClass::setAuthorizationLevel (const AuthorizationLevel_T & *iAU*) [inline]

Authorization level.

Definition at line 187 of file BookingClass.hpp.

References _au.

8.27.4.38 void stdair::BookingClass::setSegmentAvailability (const Availability_T & *iAvl*) [inline]

Set availability.

Definition at line 192 of file BookingClass.hpp.

References _segmentAvailability.

8.27.4.39 void stdair::BookingClass::setYield (const Yield_T & *iYield*) [inline]

Yield.

Definition at line 197 of file BookingClass.hpp.

References _adjustedYield, and _yield.

8.27.4.40 void stdair::BookingClass::setAdjustedYield (const Yield_T & *iYield*) [inline]

Definition at line 201 of file BookingClass.hpp.

References _adjustedYield.

8.27.4.41 void stdair::BookingClass::setMean (const MeanValue_T & *iMean*) [inline]

Demand distribution.

Definition at line 204 of file BookingClass.hpp.

References _mean.

8.27.4.42 void stdair::BookingClass::setStdDev (const StdDevValue_T & *iStdDev*) [inline]

Definition at line 205 of file BookingClass.hpp.

References _stdDev.

8.27.4.43 void stdair::BookingClass::setPriceDemMean (const MeanValue_T & *iMean*) [inline]

Definition at line 206 of file BookingClass.hpp.

References _priceDemMean.

8.27.4.44 void stdair::BookingClass::setPriceDemStdDev (const StdDevValue_T & *iStdDev*) [inline]

Definition at line 207 of file BookingClass.hpp.

References _priceDemStdDev.

8.27.4.45 void stdair::BookingClass::setCumuPriceDemMean (const MeanValue_T & *iMean*) [inline]

Definition at line 210 of file BookingClass.hpp.

References _cumuPriceDemMean.

8.27.4.46 void stdair::BookingClass::setCumuPriceDemStdDev (const StdDevValue_T & *iStdDev*) [inline]

Definition at line 212 of file BookingClass.hpp.

References _cumuPriceDemStdDev.

8.27.4.47 void stdair::BookingClass::setProductDemMean (const MeanValue_T & *iMean*) [inline]

Definition at line 215 of file BookingClass.hpp.

References _productDemMean.

8.27.4.48 void stdair::BookingClass::setProductDemStdDev (const StdDevValue_T & *iStdDev*) [inline]

Definition at line 218 of file BookingClass.hpp.

References _productDemStdDev.

8.27.4.49 void stdair::BookingClass::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 226 of file BookingClass.hpp.

References toString().

8.27.4.50 void stdair::BookingClass::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 232 of file BookingClass.hpp.

8.27.4.51 std::string stdair::BookingClass::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 65 of file BookingClass.cpp.

References describeKey().

Referenced by toStream().

8.27.4.52 const std::string stdair::BookingClass::describeKey () const [inline]

Get a string describing the key.

Definition at line 239 of file BookingClass.hpp.

References _key, and stdair::BookingClassKey::toString().

Referenced by toString().

8.27.4.53 void stdair::BookingClass::sell (const NbOfBookings_T &)

Register a sale.

Definition at line 72 of file BookingClass.cpp.

References _nbOfBookings.

8.27.4.54 void stdair::BookingClass::cancel (const NbOfBookings_T &)

Register a cancellation.

Definition at line 77 of file BookingClass.cpp.

References _nbOfBookings, and _nbOfCancellations.

8.27.4.55 void stdair::BookingClass::generateDemandSamples (const NbOfSamples_T &)

Generate demand samples for Monte-Carlo method with the default random seed.

Definition at line 83 of file BookingClass.cpp.

References _generatedDemandVector, _mean, _stdDev, stdair::DEFAULT_RANDOM_SEED, and stdair::RandomGeneration::generateNormal().

8.27.4.56 void stdair::BookingClass::generateDemandSamples (const NbOfSamples_T &, const RandomSeed_T &)

Generate demand samples for Monte-Carlo method with the given random seed.

Definition at line 95 of file BookingClass.cpp.

References _generatedDemandVector, _mean, _stdDev, and stdair::RandomGeneration::generateNormal().

8.27.5 Friends And Related Function Documentation

8.27.5.1 friend class FacBom [friend]

Definition at line 25 of file BookingClass.hpp.

8.27.5.2 friend class FacCloneBom [friend]

Definition at line 26 of file BookingClass.hpp.

8.27.5.3 friend class FacBomManager [friend]

Definition at line 27 of file BookingClass.hpp.

8.27.6 Member Data Documentation

8.27.6.1 Key_T stdair::BookingClass::_key [protected]

Primary key (booking class code).

Definition at line 276 of file BookingClass.hpp.

Referenced by describeKey(), getClassCode(), and getKey().

8.27.6.2 BomAbstract* stdair::BookingClass::_parent [protected]

Pointer on the parent class (SegmentCabin).

Definition at line 279 of file BookingClass.hpp.

Referenced by getParent().

8.27.6.3 HolderMap_T stdair::BookingClass::_holderMap [protected]

Map holding the children (SegmentDate and LegDate objects).

Definition at line 282 of file BookingClass.hpp.

Referenced by getHolderMap().

8.27.6.4 SubclassCode_T stdair::BookingClass::_subclassCode [protected]

Sub-class code.

Definition at line 285 of file BookingClass.hpp.

Referenced by getSubclassCode().

8.27.6.5 ProtectionLevel_T stdair::BookingClass::_cumulatedProtection [protected]

Cumulated protection.

Definition at line 288 of file BookingClass.hpp.

Referenced by getCumulatedProtection(), and setCumulatedProtection().

8.27.6.6 ProtectionLevel_T stdair::BookingClass::_protection [protected]

Protection.

Definition at line 291 of file BookingClass.hpp.

Referenced by getProtection(), and setProtection().

8.27.6.7 BookingLimit_T stdair::BookingClass::_cumulatedBookingLimit [protected]

Cumulated booking limit.

Definition at line 294 of file BookingClass.hpp.

Referenced by getCumulatedBookingLimit(), and setCumulatedBookingLimit().

8.27.6.8 AuthorizationLevel_T stdair::BookingClass::_au [protected]

Authorization level.

Definition at line 297 of file BookingClass.hpp.

Referenced by getAuthorizationLevel(), and setAuthorizationLevel().

8.27.6.9 NbOfSeats_T stdair::BookingClass::_nego [protected]

Negotiated space.

Definition at line 300 of file BookingClass.hpp.

Referenced by getNegotiatedSpace().

8.27.6.10 OverbookingRate_T stdair::BookingClass::_noShowPercentage [protected]

Overbooking rate.

Definition at line 303 of file BookingClass.hpp.

Referenced by getNoShowPercentage().

8.27.6.11 OverbookingRate_T stdair::BookingClass::_cancellationPercentage [protected]

Cancellation rate.

Definition at line 306 of file BookingClass.hpp.

Referenced by getCancellationPercentage().

8.27.6.12 NbOfBookings_T stdair::BookingClass::_nbOfBookings [protected]

Number of bookings.

Definition at line 309 of file BookingClass.hpp.

Referenced by cancel(), getNbOfBookings(), and sell().

8.27.6.13 NbOfBookings_T stdair::BookingClass::_groupNbOfBookings [protected]

Number of group bookings.

Definition at line 312 of file BookingClass.hpp.

Referenced by getNbOfGroupBookings().

8.27.6.14 NbOfBookings_T stdair::BookingClass::_groupPendingNbOfBookings [protected]

Number of pending group bookings.

Definition at line 315 of file BookingClass.hpp.

Referenced by getNbOfPendingGroupBookings().

8.27.6.15 NbOfBookings_T stdair::BookingClass::_staffNbOfBookings [protected]

Number of staff bookings.

Definition at line 318 of file BookingClass.hpp.

Referenced by getNbOfStaffBookings().

8.27.6.16 NbOfBookings_T stdair::BookingClass::_wlNbOfBookings [protected]

Number of wait-list bookings.

Definition at line 321 of file BookingClass.hpp.

Referenced by getNbOfWLBookings().

8.27.6.17 NbOfCancellations_T stdair::BookingClass::_nbOfCancellations [protected]

Number of cancellations.

Definition at line 324 of file BookingClass.hpp.

Referenced by cancel(), and getNbOfCancellations().

8.27.6.18 NbOfBookings_T stdair::BookingClass::_etb [protected]

Expected to board (ETB).

Definition at line 327 of file BookingClass.hpp.

Referenced by getETB().

8.27.6.19 Availability_T stdair::BookingClass::_netClassAvailability [protected]

Net segment class availability.

Definition at line 330 of file BookingClass.hpp.

Referenced by getNetClassAvailability().

8.27.6.20 Availability_T stdair::BookingClass::_segmentAvailability [protected]

Segment class availability.

Definition at line 333 of file BookingClass.hpp.

Referenced by getSegmentAvailability(), and setSegmentAvailability().

8.27.6.21 Availability_T stdair::BookingClass::_netRevenueAvailability [protected]

Net revenue availability.

Definition at line 336 of file BookingClass.hpp.

Referenced by getNetRevenueAvailability().

8.27.6.22 Yield_T stdair::BookingClass::_yield [protected]

Yield.

Definition at line 339 of file BookingClass.hpp.

Referenced by getYield(), and setYield().

8.27.6.23 Yield_T stdair::BookingClass::_adjustedYield [protected]

Definition at line 340 of file BookingClass.hpp.

Referenced by getAdjustedYield(), setAdjustedYield(), and setYield().

8.27.6.24 MeanValue_T stdair::BookingClass::_mean [protected]

Demand distribution forecast.

Definition at line 343 of file BookingClass.hpp.

Referenced by generateDemandSamples(), getMean(), and setMean().

8.27.6.25 StdDevValue_T stdair::BookingClass::_stdDev [protected]

Definition at line 344 of file BookingClass.hpp.

Referenced by generateDemandSamples(), getStdDev(), and setStdDev().

8.27.6.26 MeanValue_T stdair::BookingClass::_priceDemMean [protected]

Price-oriented demand distribution forecast.

Definition at line 347 of file BookingClass.hpp.

Referenced by getPriceDemMean(), and setPriceDemMean().

8.27.6.27 StdDevValue_T stdair::BookingClass::_priceDemStdDev [protected]

Definition at line 348 of file BookingClass.hpp.

Referenced by getPriceDemStdDev(), and setPriceDemStdDev().

8.27.6.28 MeanValue_T stdair::BookingClass::_cumuPriceDemMean [protected]

Cumulative price-oriented demand distribution forecast.

Definition at line 351 of file BookingClass.hpp.

Referenced by getCumuPriceDemMean(), and setCumuPriceDemMean().

8.27.6.29 StdDevValue_T stdair::BookingClass::_cumuPriceDemStdDev [protected]

Definition at line 352 of file BookingClass.hpp.

Referenced by getCumuPriceDemStdDev(), and setCumuPriceDemStdDev().

8.27.6.30 MeanValue_T stdair::BookingClass::_productDemMean [protected]

Product-oriented demand distribution forecast.

Definition at line 355 of file BookingClass.hpp.

Referenced by getProductDemMean(), and setProductDemMean().

8.27.6.31 StdDevValue_T stdair::BookingClass::_productDemStdDev [protected]

Definition at line 356 of file BookingClass.hpp.

Referenced by getProductDemStdDev(), and setProductDemStdDev().

8.27.6.32 GeneratedDemandVector_T stdair::BookingClass::_generatedDemandVector [protected]

Vector of number of demand samples drawn from the demand distribution.

Definition at line 359 of file BookingClass.hpp.

Referenced by generateDemandSamples(), and getGeneratedDemandVector().

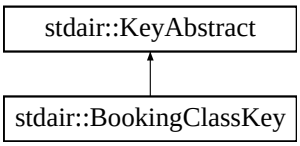
The documentation for this class was generated from the following files:

- stdair/bom/BookingClass.hpp
- stdair/bom/BookingClass.cpp

8.28 stdair::BookingClassKey Struct Reference

`#include <stdair/bom/BookingClassKey.hpp>`

Inheritance diagram for stdair::BookingClassKey::



Public Member Functions

- BookingClassKey (const ClassCode_T &iClassCode)
- BookingClassKey (const BookingClassKey &)
- ~BookingClassKey ()
- const ClassCode_T & getClassCode () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const

8.28.1 Detailed Description

Key of a given leg-cabin, made of a cabin code.

Definition at line 16 of file BookingClassKey.hpp.

8.28.2 Constructor & Destructor Documentation

8.28.2.1 stdair::BookingClassKey::BookingClassKey (const ClassCode_T & *iClassCode*)

Constructor.

Definition at line 24 of file BookingClassKey.cpp.

8.28.2.2 stdair::BookingClassKey::BookingClassKey (const BookingClassKey &)

Default copy constructor.

Definition at line 19 of file BookingClassKey.cpp.

8.28.2.3 stdair::BookingClassKey::~BookingClassKey ()

Destructor.

Definition at line 29 of file BookingClassKey.cpp.

8.28.3 Member Function Documentation

8.28.3.1 const ClassCode_T& stdair::BookingClassKey::getClassCode () const [inline]

Get the class code.

Definition at line 34 of file BookingClassKey.hpp.

Referenced by stdair::BookingClass::getClassCode().

8.28.3.2 void stdair::BookingClassKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 33 of file BookingClassKey.cpp.

References toString().

8.28.3.3 void stdair::BookingClassKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 38 of file BookingClassKey.cpp.

8.28.3.4 const std::string stdair::BookingClassKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-cabin.

Reimplemented from stdair::KeyAbstract.

Definition at line 42 of file BookingClassKey.cpp.

Referenced by stdair::BookingClass::describeKey(), and toStream().

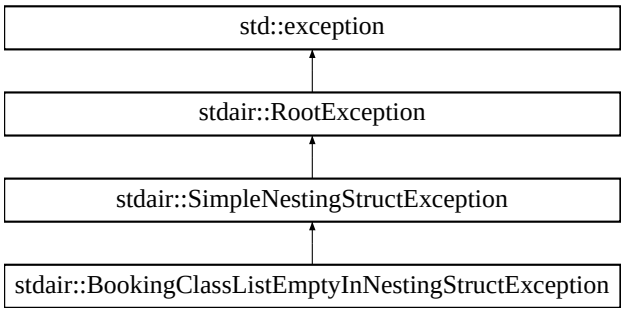
The documentation for this struct was generated from the following files:

- stdair/bom/BookingClassKey.hpp
- stdair/bom/BookingClassKey.cpp

8.29 stdair::BookingClassListEmptyInNestingStructException Class Reference

`#include <stdair/stdair_exceptions.hpp>`

Inheritance diagram for stdair::BookingClassListEmptyInNestingStructException::



Public Member Functions

- BookingClassListEmptyInNestingStructException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

8.29.1 Detailed Description

Empty booking class list in Simple Nesting Structure.

Definition at line 219 of file stdair_exceptions.hpp.

8.29.2 Constructor & Destructor Documentation

8.29.2.1 stdair::BookingClassListEmptyInNestingStructException::BookingClassListEmptyInNestingStructException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 223 of file stdair_exceptions.hpp.

8.29.3 Member Function Documentation

8.29.3.1 const char* stdair::RootException::what () const throw () [inline, inherited]

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

8.29.4 Member Data Documentation

8.29.4.1 std::string stdair::RootException::_what [protected, inherited]

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

Referenced by stdair::RootException::what().

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

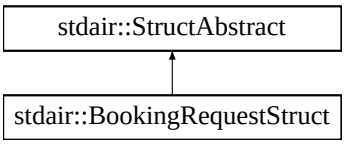
- stdair/stdair_exceptions.hpp

8.30 stdair::BookingRequestStruct Struct Reference

Structure holding the elements of a booking request.

`#include <stdair/bom/BookingRequestStruct.hpp>`

Inheritance diagram for stdair::BookingRequestStruct::



Public Member Functions

- const DemandGeneratorKey_T & getDemandGeneratorKey () const
- const AirportCode_T & getOrigin () const
- const AirportCode_T & getDestination () const
- const CityCode_T & getPOS () const
- const Date_T & getPreferedDepartureDate () const

- const Duration_T & getPreferredDepartureTime () const
- const DateTime_T & getRequestDateTime () const
- const CabinCode_T & getPreferredCabin () const
- const NbOfSeats_T & getPartySize () const
- const ChannelLabel_T & getBookingChannel () const
- const TripType_T & getTripType () const
- const DayDuration_T & getStayDuration () const
- const FrequentFlyer_T & getFrequentFlyerType () const
- const WTP_T & getWTP () const
- const PriceValue_T & getValueOfTime () const
- const ChangeFees_T & getChangeFees () const
- const Disutility_T & getChangeFeeDisutility () const
- const NonRefundable_T & getNonRefundable () const
- const Disutility_T & getNonRefundableDisutility () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string describe () const
- const std::string display () const
- BookingRequestStruct (const DemandGeneratorKey_T &iGeneratorKey, const AirportCode_T &iOrigin, const AirportCode_T &iDestination, const CityCode_T &iPOS, const Date_T &iDepartureDate, const DateTime_T &iRequestDateTime, const CabinCode_T &iPreferredCabin, const NbOfSeats_T &iPartySize, const ChannelLabel_T &iChannel, const TripType_T &iTripType, const DayDuration_T &iStayDuration, const FrequentFlyer_T &iFrequentFlyerType, const Duration_T &iPreferredDepartureTime, const WTP_T &iWTP, const PriceValue_T &iValueOfTime, const ChangeFees_T &iChangeFees, const Disutility_T &iChangeFeeDisutility, const NonRefundable_T &iNonRefundable, const Disutility_T &iNonRefundableDisutility)
- BookingRequestStruct (const AirportCode_T &iOrigin, const AirportCode_T &iDestination, const CityCode_T &iPOS, const Date_T &iDepartureDate, const DateTime_T &iRequestDateTime, const CabinCode_T &iPreferredCabin, const NbOfSeats_T &iPartySize, const ChannelLabel_T &iChannel, const TripType_T &iTripType, const DayDuration_T &iStayDuration, const FrequentFlyer_T &iFrequentFlyerType, const Duration_T &iPreferredDepartureTime, const WTP_T &iWTP, const PriceValue_T &iValueOfTime, const ChangeFees_T &iChangeFees, const Disutility_T &iChangeFeeDisutility, const NonRefundable_T &iNonRefundable, const Disutility_T &iNonRefundableDisutility)
- BookingRequestStruct (const BookingRequestStruct &)
- ~BookingRequestStruct ()

8.30.1 Detailed Description

Structure holding the elements of a booking request.

Definition at line 21 of file BookingRequestStruct.hpp.

8.30.2 Constructor & Destructor Documentation

8.30.2.1 stdair::BookingRequestStruct::BookingRequestStruct (const DemandGeneratorKey_T & *iGeneratorKey*, const AirportCode_T & *iOrigin*, const AirportCode_T & *iDestination*, const CityCode_T & *iPOS*, const Date_T & *iDepartureDate*, const DateTime_T & *iRequestDateTime*, const

CabinCode_T & *iPreferredCabin*, const NbOfSeats_T & *iPartySize*, const ChannelLabel_T & *iChannel*, const TripType_T & *iTripType*, const DayDuration_T & *iStayDuration*, const FrequentFlyer_T & *iFrequentFlyerType*, const Duration_T & *iPreferredDepartureTime*, const WTP_T & *iWTP*, const PriceValue_T & *iValueOfTime*, const ChangeFees_T & *iChangeFees*, const Disutility_T & *iChangeFeeDisutility*, const NonRefundable_T & *iNonRefundable*, const Disutility_T & *iNonRefundableDisutility*)

Default constructor.

Definition at line 63 of file BookingRequestStruct.cpp.

8.30.2.2 stdair::BookingRequestStruct::BookingRequestStruct (const AirportCode_T & *iOrigin*, const AirportCode_T & *iDestination*, const CityCode_T & *iPOS*, const Date_T & *iDepartureDate*, const DateTime_T & *iRequestDateTime*, const CabinCode_T & *iPreferredCabin*, const NbOfSeats_T & *iPartySize*, const ChannelLabel_T & *iChannel*, const TripType_T & *iTripType*, const DayDuration_T & *iStayDuration*, const FrequentFlyer_T & *iFrequentFlyerType*, const Duration_T & *iPreferredDepartureTime*, const WTP_T & *iWTP*, const PriceValue_T & *iValueOfTime*, const ChangeFees_T & *iChangeFees*, const Disutility_T & *iChangeFeeDisutility*, const NonRefundable_T & *iNonRefundable*, const Disutility_T & *iNonRefundableDisutility*)

Constructor without the demand generator key, used for batches.

Definition at line 98 of file BookingRequestStruct.cpp.

8.30.2.3 stdair::BookingRequestStruct::BookingRequestStruct (const BookingRequestStruct &)

Copy constructor.

Definition at line 39 of file BookingRequestStruct.cpp.

8.30.2.4 stdair::BookingRequestStruct::~BookingRequestStruct ()

Destructor.

Definition at line 131 of file BookingRequestStruct.cpp.

8.30.3 Member Function Documentation

8.30.3.1 const DemandGeneratorKey_T& stdair::BookingRequestStruct::getDemandGeneratorKey () const [inline]

Get the demand generator key.

Definition at line 25 of file BookingRequestStruct.hpp.

8.30.3.2 const AirportCode_T& stdair::BookingRequestStruct::getOrigin () const [inline]

Get the requested origin.

Definition at line 30 of file BookingRequestStruct.hpp.

Referenced by stdair::BomJSONExport::jsonExportBookingRequestObject().

8.30.3.3 const AirportCode_T& stdair::BookingRequestStruct::getDestination () const [inline]

Get the requested destination.

Definition at line 35 of file BookingRequestStruct.hpp.

Referenced by stdair::BomJSONExport::jsonExportBookingRequestObject().

8.30.3.4 const CityCode_T& stdair::BookingRequestStruct::getPOS () const [inline]

Get the point-of-sale.

Definition at line 40 of file BookingRequestStruct.hpp.

Referenced by stdair::BomJSONExport::jsonExportBookingRequestObject().

8.30.3.5 const Date_T& stdair::BookingRequestStruct::getPreferedDepartureDate () const [inline]

Get the requested departure date.

Definition at line 45 of file BookingRequestStruct.hpp.

Referenced by stdair::BomJSONExport::jsonExportBookingRequestObject().

8.30.3.6 const Duration_T& stdair::BookingRequestStruct::getPreferredDepartureTime () const [inline]

Get the preferred departure time.

Definition at line 50 of file BookingRequestStruct.hpp.

Referenced by stdair::BomJSONExport::jsonExportBookingRequestObject().

8.30.3.7 const DateTime_T& stdair::BookingRequestStruct::getRequestDateTime () const [inline]

Get the request datetime.

Definition at line 55 of file BookingRequestStruct.hpp.

Referenced by stdair::BomJSONExport::jsonExportBookingRequestObject().

8.30.3.8 const CabinCode_T& stdair::BookingRequestStruct::getPreferredCabin () const [inline]

Get the preferred cabin.

Definition at line 60 of file BookingRequestStruct.hpp.

Referenced by stdair::BomJSONExport::jsonExportBookingRequestObject().

8.30.3.9 const NbOfSeats_T& stdair::BookingRequestStruct::getPartySize () const [inline]

Get the party size.

Definition at line 65 of file BookingRequestStruct.hpp.

Referenced by stdair::BomJSONExport::jsonExportBookingRequestObject().

8.30.3.10 const ChannelLabel_T& stdair::BookingRequestStruct::getBookingChannel () const [inline]

Get the reservation channel.

Definition at line 70 of file BookingRequestStruct.hpp.

Referenced by stdair::BomJSONExport::jsonExportBookingRequestObject().

8.30.3.11 const TripType_T& stdair::BookingRequestStruct::getTripType () const [inline]

Get the trip type.

Definition at line 75 of file BookingRequestStruct.hpp.

8.30.3.12 const DayDuration_T& stdair::BookingRequestStruct::getStayDuration () const [inline]

Get the duration of stay.

Definition at line 80 of file BookingRequestStruct.hpp.

Referenced by stdair::BomJSONExport::jsonExportBookingRequestObject().

8.30.3.13 const FrequentFlyer_T& stdair::BookingRequestStruct::getFrequentFlyerType () const [inline]

Get the frequent flyer type.

Definition at line 85 of file BookingRequestStruct.hpp.

8.30.3.14 const WTP_T& stdair::BookingRequestStruct::getWTP () const [inline]

Get the willingness-to-pay.

Definition at line 90 of file BookingRequestStruct.hpp.

Referenced by stdair::BomJSONExport::jsonExportBookingRequestObject().

8.30.3.15 const PriceValue_T& stdair::BookingRequestStruct::getValueOfTime () const [inline]

Get the value of time.

Definition at line 95 of file BookingRequestStruct.hpp.

8.30.3.16 const ChangeFees_T& stdair::BookingRequestStruct::getChangeFees () const [inline]

Get the change fee acceptation.

Definition at line 100 of file BookingRequestStruct.hpp.

8.30.3.17 const Disutility_T& stdair::BookingRequestStruct::getChangeFeeDisutility () const [inline]

Get the change disutility.

Definition at line 105 of file BookingRequestStruct.hpp.

8.30.3.18 const NonRefundable_T& stdair::BookingRequestStruct::getNonRefundable () const [inline]

Get the non refundable acceptation.

Definition at line 110 of file BookingRequestStruct.hpp.

8.30.3.19 const Disutility_T& stdair::BookingRequestStruct::getNonRefundableDisutility () const [inline]

Get the non refundable disutility.

Definition at line 115 of file BookingRequestStruct.hpp.

8.30.3.20 void stdair::BookingRequestStruct::toStream (std::ostream & *ioOut*) const

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::StructAbstract.

Definition at line 135 of file BookingRequestStruct.cpp.

References describe().

8.30.3.21 void stdair::BookingRequestStruct::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::StructAbstract.

Definition at line 140 of file BookingRequestStruct.cpp.

8.30.3.22 const std::string stdair::BookingRequestStruct::describe () const [virtual]

Display of the structure.

Implements stdair::StructAbstract.

Definition at line 144 of file BookingRequestStruct.cpp.

Referenced by toStream().

8.30.3.23 const std::string stdair::BookingRequestStruct::display () const

Display of the structure.

- id,
- request_date (YYMMDD),
- request_time (HHMMSS),

- POS (three-letter code),
- Channel (two-letter code):
  - 'D' for direct or 'I' for indirect,
  - 'N' for oNline or 'F' for oFfline,
- Origin (three-letter code),
- Destination (three-letter code),
- Preferred departure date (YYMMDD),
- Preferred departure time (HHMM),
- Min departure time (HHMM),
- Max departure time (HHMM),
- Preferred arrival date (YYMMDD),
- Preferred arrival time (HHMM),
- Preferred cabin:
  - 'F' for first,
  - 'C' for club/business,
  - 'W' for economy plus,
  - 'M' for economy,
- Trip type:
  - 'OW' for a one-way trip,
  - 'RO' for the outbound part of a rount-trip,
  - 'RI' for the inbound part of a rount-trip,
- Duration of stay (expressed as a number of days),
- Frequent flyer tier:
  - 'G' for gold,
  - 'S' for silver,
  - 'K' for basic,
  - 'N' for none,
- Willingness-to-pay (WTP, expressed as a monetary unit, e.g., EUR),
- Disutility per stop (expressed as a monetary unit, e.g., EUR),
- Value of time (EUR per hour),

Returns:

const std::string The output of the booking request structure.

Definition at line 169 of file BookingRequestStruct.cpp.

References stdair::TRIP_TYPE_ONE_WAY.

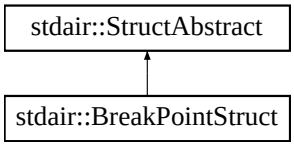
The documentation for this struct was generated from the following files:

- stdair/bom/BookingRequestStruct.hpp
- stdair/bom/BookingRequestStruct.cpp

8.31 stdair::BreakPointStruct Struct Reference

`#include <stdair/bom/BreakPointStruct.hpp>`

Inheritance diagram for stdair::BreakPointStruct::



Public Member Functions

- const DateTime_T & getBreakPointTime () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string describe () const
- BreakPointStruct (const DateTime_T &)
- BreakPointStruct (const Date_T &)
- BreakPointStruct (const BreakPointStruct &)
- ~BreakPointStruct ()

8.31.1 Detailed Description

Structure holding the elements of a break point.

Definition at line 18 of file BreakPointStruct.hpp.

8.31.2 Constructor & Destructor Documentation

8.31.2.1 stdair::BreakPointStruct::BreakPointStruct (const DateTime_T &)

Constructor.

Definition at line 26 of file BreakPointStruct.cpp.

8.31.2.2 stdair::BreakPointStruct::BreakPointStruct (const Date_T &)

Constructor.

Definition at line 32 of file BreakPointStruct.cpp.

8.31.2.3 stdair::BreakPointStruct::BreakPointStruct (const BreakPointStruct &)

Copy constructor.

Definition at line 20 of file BreakPointStruct.cpp.

8.31.2.4 stdair::BreakPointStruct::~BreakPointStruct ()

Destructor.

Definition at line 37 of file BreakPointStruct.cpp.

8.31.3 Member Function Documentation

8.31.3.1 const DateTime_T& stdair::BreakPointStruct::getBreakPointTime () const [inline]

Get the break point action time.

Definition at line 22 of file BreakPointStruct.hpp.

Referenced by stdair::BomJSONExport::jsonExportBreakPointObject().

8.31.3.2 void stdair::BreakPointStruct::toStream (std::ostream & *ioOut*) const

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::StructAbstract.

Definition at line 41 of file BreakPointStruct.cpp.

References describe().

8.31.3.3 void stdair::BreakPointStruct::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::StructAbstract.

Definition at line 46 of file BreakPointStruct.cpp.

8.31.3.4 const std::string stdair::BreakPointStruct::describe () const [virtual]

Display of the structure.

Implements stdair::StructAbstract.

Definition at line 50 of file BreakPointStruct.cpp.

Referenced by toStream().

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

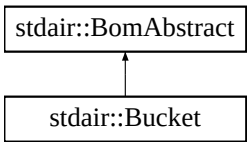
- stdair/bom/BreakPointStruct.hpp
- stdair/bom/BreakPointStruct.cpp

8.32 stdair::Bucket Class Reference

Class representing the actual attributes for an airline booking class.

`#include <stdair/bom/Bucket.hpp>`

Inheritance diagram for stdair::Bucket::



Public Types

- typedef BucketKey Key_T

Public Member Functions

- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const HolderMap_T & getHolderMap () const
- const SeatIndex_T & getSeatIndex () const
- const Yield_T & getYieldRangeUpperValue () const
- const CabinCapacity_T & getAvailability () const
- const NbOfSeats_T & getSoldSeats () const
- void setYieldRangeUpperValue (const Yield_T &iYield)
- void setAvailability (const CabinCapacity_T &iAvl)
- void setSoldSeats (const NbOfSeats_T &iSoldSeats)
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- template<class Archive> void serialize (Archive &ar, const unsigned int iFileVersion)

Protected Member Functions

- Bucket (const Key_T &)
- virtual ~Bucket ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent
- HolderMap_T _holderMap
- Yield_T _yieldRangeUpperValue
- CabinCapacity_T _availability
- NbOfSeats_T _soldSeats

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager
- class boost::serialization::access

8.32.1 Detailed Description

Class representing the actual attributes for an airline booking class.

Definition at line 29 of file Bucket.hpp.

8.32.2 Member Typedef Documentation

8.32.2.1 typedef BucketKey stdair::Bucket::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 40 of file Bucket.hpp.

8.32.3 Constructor & Destructor Documentation

8.32.3.1 stdair::Bucket::Bucket (const Key_T &) [protected]

Default constructor.

Definition at line 34 of file Bucket.cpp.

8.32.3.2 stdair::Bucket::~Bucket () [protected, virtual]

Destructor.

Definition at line 38 of file Bucket.cpp.

8.32.4 Member Function Documentation

8.32.4.1 const Key_T& stdair::Bucket::getKey () const [inline]

Get the primary key of the bucket.

Definition at line 47 of file Bucket.hpp.

References _key.

8.32.4.2 BomAbstract* const stdair::Bucket::getParent () const [inline]

Get the parent object.

Definition at line 54 of file Bucket.hpp.

References _parent.

8.32.4.3 const HolderMap_T& stdair::Bucket::getHolderMap () const [inline]

Get the map of children holders.

Definition at line 59 of file Bucket.hpp.

References _holderMap.

8.32.4.4 const SeatIndex_T& stdair::Bucket::getSeatIndex () const [inline]

Get the seat index (part of the primary key).

Definition at line 64 of file Bucket.hpp.

References _key, and stdair::BucketKey::getSeatIndex().

8.32.4.5 const Yield_T& stdair::Bucket::getYieldRangeUpperValue () const [inline]

Get the upper yield range.

Definition at line 69 of file Bucket.hpp.

References _yieldRangeUpperValue.

8.32.4.6 const CabinCapacity_T& stdair::Bucket::getAvailability () const [inline]

Get the availability.

Definition at line 74 of file Bucket.hpp.

References _availability.

8.32.4.7 const NbOfSeats_T& stdair::Bucket::getSoldSeats () const [inline]

Get the number of seats already sold.

Definition at line 79 of file Bucket.hpp.

References _soldSeats.

8.32.4.8 void stdair::Bucket::setYieldRangeUpperValue (const Yield_T & *iYield*) [inline]

Set the upper yield range.

Definition at line 86 of file Bucket.hpp.

References _yieldRangeUpperValue.

8.32.4.9 void stdair::Bucket::setAvailability (const CabinCapacity_T & *iAvl*) [inline]

Set the availability.

Definition at line 91 of file Bucket.hpp.

References _availability.

8.32.4.10 void stdair::Bucket::setSoldSeats (const NbOfSeats_T & *iSoldSeats*) [inline]

Set the number of seats already sold.

Definition at line 96 of file Bucket.hpp.

References _soldSeats.

8.32.4.11 void stdair::Bucket::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 108 of file Bucket.hpp.

References toString().

8.32.4.12 void stdair::Bucket::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 117 of file Bucket.hpp.

8.32.4.13 std::string stdair::Bucket::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 42 of file Bucket.cpp.

References describeKey().

Referenced by toStream().

8.32.4.14 const std::string stdair::Bucket::describeKey () const [inline]

Get a string describing the key.

Definition at line 128 of file Bucket.hpp.

References _key, and stdair::BucketKey::toString().

Referenced by toString().

8.32.4.15 template<class Archive> void stdair::Bucket::serialize (Archive & *ar*, const unsigned int *iFileVersion*)

Serialisation.

Definition at line 64 of file Bucket.cpp.

References _key.

8.32.5 Friends And Related Function Documentation

8.32.5.1 friend class FacBom [friend]

Definition at line 30 of file Bucket.hpp.

8.32.5.2 friend class FacCloneBom [friend]

Definition at line 31 of file Bucket.hpp.

8.32.5.3 friend class FacBomManager [friend]

Definition at line 32 of file Bucket.hpp.

8.32.5.4 friend class boost::serialization::access [friend]

Definition at line 33 of file Bucket.hpp.

8.32.6 Member Data Documentation

8.32.6.1 Key_T stdair::Bucket::_key [protected]

Primary key (upper yield range).

Definition at line 179 of file Bucket.hpp.

Referenced by describeKey(), getKey(), getSeatIndex(), and serialize().

8.32.6.2 BomAbstract* stdair::Bucket::_parent [protected]

Pointer on the parent class (LegCabin).

Definition at line 184 of file Bucket.hpp.

Referenced by getParent().

8.32.6.3 HolderMap_T stdair::Bucket::_holderMap [protected]

Map holding the children (empty for now).

Definition at line 189 of file Bucket.hpp.

Referenced by getHolderMap().

8.32.6.4 Yield_T stdair::Bucket::_yieldRangeUpperValue [protected]

Upper yield range.

Definition at line 197 of file Bucket.hpp.

Referenced by getYieldRangeUpperValue(), and setYieldRangeUpperValue().

8.32.6.5 CabinCapacity_T stdair::Bucket::_availability [protected]

Availability.

Definition at line 202 of file Bucket.hpp.

Referenced by getAvailability(), and setAvailability().

8.32.6.6 NbOfSeats_T stdair::Bucket::_soldSeats [protected]

Number of seats already sold.

Definition at line 207 of file Bucket.hpp.

Referenced by getSoldSeats(), and setSoldSeats().

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

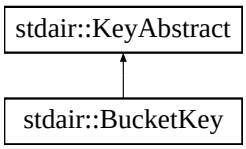
- stdair/bom/Bucket.hpp
- stdair/bom/Bucket.cpp

8.33 stdair::BucketKey Struct Reference

Key of booking-class.

`#include <stdair/bom/BucketKey.hpp>`

Inheritance diagram for stdair::BucketKey::



Public Member Functions

- BucketKey (const SeatIndex_T &)
- BucketKey (const BucketKey &)
- ~BucketKey ()
- const SeatIndex_T & getSeatIndex () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const
- template<class Archive> void serialize (Archive &ar, const unsigned int iFileVersion)

Friends

- class boost::serialization::access

8.33.1 Detailed Description

Key of booking-class.

Definition at line 26 of file BucketKey.hpp.

8.33.2 Constructor & Destructor Documentation

8.33.2.1 stdair::BucketKey::BucketKey (const SeatIndex_T &)

Main constructor.

Definition at line 22 of file BucketKey.cpp.

8.33.2.2 stdair::BucketKey::BucketKey (const BucketKey &)

Copy constructor.

Definition at line 27 of file BucketKey.cpp.

8.33.2.3 stdair::BucketKey::~BucketKey ()

Destructor.

Definition at line 32 of file BucketKey.cpp.

8.33.3 Member Function Documentation

8.33.3.1 const SeatIndex_T& stdair::BucketKey::getSeatIndex () const [inline]

Get the seat index.

Definition at line 54 of file BucketKey.hpp.

Referenced by stdair::Bucket::getSeatIndex().

8.33.3.2 void stdair::BucketKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 36 of file BucketKey.cpp.

References toString().

8.33.3.3 void stdair::BucketKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 41 of file BucketKey.cpp.

8.33.3.4 const std::string stdair::BucketKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-cabin.

Reimplemented from stdair::KeyAbstract.

Definition at line 45 of file BucketKey.cpp.

Referenced by stdair::Bucket::describeKey(), and toStream().

8.33.3.5 template<class Archive> void stdair::BucketKey::serialize (Archive & *ar*, const unsigned int *iFileVersion*)

Serialisation.

Definition at line 67 of file BucketKey.cpp.

8.33.4 Friends And Related Function Documentation

8.33.4.1 friend class boost::serialization::access [friend]

Definition at line 27 of file BucketKey.hpp.

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

- stdair/bom/BucketKey.hpp
- stdair/bom/BucketKey.cpp

8.34 stdair_test::Cabin Struct Reference

`#include <test/stdair/StdairTestLib.hpp>`

Public Types

- typedef BookingClass child

Public Member Functions

- Cabin (const BookingClass &iBkgClass)
- std::string toString () const

Public Attributes

- BookingClass _bookingClass

8.34.1 Detailed Description

Cabin

Definition at line 32 of file StdairTestLib.hpp.

8.34.2 Member Typedef Documentation

8.34.2.1 typedef BookingClass stdair_test::Cabin::child

Child type.

Definition at line 46 of file StdairTestLib.hpp.

8.34.3 Constructor & Destructor Documentation

8.34.3.1 stdair_test::Cabin::Cabin (const BookingClass & *iBkgClass*) [inline]

Definition at line 34 of file StdairTestLib.hpp.

8.34.4 Member Function Documentation

8.34.4.1 std::string stdair_test::Cabin::toString () const [inline]

Display .

Definition at line 39 of file StdairTestLib.hpp.

References _bookingClass, and stdair_test::BookingClass::_classCode.

8.34.5 Member Data Documentation

8.34.5.1 BookingClass stdair_test::Cabin::_bookingClass

Definition at line 33 of file StdairTestLib.hpp.

Referenced by toString().

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

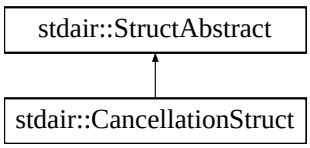
- test/stdair/StdairTestLib.hpp

8.35 stdair::CancellationStruct Struct Reference

Structure holding the elements of a travel solution.

`#include <stdair/bom/CancellationStruct.hpp>`

Inheritance diagram for stdair::CancellationStruct::



Public Member Functions

- const SegmentPath_T & getSegmentPath () const
- const ClassList_String_T & getClassList () const
- const BookingClassIDList_T & getClassIDList () const
- const PartySize_T & getPartySize () const
- const DateTime_T & getCancellationDateTime () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string describe () const
- const std::string display () const

- CancellationStruct (const SegmentPath_T &, const ClassList_String_T &, const PartySize_T &, const DateTime_T &)
- CancellationStruct (const SegmentPath_T &, const BookingClassIDList_T &, const PartySize_T &, const DateTime_T &)
- ~CancellationStruct ()

8.35.1 Detailed Description

Structure holding the elements of a travel solution.

Definition at line 23 of file CancellationStruct.hpp.

8.35.2 Constructor & Destructor Documentation

8.35.2.1 stdair::CancellationStruct::CancellationStruct (const SegmentPath_T &, const ClassList_String_T &, const PartySize_T &, const DateTime_T &)

Default constructor without class ID list.

Definition at line 14 of file CancellationStruct.cpp.

8.35.2.2 stdair::CancellationStruct::CancellationStruct (const SegmentPath_T &, const BookingClassIDList_T &, const PartySize_T &, const DateTime_T &)

Default constructor with class ID list.

Definition at line 23 of file CancellationStruct.cpp.

8.35.2.3 stdair::CancellationStruct::~CancellationStruct ()

Destructor.

Definition at line 32 of file CancellationStruct.cpp.

8.35.3 Member Function Documentation

8.35.3.1 const SegmentPath_T& stdair::CancellationStruct::getSegmentPath () const [inline]

Get the segment path.

Definition at line 27 of file CancellationStruct.hpp.

8.35.3.2 const ClassList_String_T& stdair::CancellationStruct::getClassList () const [inline]

Get the class list.

Definition at line 32 of file CancellationStruct.hpp.

8.35.3.3 const BookingClassIDList_T& stdair::CancellationStruct::getClassIDList () const [inline]

Get the class ID list.

Definition at line 37 of file CancellationStruct.hpp.

8.35.3.4 const PartySize_T& stdair::CancellationStruct::getPartySize () const [inline]

Get the party size.

Definition at line 42 of file CancellationStruct.hpp.

8.35.3.5 const DateTime_T& stdair::CancellationStruct::getCancellationDateTime () const [inline]

Get the datetime.

Definition at line 47 of file CancellationStruct.hpp.

8.35.3.6 void stdair::CancellationStruct::toStream (std::ostream & *ioOut*) const

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::StructAbstract.

Definition at line 36 of file CancellationStruct.cpp.

References describe().

8.35.3.7 void stdair::CancellationStruct::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::StructAbstract.

Definition at line 41 of file CancellationStruct.cpp.

8.35.3.8 const std::string stdair::CancellationStruct::describe () const [virtual]

Display of the structure.

Implements stdair::StructAbstract.

Definition at line 45 of file CancellationStruct.cpp.

References stdair::BookingClass::getClassCode(), and stdair::BomID< BOM >::getObject().

Referenced by toStream().

8.35.3.9 const std::string stdair::CancellationStruct::display () const

Display of the structure.

Definition at line 81 of file CancellationStruct.cpp.

References stdair::BookingClass::getClassCode(), and stdair::BomID< BOM >::getObject().

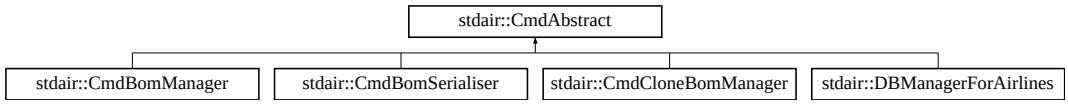
The documentation for this struct was generated from the following files:

- stdair/bom/CancellationStruct.hpp
- stdair/bom/CancellationStruct.cpp

8.36 stdair::CmdAbstract Class Reference

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

Inheritance diagram for stdair::CmdAbstract::



8.36.1 Detailed Description

Base class for the Command layer.

Definition at line 11 of file CmdAbstract.hpp.

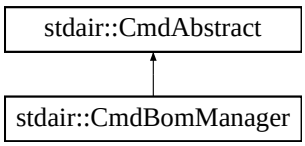
The documentation for this class was generated from the following file:

- stdair/command/CmdAbstract.hpp

8.37 stdair::CmdBomManager Class Reference

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

Inheritance diagram for stdair::CmdBomManager::



Friends

- class STDAIR_Service

8.37.1 Detailed Description

Class wrapping utility functions for handling the BOM tree objects.

Definition at line 25 of file CmdBomManager.hpp.

8.37.2 Friends And Related Function Documentation

8.37.2.1 friend class STDAIR_Service [friend]

Definition at line 27 of file CmdBomManager.hpp.

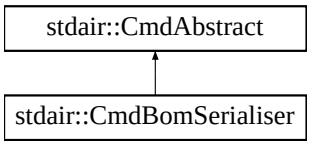
The documentation for this class was generated from the following file:

- stdair/command/CmdBomManager.hpp

8.38 stdair::CmdBomSerialiser Class Reference

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

Inheritance diagram for stdair::CmdBomSerialiser::



8.38.1 Detailed Description

Class wrapping utility functions for handling the BOM tree objects.

Definition at line 25 of file CmdBomSerialiser.hpp.

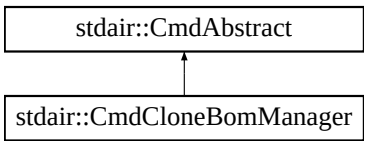
The documentation for this class was generated from the following file:

- stdair/command/CmdBomSerialiser.hpp

8.39 stdair::CmdCloneBomManager Class Reference

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

Inheritance diagram for stdair::CmdCloneBomManager::



Friends

- class STDAIR_Service

8.39.1 Detailed Description

Class wrapping utility functions for handling the BOM tree objects.

Definition at line 40 of file CmdCloneBomManager.hpp.

8.39.2 Friends And Related Function Documentation

8.39.2.1 friend class STDAIR_Service [friend]

Definition at line 42 of file CmdCloneBomManager.hpp.

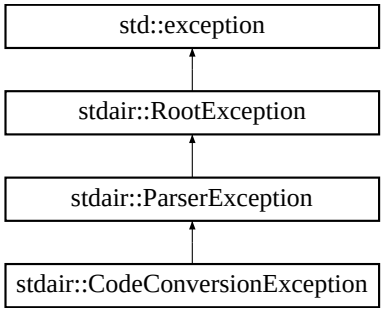
The documentation for this class was generated from the following file:

- stdair/command/CmdCloneBomManager.hpp

8.40 stdair::CodeConversionException Class Reference

`#include <stdair/stdair_exceptions.hpp>`

Inheritance diagram for stdair::CodeConversionException::



Public Member Functions

- CodeConversionException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

8.40.1 Detailed Description

Code conversion.

Definition at line 133 of file stdair_exceptions.hpp.

8.40.2 Constructor & Destructor Documentation

8.40.2.1 stdair::CodeConversionException::CodeConversionException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 136 of file stdair_exceptions.hpp.

8.40.3 Member Function Documentation

8.40.3.1 const char* stdair::RootException::what () const throw () [inline, inherited]

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

8.40.4 Member Data Documentation

8.40.4.1 std::string stdair::RootException::_what [protected, inherited]

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

Referenced by stdair::RootException::what().

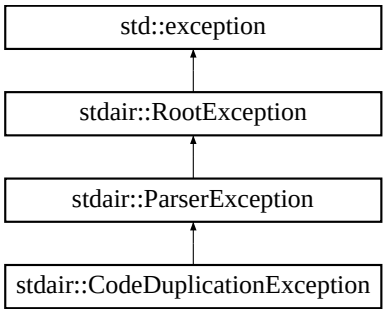
The documentation for this class was generated from the following file:

- stdair/stdair_exceptions.hpp

8.41 stdair::CodeDuplicationException Class Reference

`#include <stdair/stdair_exceptions.hpp>`

Inheritance diagram for stdair::CodeDuplicationException::



Public Member Functions

- CodeDuplicationException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

8.41.1 Detailed Description

Code duplication.

Definition at line 141 of file stdair_exceptions.hpp.

8.41.2 Constructor & Destructor Documentation

8.41.2.1 stdair::CodeDuplicationException::CodeDuplicationException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 144 of file stdair_exceptions.hpp.

8.41.3 Member Function Documentation

8.41.3.1 const char* stdair::RootException::what () const throw () [inline, inherited]

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

8.41.4 Member Data Documentation

8.41.4.1 std::string stdair::RootException::_what [protected, inherited]

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

Referenced by stdair::RootException::what().

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

- stdair/stdair_exceptions.hpp

8.42 COMMAND Struct Reference

`#include <stdair/ui/cmdline/readline_autocomp.hpp>`

Public Attributes

- char const * name
- pt2Func * func
- char * doc

8.42.1 Detailed Description

A structure which contains information on the commands this program can understand.

Definition at line 41 of file readline_autocomp.hpp.

8.42.2 Member Data Documentation

8.42.2.1 char const* COMMAND::name

User printable name of the function.

Definition at line 45 of file readline_autocomp.hpp.

Referenced by com_help(), and find_command().

8.42.2.2 pt2Func* COMMAND::func

Function to call to do the job.

Definition at line 50 of file readline_autocomp.hpp.

Referenced by execute_line().

8.42.2.3 char* COMMAND::doc

Documentation for this function.

Definition at line 55 of file readline_autocomp.hpp.

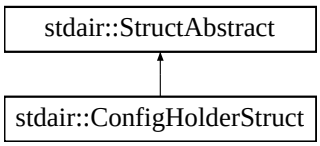
The documentation for this struct was generated from the following file:

- stdair/ui/cmdline/readline_autocomp.hpp

8.43 stdair::ConfigHolderStruct Struct Reference

`#include <stdair/bom/ConfigHolderStruct.hpp>`

Inheritance diagram for stdair::ConfigHolderStruct::



Public Member Functions

- void add (const bpt::ptree &)
- bool addValue (const std::string &iValue, const std::string &iPath)
- template<typename ValueType> bool exportValue (ValueType &ioValue, const std::string &iPath) const
- void updateAirlineFeatures (BomRoot &)
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string describe () const
- const std::string jsonExport () const
- ConfigHolderStruct ()
- ConfigHolderStruct (const ConfigHolderStruct &)
- ~ConfigHolderStruct ()
- template<> bool exportValue (Date_T &ioValue, const std::string &iPath) const

8.43.1 Detailed Description

Structure holding the configuration of the simulation.

Definition at line 40 of file ConfigHolderStruct.hpp.

8.43.2 Constructor & Destructor Documentation

8.43.2.1 stdair::ConfigHolderStruct::ConfigHolderStruct ()

Constructor.

Definition at line 27 of file ConfigHolderStruct.cpp.

8.43.2.2 stdair::ConfigHolderStruct::ConfigHolderStruct (const ConfigHolderStruct &)

Copy constructor.

Definition at line 32 of file ConfigHolderStruct.cpp.

8.43.2.3 stdair::ConfigHolderStruct::~ConfigHolderStruct ()

Destructor.

Definition at line 37 of file ConfigHolderStruct.cpp.

8.43.3 Member Function Documentation

8.43.3.1 void stdair::ConfigHolderStruct::add (const bpt::ptree &)

Merge the given property tree with the existing configuration property tree gathering all the configuration information.

Parameters:

const bpt::ptree& Property tree to add to the configuration tree.

Definition at line 144 of file ConfigHolderStruct.cpp.

Referenced by stdair::BomINIImport::importINIConfig(), and stdair::BomJSONImport::jsonImportConfig().

8.43.3.2 bool stdair::ConfigHolderStruct::addValue (const std::string & *iValue*, const std::string & *iPath*)

Create the given specified path in the configuration tree and add the corresponding given value (or replace the value if the path already exists).

Parameters:

const std::string& Value to add at the given path.

const std::string& Path to create (or to look for).

Definition at line 191 of file ConfigHolderStruct.cpp.

Referenced by stdair::STDAIR_Service::importConfigValue().

8.43.3.3 template<typename ValueType> bool stdair::ConfigHolderStruct::exportValue (ValueType & *ioValue*, const std::string & *iPath*) const

Look for the specified path in the configuration tree and, if existing, try to extract the corresponding value. The type of the value to extract is a template parameter.

Parameters:

ValueType& Value to add in the configuration tree.

const std::string& Path to look for.

Definition at line 144 of file ConfigHolderStruct.hpp.

Referenced by stdair::STDAIR_Service::exportConfigValue().

8.43.3.4 void stdair::ConfigHolderStruct::updateAirlineFeatures (BomRoot &)

Update the airline features objects thanks to the configuration holder.

Parameters:

BomRoot& Reference on the BomRoot to update.

Definition at line 220 of file ConfigHolderStruct.cpp.

References stdair::BomRetriever::retrieveAirlineFeatureFromKey(), stdair::AirlineFeature::setForecastingMethod(), stdair::AirlineFeature::setOptimisationMethod(), stdair::AirlineFeature::setPartnershipTechnique(), stdair::AirlineFeature::setPreOptimisationMethod(), stdair::AirlineFeature::setUnconstrainingMethod(), STDAIR_LOG_ERROR, and stdair::RootException::what().

Referenced by stdair::STDAIR_Service::updateAirlineFeatures().

8.43.3.5 void stdair::ConfigHolderStruct::toStream (std::ostream & *ioOut*) const

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::StructAbstract.

Definition at line 41 of file ConfigHolderStruct.cpp.

References describe().

8.43.3.6 void stdair::ConfigHolderStruct::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::StructAbstract.

Definition at line 46 of file ConfigHolderStruct.cpp.

8.43.3.7 const std::string stdair::ConfigHolderStruct::describe () const [virtual]

Display of the structure.

Implements stdair::StructAbstract.

Definition at line 50 of file ConfigHolderStruct.cpp.

Referenced by stdair::STDAIR_Service::configDisplay(), and toStream().

8.43.3.8 const std::string stdair::ConfigHolderStruct::jsonExport () const

Display of the configuration in a JSON-ified format.

Definition at line 134 of file ConfigHolderStruct.cpp.

Referenced by stdair::STDAIR_Service::jsonExportConfiguration().

8.43.3.9 template<> bool stdair::ConfigHolderStruct::exportValue (Date_T & *ioValue*, const std::string & *iPath*) const [inline]

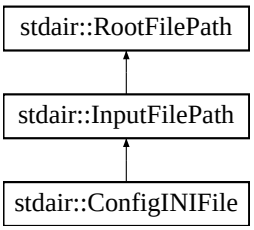
The documentation for this struct was generated from the following files:

- stdair/bom/ConfigHolderStruct.hpp
- stdair/bom/ConfigHolderStruct.cpp

8.44 stdair::ConfigINIFile Class Reference

`#include <stdair/stdair_file.hpp>`

Inheritance diagram for stdair::ConfigINIFile::



Public Member Functions

- ConfigINIFile (const Filename_T &iFilename)
- const char * name () const

Protected Attributes

- const Filename_T _filename

8.44.1 Detailed Description

Config file: INI format

Definition at line 112 of file stdair_file.hpp.

8.44.2 Constructor & Destructor Documentation

8.44.2.1 stdair::ConfigINIFile::ConfigINIFile (const Filename_T & *iFilename*) [inline, explicit]

Constructor.

Definition at line 117 of file stdair_file.hpp.

8.44.3 Member Function Documentation

8.44.3.1 const char* stdair::RootFilePath::name () const [inline, inherited]

Give the details of the exception.

Definition at line 42 of file stdair_file.hpp.

References stdair::RootFilePath::_filename.

Referenced by stdair::BomINIImport::importINIConfig().

8.44.4 Member Data Documentation

8.44.4.1 const Filename_T stdair::RootFilePath::_filename [protected, inherited]

Name of the file.

Definition at line 50 of file stdair_file.hpp.

Referenced by stdair::RootFilePath::name().

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

- stdair/stdair_file.hpp

8.45 stdair::ContinuousAttributeLite< T > Struct Template Reference

Class modeling the distribution of values that can be taken by a continuous attribute.

`#include <stdair/basic/ContinuousAttributeLite.hpp>`

Public Types

- typedef std::map< T, stdair::Probability_T > ContinuousDistribution_T

Public Member Functions

- const T getValue (const stdair::Probability_T &iCumulativeProbability) const
- const stdair::Probability_T getRemainingProportion (const T &iValue) const
- const double getDerivativeValue (const T iKey) const
- const T getUpperBound (const T iKey) const
- const std::string displayCumulativeDistribution () const
- ContinuousAttributeLite (const ContinuousDistribution_T &iValueMap)
- ContinuousAttributeLite (const ContinuousAttributeLite &iCAL)
- ContinuousAttributeLite & operator= (const ContinuousAttributeLite &iCAL)
- virtual ~ContinuousAttributeLite ()

8.45.1 Detailed Description

template<typename T> struct stdair::ContinuousAttributeLite< T >

Class modeling the distribution of values that can be taken by a continuous attribute.

Definition at line 26 of file ContinuousAttributeLite.hpp.

8.45.2 Member Typedef Documentation

8.45.2.1 template<typename T> typedef std::map<T, stdair::Probability_T> stdair::ContinuousAttributeLite< T >::ContinuousDistribution_T

Type for the probability mass function.

Definition at line 32 of file ContinuousAttributeLite.hpp.

8.45.3 Constructor & Destructor Documentation

8.45.3.1 template<typename T> stdair::ContinuousAttributeLite< T >::ContinuousAttributeLite (const ContinuousDistribution_T & *iValueMap*) [inline]

Constructor.

Definition at line 204 of file ContinuousAttributeLite.hpp.

8.45.3.2 template<typename T> stdair::ContinuousAttributeLite< T >::ContinuousAttributeLite (const ContinuousAttributeLite< T > & *iCAL*) [inline]

Copy constructor.

Definition at line 212 of file ContinuousAttributeLite.hpp.

8.45.3.3 template<typename T> virtual stdair::ContinuousAttributeLite< T >::~ContinuousAttributeLite () [inline, virtual]

Destructor.

Definition at line 231 of file ContinuousAttributeLite.hpp.

8.45.4 Member Function Documentation

8.45.4.1 template<typename T> const T stdair::ContinuousAttributeLite< T >::getValue (const stdair::Probability_T & *iCumulativeProbability*) const [inline]

Get value from inverse cumulative distribution.

Definition at line 39 of file ContinuousAttributeLite.hpp.

References stdair::DictionaryManager::keyToValue(), and stdair::DictionaryManager::valueToKey().

8.45.4.2 template<typename T> const stdair::Probability_T stdair::ContinuousAttributeLite< T >::getRemainingProportion (const T & *iValue*) const [inline]

Get remaining proportion from cumulative distribution.

Definition at line 84 of file ContinuousAttributeLite.hpp.

References stdair::DictionaryManager::keyToValue().

8.45.4.3 template<typename T> const double stdair::ContinuousAttributeLite< T >::getDerivativeValue (const T *iKey*) const [inline]

Get the value of the derivative function in a key point.

Definition at line 131 of file ContinuousAttributeLite.hpp.

References stdair::DictionaryManager::keyToValue().

8.45.4.4 template<typename T> const T stdair::ContinuousAttributeLite< T >::getUpperBound (const T *iKey*) const [inline]

Get the upper bound.

Definition at line 163 of file ContinuousAttributeLite.hpp.

8.45.4.5 template<typename T> const std::string stdair::ContinuousAttributeLite< T >::displayCumulativeDistribution () const [inline]

Display cumulative distribution.

Definition at line 182 of file ContinuousAttributeLite.hpp.

References stdair::DictionaryManager::keyToValue().

8.45.4.6 template<typename T> ContinuousAttributeLite& stdair::ContinuousAttributeLite< T >::operator= (const ContinuousAttributeLite< T > & *iCAL*) [inline]

Copy operator.

Definition at line 221 of file ContinuousAttributeLite.hpp.

References stdair::ContinuousAttributeLite< T >::_cumulativeDistribution, stdair::ContinuousAttributeLite< T >::_size, and stdair::ContinuousAttributeLite< T >::_valueArray.

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

- stdair/basic/ContinuousAttributeLite.hpp

8.46 stdair::date_time_element< MIN, MAX > Struct Template Reference

`#include <stdair/basic/BasParserHelperTypes.hpp>`

Public Member Functions

- date_time_element ()
- date_time_element (const date_time_element &t)
- date_time_element (int i)
- void check () const

Public Attributes

- unsigned int _value

8.46.1 Detailed Description

template<int MIN = 0, int MAX = 0> struct stdair::date_time_element< MIN, MAX >

Date & time element parser.

Definition at line 23 of file BasParserHelperTypes.hpp.

8.46.2 Constructor & Destructor Documentation

8.46.2.1 template<int MIN = 0, int MAX = 0> stdair::date_time_element< MIN, MAX >::date_time_element () [inline]

Default constructor.

Definition at line 28 of file BasParserHelperTypes.hpp.

8.46.2.2 template<int MIN = 0, int MAX = 0> stdair::date_time_element< MIN, MAX >::date_time_element (const date_time_element< MIN, MAX > & *t*) [inline]

Default copy constructor.

Definition at line 30 of file BasParserHelperTypes.hpp.

8.46.2.3 template<int MIN = 0, int MAX = 0> stdair::date_time_element< MIN, MAX >::date_time_element (int *i*) [inline]

Constructor.

Definition at line 32 of file BasParserHelperTypes.hpp.

8.46.3 Member Function Documentation

8.46.3.1 template<int MIN = 0, int MAX = 0> void stdair::date_time_element< MIN, MAX >::check () const [inline]

Checker.

Definition at line 34 of file BasParserHelperTypes.hpp.

References stdair::date_time_element< MIN, MAX >::_value.

8.46.4 Member Data Documentation

8.46.4.1 template<int MIN = 0, int MAX = 0> unsigned int stdair::date_time_element< MIN, MAX >::_value

Definition at line 24 of file BasParserHelperTypes.hpp.

Referenced by stdair::date_time_element< MIN, MAX >::check(), stdair::operator *(), and stdair::operator+().

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

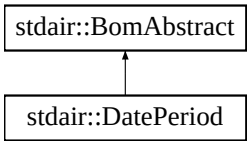
- stdair/basic/BasParserHelperTypes.hpp

8.47 stdair::DatePeriod Class Reference

Class representing the actual attributes for a fare date-period.

`#include <stdair/bom/DatePeriod.hpp>`

Inheritance diagram for stdair::DatePeriod::



Public Types

- typedef DatePeriodKey Key_T

Public Member Functions

- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const HolderMap_T & getHolderMap () const
- const DatePeriod_T & getDatePeriod () const
- bool isDepartureDateValid (const Date_T &) const

Protected Member Functions

- DatePeriod (const Key_T &)
- virtual ~DatePeriod ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent
- HolderMap_T _holderMap

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager

8.47.1 Detailed Description

Class representing the actual attributes for a fare date-period.

Definition at line 18 of file DatePeriod.hpp.

8.47.2 Member Typedef Documentation

8.47.2.1 typedef DatePeriodKey stdair::DatePeriod::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 28 of file DatePeriod.hpp.

8.47.3 Constructor & Destructor Documentation

8.47.3.1 stdair::DatePeriod::DatePeriod (const Key_T &) [protected]

Main constructor.

Definition at line 27 of file DatePeriod.cpp.

8.47.3.2 stdair::DatePeriod::~DatePeriod () [protected, virtual]

Destructor.

Definition at line 32 of file DatePeriod.cpp.

8.47.4 Member Function Documentation

8.47.4.1 void stdair::DatePeriod::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 37 of file DatePeriod.hpp.

References toString().

8.47.4.2 void stdair::DatePeriod::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 46 of file DatePeriod.hpp.

8.47.4.3 std::string stdair::DatePeriod::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 36 of file DatePeriod.cpp.

References describeKey().

Referenced by toStream().

8.47.4.4 const std::string stdair::DatePeriod::describeKey () const [inline]

Get a string describing the key.

Definition at line 57 of file DatePeriod.hpp.

References _key, and stdair::DatePeriodKey::toString().

Referenced by toString().

8.47.4.5 const Key_T& stdair::DatePeriod::getKey () const [inline]

Get the primary key (date period).

Definition at line 66 of file DatePeriod.hpp.

References _key.

8.47.4.6 BomAbstract* const stdair::DatePeriod::getParent () const [inline]

Get a reference on the parent object instance.

Definition at line 73 of file DatePeriod.hpp.

References _parent.

8.47.4.7 const HolderMap_T& stdair::DatePeriod::getHolderMap () const [inline]

Get a reference on the children holder.

Definition at line 80 of file DatePeriod.hpp.

References _holderMap.

8.47.4.8 const DatePeriod_T& stdair::DatePeriod::getDatePeriod () const [inline]

Get the date period.

Definition at line 87 of file DatePeriod.hpp.

References _key, and stdair::DatePeriodKey::getDatePeriod().

Referenced by isDepartureDateValid().

8.47.4.9 bool stdair::DatePeriod::isDepartureDateValid (const Date_T &) const

Check if the given departure date is included in the departure period of the segment path.

Definition at line 44 of file DatePeriod.cpp.

References getDatePeriod().

8.47.5 Friends And Related Function Documentation

8.47.5.1 friend class FacBom [friend]

Definition at line 19 of file DatePeriod.hpp.

8.47.5.2 friend class FacCloneBom [friend]

Definition at line 20 of file DatePeriod.hpp.

8.47.5.3 friend class FacBomManager [friend]

Definition at line 21 of file DatePeriod.hpp.

8.47.6 Member Data Documentation

8.47.6.1 Key_T stdair::DatePeriod::_key [protected]

Primary key (date period).

Definition at line 126 of file DatePeriod.hpp.

Referenced by describeKey(), getDatePeriod(), and getKey().

8.47.6.2 BomAbstract* stdair::DatePeriod::_parent [protected]

Pointer on the parent class.

Definition at line 131 of file DatePeriod.hpp.

Referenced by getParent().

8.47.6.3 HolderMap_T stdair::DatePeriod::_holderMap [protected]

Map holding the children.

Definition at line 136 of file DatePeriod.hpp.

Referenced by getHolderMap().

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

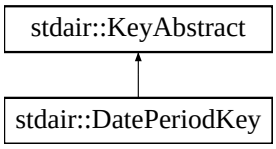
- stdair/bom/DatePeriod.hpp
- stdair/bom/DatePeriod.cpp

8.48 stdair::DatePeriodKey Struct Reference

Key of date-period.

`#include <stdair/bom/DatePeriodKey.hpp>`

Inheritance diagram for stdair::DatePeriodKey::



Public Member Functions

- DatePeriodKey (const DatePeriod_T &)
- DatePeriodKey (const DatePeriodKey &)
- ~DatePeriodKey ()
- const DatePeriod_T & getDatePeriod () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const

8.48.1 Detailed Description

Key of date-period.

Definition at line 14 of file DatePeriodKey.hpp.

8.48.2 Constructor & Destructor Documentation

8.48.2.1 stdair::DatePeriodKey::DatePeriodKey (const DatePeriod_T &)

Main Constructor.

Definition at line 22 of file DatePeriodKey.cpp.

8.48.2.2 stdair::DatePeriodKey::DatePeriodKey (const DatePeriodKey &)

Copy constructor.

Definition at line 27 of file DatePeriodKey.cpp.

8.48.2.3 stdair::DatePeriodKey::~DatePeriodKey ()

Destructor.

Definition at line 32 of file DatePeriodKey.cpp.

8.48.3 Member Function Documentation

8.48.3.1 const DatePeriod_T& stdair::DatePeriodKey::getDatePeriod () const [inline]

Get the date period.

Definition at line 32 of file DatePeriodKey.hpp.

Referenced by stdair::DatePeriod::getDatePeriod().

8.48.3.2 void stdair::DatePeriodKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 36 of file DatePeriodKey.cpp.

References toString().

8.48.3.3 void stdair::DatePeriodKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 41 of file DatePeriodKey.cpp.

8.48.3.4 const std::string stdair::DatePeriodKey::toString () const [virtual]

Get the serialised version of the Business Object Key. That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

Reimplemented from stdair::KeyAbstract.

Definition at line 45 of file DatePeriodKey.cpp.

Referenced by stdair::DatePeriod::describeKey(), and toStream().

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

- stdair/bom/DatePeriodKey.hpp
- stdair/bom/DatePeriodKey.cpp

8.49 stdair::DbaAbstract Class Reference

`#include <stdair/dbadaptor/DbaAbstract.hpp>`

Public Member Functions

- virtual ~DbaAbstract ()
- virtual void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

Protected Member Functions

- DbaAbstract ()

8.49.1 Detailed Description

Base class for the Database Adaptor (DBA) layer.

Definition at line 13 of file DbaAbstract.hpp.

8.49.2 Constructor & Destructor Documentation

8.49.2.1 virtual stdair::DbaAbstract::~DbaAbstract () [inline, virtual]

Destructor.

Definition at line 17 of file DbaAbstract.hpp.

8.49.2.2 stdair::DbaAbstract::DbaAbstract () [inline, protected]

Protected Default Constructor to ensure this class is abtract.

Definition at line 29 of file DbaAbstract.hpp.

8.49.3 Member Function Documentation

8.49.3.1 virtual void stdair::DbaAbstract::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Definition at line 21 of file DbaAbstract.hpp.

8.49.3.2 virtual void stdair::DbaAbstract::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Definition at line 25 of file DbaAbstract.hpp.

Referenced by operator>>().

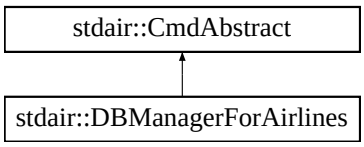
The documentation for this class was generated from the following file:

- stdair/dbadaptor/DbaAbstract.hpp

8.50 stdair::DBManagerForAirlines Class Reference

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

Inheritance diagram for stdair::DBManagerForAirlines::



Static Public Member Functions

- static void updateAirlineInDB (DBSession_T &, const AirlineStruct &)
- static bool retrieveAirline (DBSession_T &, const AirlineCode_T &, AirlineStruct &)
- static void prepareSelectStatement (DBSession_T &, DBRequestStatement_T &, AirlineStruct &)
- static bool iterateOnStatement (DBRequestStatement_T &, AirlineStruct &)

8.50.1 Detailed Description

Class building the Business Object Model (BOM) from data retrieved from the database.

Definition at line 18 of file DBManagerForAirlines.hpp.

8.50.2 Member Function Documentation

8.50.2.1 void stdair::DBManagerForAirlines::updateAirlineInDB (DBSession_T &, const AirlineStruct &) [static]

Update the fields of the database row corresponding to the given BOM object. DBSession_T& AirlineStruct& .

Definition at line 99 of file DBManagerForAirlines.cpp.

References stdair::AirlineStruct::getAirlineCode(), and stdair::AirlineStruct::getAirlineName().

8.50.2.2 bool stdair::DBManagerForAirlines::retrieveAirline (DBSession_T &, const AirlineCode_T &, AirlineStruct &) [static]

Retrieve, from the (MySQL) database, the row corresponding to the given BOM code, and fill the given BOM object with that retrieved data. DBSession_T& const AirlineCode_T& AirlineStruct& .

Definition at line 134 of file DBManagerForAirlines.cpp.

References iterateOnStatement().

8.50.2.3 void stdair::DBManagerForAirlines::prepareSelectStatement (DBSession_T &, DBRequestStatement_T &, AirlineStruct &) [static]

Prepare (parse and put in cache) the SQL statement. DBSession_T& DBRequestStatement_T& AirlineStruct& .

Definition at line 26 of file DBManagerForAirlines.cpp.

8.50.2.4 bool stdair::DBManagerForAirlines::iterateOnStatement (DBRequestStatement_T &, AirlineStruct &) [static]

Iterate on the SQL statement.

The SQL has to be already prepared. DBRequestStatement_T& AirlineStruct& .

Definition at line 82 of file DBManagerForAirlines.cpp.

Referenced by retrieveAirline().

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

- stdair/command/DBManagerForAirlines.hpp
- stdair/command/DBManagerForAirlines.cpp

8.51 stdair::DBSessionManager Class Reference

`#include <stdair/service/DBSessionManager.hpp>`

Public Member Functions

- DBSession_T & getDBSession () const

Static Public Member Functions

- static DBSessionManager & instance ()

Friends

- class FacSupervisor
- class STDAIR_Service

8.51.1 Detailed Description

Class holding the database session.

Note that the database access is handled by the SOCI library.

Definition at line 17 of file DBSessionManager.hpp.

8.51.2 Member Function Documentation

8.51.2.1 DBSessionManager & stdair::DBSessionManager::instance () [static]

Return the static DBSessionManager instance.

Definition at line 82 of file DBSessionManager.cpp.

8.51.2.2 DBSession_T & stdair::DBSessionManager::getDBSession () const

Retrieve the database session handler, held by the static instance of DBSessionManager.

Definition at line 92 of file DBSessionManager.cpp.

8.51.3 Friends And Related Function Documentation

8.51.3.1 friend class FacSupervisor [friend]

Definition at line 19 of file DBSessionManager.hpp.

8.51.3.2 friend class STDAIR_Service [friend]

Definition at line 20 of file DBSessionManager.hpp.

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

- stdair/service/DBSessionManager.hpp
- stdair/service/DBSessionManager.cpp

8.52 stdair::DefaultDCPList Struct Reference

`#include <stdair/basic/BasConst_Inventory.hpp>`

Static Public Member Functions

- static DCPList_T init ()

8.52.1 Detailed Description

Definition at line 126 of file BasConst_Inventory.hpp.

8.52.2 Member Function Documentation

8.52.2.1 DCPList_T stdair::DefaultDCPList::init () [static]

Definition at line 518 of file BasConst.cpp.

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

- stdair/basic/BasConst_Inventory.hpp
- stdair/basic/BasConst.cpp

8.53 stdair::DefaultDtdFratMap Struct Reference

`#include <stdair/basic/BasConst_Inventory.hpp>`

Static Public Member Functions

- static DTDFratMap_T init ()

8.53.1 Detailed Description

Definition at line 130 of file BasConst_Inventory.hpp.

8.53.2 Member Function Documentation

8.53.2.1 DTDFratMap_T stdair::DefaultDtdFratMap::init () [static]

Definition at line 697 of file BasConst.cpp.

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

- stdair/basic/BasConst_Inventory.hpp
- stdair/basic/BasConst.cpp

8.54 stdair::DefaultDtdProbMap Struct Reference

`#include <stdair/basic/BasConst_Inventory.hpp>`

Static Public Member Functions

- static DTDProbMap_T init ()

8.54.1 Detailed Description

Definition at line 134 of file BasConst_Inventory.hpp.

8.54.2 Member Function Documentation

8.54.2.1 DTDProbMap_T stdair::DefaultDtdProbMap::init () [static]

Definition at line 714 of file BasConst.cpp.

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

- stdair/basic/BasConst_Inventory.hpp
- stdair/basic/BasConst.cpp

8.55 stdair::DefaultMap Struct Reference

`#include <stdair/basic/BasConst_SellUpCurves.hpp>`

Static Public Member Functions

- static FRAT5Curve_T createFRAT5CurveA ()
- static FRAT5Curve_T createFRAT5CurveB ()
- static FRAT5Curve_T createFRAT5CurveC ()
- static FRAT5Curve_T createFRAT5CurveD ()
- static FFDisutilityCurve_T createFFDisutilityCurveA ()
- static FFDisutilityCurve_T createFFDisutilityCurveB ()
- static FFDisutilityCurve_T createFFDisutilityCurveC ()
- static FFDisutilityCurve_T createFFDisutilityCurveD ()
- static FFDisutilityCurve_T createFFDisutilityCurveE ()
- static FFDisutilityCurve_T createFFDisutilityCurveF ()

8.55.1 Detailed Description

FRAT5 curves.

Definition at line 27 of file BasConst_SellUpCurves.hpp.

8.55.2 Member Function Documentation

8.55.2.1 FRAT5Curve_T stdair::DefaultMap::createFRAT5CurveA () [static]

Definition at line 533 of file BasConst.cpp.

8.55.2.2 FRAT5Curve_T stdair::DefaultMap::createFRAT5CurveB () [static]

Definition at line 547 of file BasConst.cpp.

8.55.2.3 FRAT5Curve_T stdair::DefaultMap::createFRAT5CurveC () [static]

Definition at line 561 of file BasConst.cpp.

8.55.2.4 FRAT5Curve_T stdair::DefaultMap::createFRAT5CurveD () [static]

Definition at line 575 of file BasConst.cpp.

8.55.2.5 FFDisutilityCurve_T stdair::DefaultMap::createFFDisutilityCurveA () [static]

Definition at line 593 of file BasConst.cpp.

8.55.2.6 FFDisutilityCurve_T stdair::DefaultMap::createFFDisutilityCurveB () [static]

Definition at line 611 of file BasConst.cpp.

8.55.2.7 FFDisutilityCurve_T stdair::DefaultMap::createFFDisutilityCurveC () [static]

Definition at line 629 of file BasConst.cpp.

8.55.2.8 FFDisutilityCurve_T stdair::DefaultMap::createFFDisutilityCurveD () [static]

Definition at line 647 of file BasConst.cpp.

8.55.2.9 FFDisutilityCurve_T stdair::DefaultMap::createFFDisutilityCurveE () [static]

Definition at line 665 of file BasConst.cpp.

8.55.2.10 FFDisutilityCurve_T stdair::DefaultMap::createFFDisutilityCurveF () [static]

Definition at line 683 of file BasConst.cpp.

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

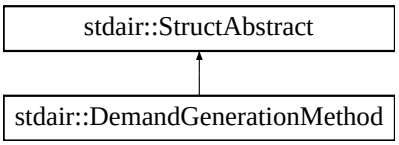
- stdair/basic/BasConst_SellUpCurves.hpp
- stdair/basic/BasConst.cpp

8.56 stdair::DemandGenerationMethod Struct Reference

Enumeration of demand (booking request) generation methods.

`#include <stdair/basic/DemandGenerationMethod.hpp>`

Inheritance diagram for stdair::DemandGenerationMethod::



Public Types

- POI_PRO = 0
- STA_ORD
- LAST_VALUE
- enum EN_DemandGenerationMethod { POI_PRO = 0, STA_ORD, LAST_VALUE }

Public Member Functions

- EN_DemandGenerationMethod getMethod () const
- char getMethodAsChar () const
- std::string getMethodAsString () const
- const std::string describe () const
- bool operator== (const EN_DemandGenerationMethod &) const
- DemandGenerationMethod (const EN_DemandGenerationMethod &)
- DemandGenerationMethod (const char iMethod)
- DemandGenerationMethod (const std::string &iMethod)
- DemandGenerationMethod (const DemandGenerationMethod &)
- void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

Static Public Member Functions

- static const std::string & getLabel (const EN_DemandGenerationMethod &)
- static EN_DemandGenerationMethod getMethod (const char)
- static char getMethodLabel (const EN_DemandGenerationMethod &)
- static std::string getMethodLabelAsString (const EN_DemandGenerationMethod &)
- static std::string describeLabels ()

8.56.1 Detailed Description

Enumeration of demand (booking request) generation methods.

Definition at line 17 of file DemandGenerationMethod.hpp.

8.56.2 Member Enumeration Documentation

8.56.2.1 enum stdair::DemandGenerationMethod::EN_DemandGenerationMethod

Enumerator:

POI_PRO

STA_ORD

LAST_VALUE

Definition at line 19 of file DemandGenerationMethod.hpp.

8.56.3 Constructor & Destructor Documentation

8.56.3.1 stdair::DemandGenerationMethod::DemandGenerationMethod (const EN_DemandGenerationMethod &)

Main constructor.

Definition at line 34 of file DemandGenerationMethod.cpp.

8.56.3.2 stdair::DemandGenerationMethod::DemandGenerationMethod (const char *iMethod*)

Alternative constructor.

Definition at line 62 of file DemandGenerationMethod.cpp.

8.56.3.3 stdair::DemandGenerationMethod::DemandGenerationMethod (const std::string & *iMethod*)

Alternative constructor.

Definition at line 68 of file DemandGenerationMethod.cpp.

References getMethod().

8.56.3.4 stdair::DemandGenerationMethod::DemandGenerationMethod (const DemandGenerationMethod &)

Default copy constructor.

Definition at line 28 of file DemandGenerationMethod.cpp.

8.56.4 Member Function Documentation

8.56.4.1 const std::string & stdair::DemandGenerationMethod::getLabel (const EN_DemandGenerationMethod &) [static]

Get the label as a string (e.g., "PoissonProcess" or "StatisticsOrder").

Definition at line 78 of file DemandGenerationMethod.cpp.

8.56.4.2 DemandGenerationMethod::EN_DemandGenerationMethod stdair::DemandGenerationMethod::getMethod (const *char*) [static]

Get the method value from parsing a single char (e.g., 'P' or 'S').

Definition at line 40 of file DemandGenerationMethod.cpp.

References describeLabels(), LAST_VALUE, POI_PRO, and STA_ORD.

8.56.4.3 char stdair::DemandGenerationMethod::getMethodLabel (const EN_DemandGenerationMethod &) [static]

Get the label as a single char (e.g., 'P' or 'S').

Definition at line 84 of file DemandGenerationMethod.cpp.

8.56.4.4 std::string stdair::DemandGenerationMethod::getMethodLabelAsString (const EN_DemandGenerationMethod &) [static]

Get the label as a string of a single char (e.g., "P" or "S").

Definition at line 90 of file DemandGenerationMethod.cpp.

8.56.4.5 std::string stdair::DemandGenerationMethod::describeLabels () [static]

List the labels.

Definition at line 97 of file DemandGenerationMethod.cpp.

References LAST_VALUE.

Referenced by getMethod().

8.56.4.6 DemandGenerationMethod::EN_DemandGenerationMethod stdair::DemandGenerationMethod::getMethod () const

Get the enumerated value.

Definition at line 110 of file DemandGenerationMethod.cpp.

Referenced by DemandGenerationMethod().

8.56.4.7 char stdair::DemandGenerationMethod::getMethodAsChar () const

Get the enumerated value as a short string (e.g., 'P' or 'S').

Definition at line 115 of file DemandGenerationMethod.cpp.

8.56.4.8 std::string stdair::DemandGenerationMethod::getMethodAsString () const

Get the enumerated value as a short string (e.g., "P" or "S").

Definition at line 121 of file DemandGenerationMethod.cpp.

8.56.4.9 const std::string stdair::DemandGenerationMethod::describe () const [virtual]

Give a description of the structure (e.g., "PoissonProcess" or "SatisticsOrder").

Implements stdair::StructAbstract.

Definition at line 128 of file DemandGenerationMethod.cpp.

8.56.4.10 bool stdair::DemandGenerationMethod::operator== (const EN_DemandGenerationMethod &) const

Comparison operator.

Definition at line 136 of file DemandGenerationMethod.cpp.

8.56.4.11 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const [inline, inherited]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References stdair::StructAbstract::describe().

8.56.4.12 virtual void stdair::StructAbstract::fromStream (std::istream & *ioIn*) [inline, virtual, inherited]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented in stdair::ProgressStatusSet, stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::EventStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 38 of file StructAbstract.hpp.

Referenced by operator>>().

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

- stdair/basic/DemandGenerationMethod.hpp
- stdair/basic/DemandGenerationMethod.cpp

8.57 stdair::DictionaryManager Class Reference

Class wrapper of dictionary business methods.

`#include <stdair/basic/DictionaryManager.hpp>`

Static Public Member Functions

- static const stdair::Probability_T keyToValue (const DictionaryKey_T)
- static const DictionaryKey_T valueToKey (const stdair::Probability_T)

8.57.1 Detailed Description

Class wrapper of dictionary business methods.

Definition at line 22 of file DictionaryManager.hpp.

8.57.2 Member Function Documentation

8.57.2.1 const stdair::Probability_T stdair::DictionaryManager::keyToValue (const *DictionaryKey_T*) [static]

Convert from key to value.

Definition at line 12 of file DictionaryManager.cpp.

References stdair::DEFAULT_NUMBER_OF_SUBDIVISIONS.

Referenced by stdair::ContinuousAttributeLite< T >::displayCumulativeDistribution(), stdair::ContinuousAttributeLite< T >::getDerivativeValue(), stdair::ContinuousAttributeLite< T >::getRemainingProportion(), and stdair::ContinuousAttributeLite< T >::getValue().

8.57.2.2 const DictionaryKey_T stdair::DictionaryManager::valueToKey (const stdair::Probability_T) [static]

Convert from value to key.

Definition at line 21 of file DictionaryManager.cpp.

References stdair::DEFAULT_NUMBER_OF_SUBDIVISIONS.

Referenced by stdair::ContinuousAttributeLite< T >::getValue().

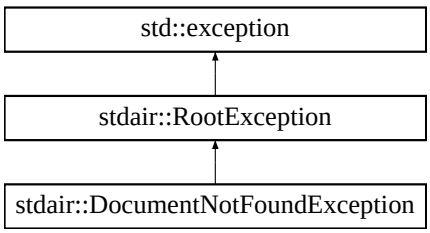
The documentation for this class was generated from the following files:

- stdair/basic/DictionaryManager.hpp
- stdair/basic/DictionaryManager.cpp

8.58 stdair::DocumentNotFoundException Class Reference

`#include <stdair/stdair_exceptions.hpp>`

Inheritance diagram for stdair::DocumentNotFoundException::



Public Member Functions

- DocumentNotFoundException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

8.58.1 Detailed Description

Document not found.

Definition at line 104 of file stdair_exceptions.hpp.

8.58.2 Constructor & Destructor Documentation

8.58.2.1 stdair::DocumentNotFoundException::DocumentNotFoundException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 107 of file stdair_exceptions.hpp.

8.58.3 Member Function Documentation

8.58.3.1 const char* stdair::RootException::what () const throw () [inline, inherited]

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

8.58.4 Member Data Documentation

8.58.4.1 std::string stdair::RootException::_what [protected, inherited]

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

Referenced by stdair::RootException::what().

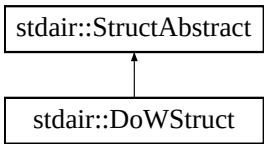
The documentation for this class was generated from the following file:

- stdair/stdair_exceptions.hpp

8.59 stdair::DoWStruct Struct Reference

`#include <stdair/bom/DoWStruct.hpp>`

Inheritance diagram for stdair::DoWStruct::



Public Types

- typedef std::vector< bool > BooleanList_T

Public Member Functions

- bool getDayOfWeek (const unsigned short i) const
- bool getStandardDayOfWeek (const unsigned short i) const
- void setDayOfWeek (const unsigned short, const bool)
- const std::string describe () const
- const std::string describeShort () const
- DoWStruct shift (const long &) const
- DoWStruct intersection (const DoWStruct &) const
- const bool isValid () const
- DoWStruct (const std::string &iDowString)
- DoWStruct ()
- DoWStruct (const DoWStruct &)
- ~DoWStruct ()
- void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

8.59.1 Detailed Description

Define a Day Of the Week (DoW) sequence.

For instance, 1..11.1 means that the period is active on Mon., Thu., Fri. and Sun.

Definition at line 18 of file DoWStruct.hpp.

8.59.2 Member Typedef Documentation

8.59.2.1 typedef std::vector<bool> stdair::DoWStruct::BooleanList_T

Define the bit set representing the DoW.

Definition at line 21 of file DoWStruct.hpp.

8.59.3 Constructor & Destructor Documentation

8.59.3.1 stdair::DoWStruct::DoWStruct (const std::string & *iDowString*)

Constructor from a given bit set (e.g., "0000011" for the week-ends).

Definition at line 21 of file DoWStruct.cpp.

8.59.3.2 stdair::DoWStruct::DoWStruct ()

Default constructors.

Definition at line 14 of file DoWStruct.cpp.

8.59.3.3 stdair::DoWStruct::DoWStruct (const DoWStruct &)

Definition at line 34 of file DoWStruct.cpp.

8.59.3.4 stdair::DoWStruct::~DoWStruct () [inline]

Default destructor.

Definition at line 63 of file DoWStruct.hpp.

8.59.4 Member Function Documentation

8.59.4.1 bool stdair::DoWStruct::getDayOfWeek (const unsigned short *i*) const

Get the i-th day of the week (Monday being the first one).

Definition at line 66 of file DoWStruct.cpp.

Referenced by intersection(), and isValid().

8.59.4.2 bool stdair::DoWStruct::getStandardDayOfWeek (const unsigned short *i*) const

Get the i-th day of the week (Sunday being the first one).

Definition at line 71 of file DoWStruct.cpp.

8.59.4.3 void stdair::DoWStruct::setDayOfWeek (const unsigned *short*, const *bool*)

Set the new value for the i-th day-of-week.

Definition at line 82 of file DoWStruct.cpp.

Referenced by intersection(), and shift().

8.59.4.4 const std::string stdair::DoWStruct::describe () const [virtual]

Display explicitly (e.g., "Mon.Tue.Wed.Thu.Fri.").

Implements stdair::StructAbstract.

Definition at line 52 of file DoWStruct.cpp.

References stdair::DOW_STR.

Referenced by stdair::PeriodStruct::describe().

8.59.4.5 const std::string stdair::DoWStruct::describeShort () const

Display as a bit set (e.g., "1111100").

Definition at line 40 of file DoWStruct.cpp.

Referenced by stdair::PeriodStruct::describeShort().

8.59.4.6 DoWStruct stdair::DoWStruct::shift (const long &) const

Build a new DoW struct by shifting the current DoW by a given number.

Definition at line 88 of file DoWStruct.cpp.

References stdair::DEFAULT_DOW_STRING, and setDayOfWeek().

Referenced by stdair::PeriodStruct::addDateOffset().

8.59.4.7 DoWStruct stdair::DoWStruct::intersection (const DoWStruct &) const

Build a new DoW struct by intersecting two DoW structs.

Definition at line 104 of file DoWStruct.cpp.

References stdair::DEFAULT_DOW_STRING, getDayOfWeek(), and setDayOfWeek().

Referenced by stdair::PeriodStruct::intersection().

8.59.4.8 const bool stdair::DoWStruct::isValid () const

Return if the DoW struct is valid (i.e., has at least one "true").

Definition at line 117 of file DoWStruct.cpp.

References getDayOfWeek().

Referenced by stdair::PeriodStruct::isValid().

8.59.4.9 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const [inline, inherited]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References stdair::StructAbstract::describe().

8.59.4.10 virtual void stdair::StructAbstract::fromStream (std::istream & *ioIn*) [inline, virtual, inherited]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented in stdair::ProgressStatusSet, stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct,

stdair::EventStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 38 of file StructAbstract.hpp.

Referenced by operator>>().

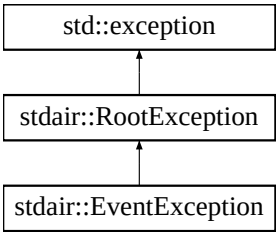
The documentation for this struct was generated from the following files:

- stdair/bom/DoWStruct.hpp
- stdair/bom/DoWStruct.cpp

8.60 stdair::EventException Class Reference

`#include <stdair/stdair_exceptions.hpp>`

Inheritance diagram for stdair::EventException::



Public Member Functions

- EventException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

8.60.1 Detailed Description

Event.

Definition at line 204 of file stdair_exceptions.hpp.

8.60.2 Constructor & Destructor Documentation

8.60.2.1 stdair::EventException::EventException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 207 of file stdair_exceptions.hpp.

8.60.3 Member Function Documentation

8.60.3.1 const char* stdair::RootException::what () const throw () [inline, inherited]

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

8.60.4 Member Data Documentation

8.60.4.1 std::string stdair::RootException::_what [protected, inherited]

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

Referenced by stdair::RootException::what().

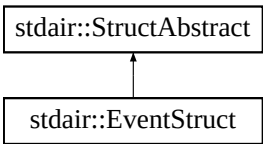
The documentation for this class was generated from the following file:

- stdair/stdair_exceptions.hpp

8.61 stdair::EventStruct Struct Reference

`#include <stdair/bom/EventStruct.hpp>`

Inheritance diagram for stdair::EventStruct::



Public Member Functions

- const EventType::EN_EventType & getEventType () const
- const LongDuration_T & getEventTimeStamp () const
- const DateTime_T & getEventTime () const
- const BookingRequestStruct & getBookingRequest () const
- const CancellationStruct & getCancellation () const
- const OptimisationNotificationStruct & getOptimisationNotificationStruct () const
- const SnapshotStruct & getSnapshotStruct () const
- const RMEventStruct & getRMEvent () const
- const BreakPointStruct & getBreakPoint () const
- void fromStream (std::istream &ioIn)
- const std::string describe () const
- EventStruct ()
- EventStruct (const EventType::EN_EventType &, BookingRequestPtr_T)
- EventStruct (const EventType::EN_EventType &, CancellationPtr_T)

- EventStruct (const EventType::EN_EventType &, const DateTime_T &iDCPDate, OptimisationNotificationPtr_T)
- EventStruct (const EventType::EN_EventType &, SnapshotPtr_T)
- EventStruct (const EventType::EN_EventType &, RMEventPtr_T)
- EventStruct (const EventType::EN_EventType &, BreakPointPtr_T)
- EventStruct (const EventStruct &)
- ~EventStruct ()
- void incrementEventTimeStamp ()
- void toStream (std::ostream &ioOut) const

8.61.1 Detailed Description

Structure holding the details of an event.

Note:

No event should be scheduled before the date-time corresponding to the DEFAULT_EVENT_OLDEST_DATETIME constant (as of Feb. 2011, that date is set to Jan. 1, 2010). That constant is specified in the stdair/basic/BasConst.cpp file. In other words, the simulation should not specified to start before that date-time.

Definition at line 36 of file EventStruct.hpp.

8.61.2 Constructor & Destructor Documentation

8.61.2.1 stdair::EventStruct::EventStruct ()

Default constructor.

Definition at line 26 of file EventStruct.cpp.

8.61.2.2 stdair::EventStruct::EventStruct (const EventType::EN_EventType &, BookingRequestPtr_T)

Constructor for events corresponding to booking requests.

Definition at line 31 of file EventStruct.cpp.

References stdair::DEFAULT_EVENT_OLDEST_DATETIME.

8.61.2.3 stdair::EventStruct::EventStruct (const EventType::EN_EventType &, CancellationPtr_T)

Constructor for events corresponding to cancellations.

Definition at line 55 of file EventStruct.cpp.

References stdair::DEFAULT_EVENT_OLDEST_DATETIME.

8.61.2.4 stdair::EventStruct::EventStruct (const EventType::EN_EventType &, const DateTime_T & *iDCPDate*, OptimisationNotificationPtr_T)

Constructor for events corresponding to optimisation requests.

Definition at line 80 of file EventStruct.cpp.

References stdair::DEFAULT_EVENT_OLDEST_DATETIME.

8.61.2.5 stdair::EventStruct::EventStruct (const EventType::EN_EventType &, SnapshotPtr_T)

Constructor for events corresponding to snapshot requests.

Definition at line 105 of file EventStruct.cpp.

References stdair::DEFAULT_EVENT_OLDEST_DATETIME.

8.61.2.6 stdair::EventStruct::EventStruct (const EventType::EN_EventType &, RMEventPtr_T)

Constructor for events corresponding to RM events.

Definition at line 130 of file EventStruct.cpp.

References stdair::DEFAULT_EVENT_OLDEST_DATETIME.

8.61.2.7 stdair::EventStruct::EventStruct (const EventType::EN_EventType &, BreakPointPtr_T)

Constructor for events corresponding to Break Point events.

Definition at line 155 of file EventStruct.cpp.

References stdair::DEFAULT_EVENT_OLDEST_DATETIME.

8.61.2.8 stdair::EventStruct::EventStruct (const EventStruct &)

Copy constructor.

Definition at line 180 of file EventStruct.cpp.

References _bookingRequest, _breakPoint, _cancellation, _optimisationNotification, _rmEvent, and _snapshot.

8.61.2.9 stdair::EventStruct::~EventStruct ()

Destructor.

Definition at line 243 of file EventStruct.cpp.

8.61.3 Member Function Documentation

8.61.3.1 const EventType::EN_EventType& stdair::EventStruct::getEventType () const [inline]

Get the event type

Definition at line 41 of file EventStruct.hpp.

Referenced by stdair::BomJSONExport::jsonExportBookingRequestObject(), stdair::BomJSONExport::jsonExportBreakPointObject(), and stdair::STDAIR_Service::jsonExportEventObject().

8.61.3.2 const LongDuration_T& stdair::EventStruct::getEventTimeStamp () const [inline]

Get the event time stamp

Definition at line 46 of file EventStruct.hpp.

8.61.3.3 const DateTime_T & stdair::EventStruct::getEventTime () const

Get the event date-time i.e the stamp converted to a date-time format.

Definition at line 311 of file EventStruct.cpp.

References stdair::EventType::BKG_REQ, stdair::EventType::BRK_PT, stdair::EventType::CX, stdair::DEFAULT_EVENT_OLDEST_DATETIME, stdair::EventType::OPT_NOT_4_FD, stdair::EventType::RM, and stdair::EventType::SNAPSHOT.

8.61.3.4 const BookingRequestStruct& stdair::EventStruct::getBookingRequest () const [inline]

Get a reference on the booking request referred to by event.

Note:

When that event is not of type booking request (EventType::BKG_REQ), an assertion fails.

Definition at line 59 of file EventStruct.hpp.

Referenced by stdair::BomJSONExport::jsonExportBookingRequestObject().

8.61.3.5 const CancellationStruct& stdair::EventStruct::getCancellation () const [inline]

Get a reference on the cancellation referred to by event.

Note:

When that event is not of type cancellation (EventType::CX), an assertion fails.

Definition at line 70 of file EventStruct.hpp.

8.61.3.6 const OptimisationNotificationStruct& stdair::EventStruct::getOptimisationNotificationStruct () const [inline]

Get a reference on the optimisation notification referred to by event.

Note:

When that event is not of type optimisation notification for optimisation notification (EventType::OPT_NOT_4_FD), an assertion fails.

Definition at line 83 of file EventStruct.hpp.

8.61.3.7 const SnapshotStruct& stdair::EventStruct::getSnapshotStruct () const [inline]

Get a reference on the snapshot referred to by event.

Note:

When that event is not of type snapshot for snapshot (EventType::OPT_NOT_4_FD), an assertion fails.

Definition at line 95 of file EventStruct.hpp.

8.61.3.8 const RMEventStruct& stdair::EventStruct::getRMEvent () const [inline]

Get a reference on the RM event referred to by the generic event.

Note:

When that event is not of type RM event for snapshot (EventType::OPT_NOT_4_FD), an assertion fails.

Definition at line 107 of file EventStruct.hpp.

8.61.3.9 const BreakPointStruct& stdair::EventStruct::getBreakPoint () const [inline]

Get a reference on the break point referred to by event.

Note:

When that event is not of type booking break point (EventType::BRK_PT), an assertion fails.

Definition at line 118 of file EventStruct.hpp.

Referenced by stdair::BomJSONExport::jsonExportBreakPointObject().

8.61.3.10 void stdair::EventStruct::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::StructAbstract.

Definition at line 247 of file EventStruct.cpp.

8.61.3.11 const std::string stdair::EventStruct::describe () const [virtual]

Display of the structure.

Implements stdair::StructAbstract.

Definition at line 251 of file EventStruct.cpp.

References stdair::EventType::BKG_REQ, stdair::EventType::BRK_PT, stdair::EventType::CX, stdair::DEFAULT_EVENT_OLDEST_DATETIME, stdair::EventType::getLabel(), stdair::EventType::OPT_NOT_4_FD, stdair::EventType::RM, and stdair::EventType::SNAPSHOT.

8.61.3.12 void stdair::EventStruct::incrementEventTimeStamp ()

Increment the date-time stamp which is counted in milliseconds.

This incrementation of one millisecond is needed when the insertion in the event queue failed, that is to say when an event with the exact same time stamp has already been inserted in the queue.

Definition at line 357 of file EventStruct.cpp.

8.61.3.13 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const [inline, inherited]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References stdair::StructAbstract::describe().

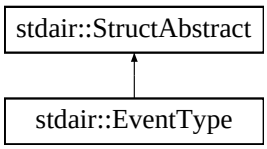
The documentation for this struct was generated from the following files:

- stdair/bom/EventStruct.hpp
- stdair/bom/EventStruct.cpp

8.62 stdair::EventType Struct Reference

`#include <stdair/basic/EventType.hpp>`

Inheritance diagram for stdair::EventType::



Public Types

- BKG_REQ = 0
- CX
- OPT_NOT_4_FD
- OPT_NOT_4_NET
- SKD_CHG
- SNAPSHOT
- RM
- BRK_PT
- LAST_VALUE
- enum EN_EventType {
  BKG_REQ = 0, CX, OPT_NOT_4_FD, OPT_NOT_4_NET,
  SKD_CHG, SNAPSHOT, RM, BRK_PT,
  LAST_VALUE }

Public Member Functions

- EN_EventType getType () const
- std::string getTypeAsString () const
- const std::string describe () const
- bool operator== (const EN_EventType &) const
- EventType (const EN_EventType &)
- EventType (const char iType)
- EventType (const std::string &iTypeStr)
- EventType (const EventType &)
- void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

Static Public Member Functions

- static const std::string & getLabel (const EN_EventType &)
- static char getTypeLabel (const EN_EventType &)
- static std::string getTypeLabelAsString (const EN_EventType &)
- static std::string describeLabels ()

8.62.1 Detailed Description

Enumeration of event types.

Definition at line 15 of file EventType.hpp.

8.62.2 Member Enumeration Documentation

8.62.2.1 enum stdair::EventType::EN_EventType

Enumerator:

BKG_REQ
CX
OPT_NOT_4_FD
OPT_NOT_4_NET
SKD_CHG
SNAPSHOT
RM
BRK_PT
LAST_VALUE

Definition at line 17 of file EventType.hpp.

8.62.3 Constructor & Destructor Documentation

8.62.3.1 stdair::EventType::EventType (const EN_EventType &)

Constructor.

Definition at line 36 of file EventType.cpp.

8.62.3.2 stdair::EventType::EventType (const char *iType*)

Constructor using a char.

Definition at line 41 of file EventType.cpp.

References BKG_REQ, BRK_PT, CX, describeLabels(), LAST_VALUE, OPT_NOT_4_FD, OPT_NOT_-4_NET, RM, SKD_CHG, and SNAPSHOT.

8.62.3.3 stdair::EventType::EventType (const std::string & *iTypeStr*)

Constructor using a string.

Definition at line 64 of file EventType.cpp.

References describeLabels(), and LAST_VALUE.

8.62.3.4 stdair::EventType::EventType (const EventType &)

Default copy constructor.

Definition at line 31 of file EventType.cpp.

8.62.4 Member Function Documentation

8.62.4.1 const std::string & stdair::EventType::getLabel (const EN_EventType &) [static]

Get the label as a string (e.g., "BookingRequest", "Cancellation", "OptimisationNotificationForFlight-Date", "OptimisationNotificationForNetwork", "ScheduleChange", "Snapshot", "RevenueManagement", "BreakPoint" or "BookingRequest").

Definition at line 83 of file EventType.cpp.

Referenced by stdair::EventStruct::describe(), stdair::BomJSONExport::jsonExportBookingRequest-Object(), and stdair::BomJSONExport::jsonExportBreakPointObject().

8.62.4.2 char stdair::EventType::getTypeLabel (const EN_EventType &) [static]

Get the label as a single char (e.g., 'B', 'X', 'F', 'N', 'C', 'S', 'R' or 'P').

Definition at line 88 of file EventType.cpp.

8.62.4.3 std::string stdair::EventType::getTypeLabelAsString (const EN_EventType &) [static]

Get the label as a string of a single char (e.g., "B", "X", "F", "N", "C", "S", "R" or "P").

Definition at line 93 of file EventType.cpp.

8.62.4.4 std::string stdair::EventType::describeLabels () [static]

List the labels.

Definition at line 100 of file EventType.cpp.

References LAST_VALUE.

Referenced by EventType().

8.62.4.5 EventType::EN_EventType stdair::EventType::getType () const

Get the enumerated value.

Definition at line 112 of file EventType.cpp.

8.62.4.6 std::string stdair::EventType::getTypeAsString () const

Get the enumerated value as a short string (e.g., "B", "X", "F", "N", "C", "S", "R" or "P").

Definition at line 117 of file EventType.cpp.

8.62.4.7 const std::string stdair::EventType::describe () const [virtual]

Give a description of the structure (e.g., "BookingRequest", "Cancellation", "OptimisationNotificationForFlightDate", "OptimisationNotificationForNetwork", "ScheduleChange", "Snapshot", "RevenueManagement", "BreakPoint" or "BookingRequest").

Implements stdair::StructAbstract.

Definition at line 124 of file EventType.cpp.

8.62.4.8 bool stdair::EventType::operator== (const EN_EventType &) const

Comparison operator.

Definition at line 131 of file EventType.cpp.

8.62.4.9 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const [inline, inherited]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References stdair::StructAbstract::describe().

8.62.4.10 virtual void stdair::StructAbstract::fromStream (std::istream & *ioIn*) [inline, virtual, inherited]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented in stdair::ProgressStatusSet, stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct,

stdair::EventStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 38 of file StructAbstract.hpp.

Referenced by operator>>().

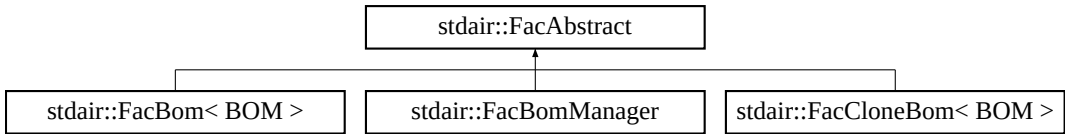
The documentation for this struct was generated from the following files:

- stdair/basic/EventType.hpp
- stdair/basic/EventType.cpp

8.63 stdair::FacAbstract Class Reference

`#include <stdair/factory/FacAbstract.hpp>`

Inheritance diagram for stdair::FacAbstract::



Public Member Functions

- virtual ~FacAbstract ()

Protected Member Functions

- FacAbstract ()

8.63.1 Detailed Description

Base class for Factory layer.

Definition at line 10 of file FacAbstract.hpp.

8.63.2 Constructor & Destructor Documentation

8.63.2.1 stdair::FacAbstract::~FacAbstract () [virtual]

Destructor.

Definition at line 13 of file FacAbstract.cpp.

8.63.2.2 stdair::FacAbstract::FacAbstract () [inline, protected]

Default Constructor.

This constructor is protected to ensure the class is abstract.

Definition at line 18 of file FacAbstract.hpp.

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

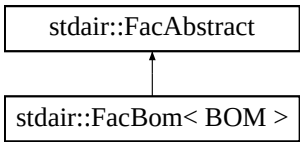
- stdair/factory/FacAbstract.hpp
- stdair/factory/FacAbstract.cpp

8.64 stdair::FacBom< BOM > Class Template Reference

Base class for Factory layer.

`#include <stdair/factory/FacBom.hpp>`

Inheritance diagram for stdair::FacBom< BOM >::



Public Member Functions

- BOM & create ()
- BOM & create (const Key_T &)
- BOM & create (const BOM &)
- ~FacBom ()
- void clean ()

Static Public Member Functions

- static FacBom & instance ()

Protected Member Functions

- FacBom ()

8.64.1 Detailed Description

template<typename BOM> class stdair::FacBom< BOM >

Base class for Factory layer.

Definition at line 22 of file FacBom.hpp.

8.64.2 Constructor & Destructor Documentation

8.64.2.1 template<typename BOM> stdair::FacBom< BOM >::FacBom () [inline, protected]

Default Constructor.

Definition at line 50 of file FacBom.hpp.

Referenced by stdair::FacBom< BOM >::instance().

8.64.2.2 template<typename BOM> stdair::FacBom< BOM >::~FacBom () [inline]

Destructor.

Definition at line 56 of file FacBom.hpp.

References stdair::FacBom< BOM >::clean().

8.64.3 Member Function Documentation

8.64.3.1 template<typename BOM> FacBom< BOM > & stdair::FacBom< BOM >::instance () [static]

Provide the unique instance.

The singleton is instantiated when first used.

Returns:

FacBom&

Definition at line 84 of file FacBom.hpp.

References stdair::FacBom< BOM >::FacBom(), stdair::FacSupervisor::instance(), and stdair::FacSupervisor::registerPersistentBomFactory().

Referenced by stdair::FacBom< BOM >::create().

8.64.3.2 template<typename BOM> BOM & stdair::FacBom< BOM >::create ()

Create a BOM object, given a key or not.

Definition at line 112 of file FacBom.hpp.

References stdair::FacBom< BOM >::create(), and stdair::FacBom< BOM >::instance().

Referenced by stdair::FacBom< BOM >::create().

8.64.3.3 template<typename BOM> BOM & stdair::FacBom< BOM >::create (const Key_T &)

Definition at line 118 of file FacBom.hpp.

8.64.3.4 template<typename BOM> BOM & stdair::FacBom< BOM >::create (const BOM &)

Definition at line 126 of file FacBom.hpp.

8.64.3.5 template<typename BOM> void stdair::FacBom< BOM >::clean ()

Destroyed all the object instantiated by this factory.

Definition at line 95 of file FacBom.hpp.

Referenced by stdair::FacBom< BOM >::~FacBom().

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

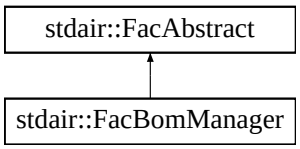
- stdair/factory/FacBom.hpp

8.65 stdair::FacBomManager Class Reference

Utility class for linking StdAir-based objects.

`#include <stdair/factory/FacBomManager.hpp>`

Inheritance diagram for stdair::FacBomManager::



Public Member Functions

- ~FacBomManager ()
- template<> void addToList (SegmentDate &ioSegmentDate, SegmentDate &ioMarketingSegmentDate)

Static Public Member Functions

- template<typename OBJECT2, typename OBJECT1> static BomHolder< OBJECT2 > * getBomHolderPtr (OBJECT1 &)
- template<typename OBJECT2, typename OBJECT1> static BomHolder< OBJECT2 > & addBomHolder (OBJECT1 &)
- template<typename OBJECT1, typename OBJECT2> static void addToList (OBJECT1 &, OBJECT2 &)
- template<typename OBJECT1, typename OBJECT2> static void addToMap (OBJECT1 &, OBJECT2 &, const MapKey_T &)
- template<typename OBJECT1, typename OBJECT2> static void addToMap (OBJECT1 &, OBJECT2 &)
- template<typename OBJECT1, typename OBJECT2> static void addToListAndMap (OBJECT1 &, OBJECT2 &)
- template<typename OBJECT1, typename OBJECT2> static void addToListAndMap (OBJECT1 &, OBJECT2 &, const MapKey_T &)
- template<typename PARENT, typename CHILD> static void linkWithParent (PARENT &, CHILD &)
- template<typename OBJECT2, typename OBJECT1> static void cloneHolder (OBJECT1 &, const OBJECT1 &)
- static void resetYieldBasedNestingStructure (const SegmentCabin &)
- static void setAirlineFeature (Inventory &iInventory, AirlineFeature &iAirlineFeature)
- static void linkWithOperating (SegmentDate &iSegmentDate, SegmentDate &iOperatingSegmentDate)

Protected Member Functions

- FacBomManager ()

8.65.1 Detailed Description

Utility class for linking StdAir-based objects.

Definition at line 30 of file FacBomManager.hpp.

8.65.2 Constructor & Destructor Documentation

8.65.2.1 stdair::FacBomManager::FacBomManager () [inline, protected]

Default Constructor.

This constructor is protected to comply with the singleton pattern.

Definition at line 225 of file FacBomManager.hpp.

8.65.2.2 stdair::FacBomManager::~FacBomManager () [inline]

Destructor.

Definition at line 231 of file FacBomManager.hpp.

8.65.3 Member Function Documentation

8.65.3.1 template<typename OBJECT2, typename OBJECT1> BomHolder< OBJECT2 > * stdair::FacBomManager::getBomHolderPtr (OBJECT1 &) [static]

Retrieve a pointer on the holder of children (OBJECT2 type) for the parent (OBJECT1 type). If the holder does not exist, return NULL.

Parameters:

typename OBJECT1& Parent object.

Returns:

typename BomHolder<OBJECT2>* BomHolder for the children objects.

Definition at line 268 of file FacBomManager.hpp.

8.65.3.2 template<typename OBJECT2, typename OBJECT1> BomHolder< OBJECT2 > & stdair::FacBomManager::addBomHolder (OBJECT1 &) [static]

Instantiate a BomHolder<OBJECT2> object, add it to the OBJECT1-typed object, given as parameter, and return a reference on that newly created BomHolder.

Parameters:

typename OBJECT1& Parent object.

Returns:

typename BomHolder<OBJECT2>& Just created BomHolder (e.g., for the children objects).

Definition at line 238 of file FacBomManager.hpp.

8.65.3.3 template<typename OBJECT1, typename OBJECT2> void stdair::FacBomManager::addToList (OBJECT1 &, OBJECT2 &) [static]

Add an OBJECT2-typed object (typically, a child) to the dedicated list held by the OBJECT1-typed object (typically, a parent).

Note:

The underlying list is actually stored within an object of type BomHolder<OBJECT2>.

Parameters:

typename OBJECT1& Parent object.

typename OBJECT2& Child object.

Definition at line 354 of file FacBomManager.hpp.

8.65.3.4 template<typename OBJECT1, typename OBJECT2> void stdair::FacBomManager::addToMap (OBJECT1 &, OBJECT2 &, const MapKey_T &) [static]

Add an OBJECT2-typed object (typically, a child) to the dedicated map held by the OBJECT1-typed object (typically, a parent).

Note:

The underlying map is actually stored within an object of type BomHolder<OBJECT2>.

Parameters:

typename OBJECT1& Parent object.

typename OBJECT2& Child object.

const MapKey_T&

Definition at line 424 of file FacBomManager.hpp.

Referenced by addToMap().

8.65.3.5 template<typename OBJECT1, typename OBJECT2> void stdair::FacBomManager::addToMap (OBJECT1 &, OBJECT2 &) [static]

Add an OBJECT2-typed object (typically, a child) to the dedicated map held by the OBJECT1-typed object (typically, a parent).

Note:

The underlying map is actually stored within an object of type BomHolder<OBJECT2>.

Parameters:

typename OBJECT1& Parent object.

typename OBJECT2& Child object.

Definition at line 446 of file FacBomManager.hpp.

References addToMap().

8.65.3.6 template<typename OBJECT1, typename OBJECT2> void stdair::FacBomManager::addToListAndMap (OBJECT1 &, OBJECT2 &) [static]

Add an OBJECT2-typed object (typically, a child) to the dedicated containers (list and map) held by the OBJECT1-typed object (typically, a parent).

Note:

The underlying containers are actually stored within an object of type BomHolder<OBJECT2>.

Parameters:

typename OBJECT1& Parent object.

typename OBJECT2& Child object.

Definition at line 490 of file FacBomManager.hpp.

8.65.3.7 template<typename OBJECT1, typename OBJECT2> void stdair::FacBomManager::addToListAndMap (OBJECT1 &, OBJECT2 &, const MapKey_T &) [static]

Add an OBJECT2-typed object (typically, a child) to the dedicated containers (list and map) held by the OBJECT1-typed object (typically, a parent).

Note:

The underlying containers are actually stored within an object of type BomHolder<OBJECT2>.

Parameters:

typename OBJECT1& Parent object.

typename OBJECT2& Child object.

const MapKey_T&

Definition at line 467 of file FacBomManager.hpp.

8.65.3.8 template<typename PARENT, typename CHILD> void stdair::FacBomManager::linkWithParent (PARENT &, CHILD &) [static]

Allow the CHILD object to store a pointer on its PARENT object.

Parameters:

typename PARENT& Parent object.

typename CHILD& Child object.

Definition at line 511 of file FacBomManager.hpp.

Referenced by stdair::serialiseHelper().

8.65.3.9 template<typename OBJECT2, typename OBJECT1> void stdair::FacBomManager::cloneHolder (OBJECT1 &, const OBJECT1 &) [static]

Clone the underlying containers (held by the BomHolder<OBJECT2>-typed holder) of the OBJECT1-typed object.

Note:

The underlying containers are actually stored within an object of type BomHolder<OBJECT2>.

Parameters:

typename OBJECT1& Parent object.

typename OBJECT2& Child object.

Definition at line 519 of file FacBomManager.hpp.

References stdair::BomHolder< BOM >::_bomList, and stdair::BomHolder< BOM >::_bomMap.

8.65.3.10 void stdair::FacBomManager::resetYieldBasedNestingStructure (const SegmentCabin &) [static]

Reset the yield-based nesting structure of a segment-cabin. This method is used with FA or MRT.

Definition at line 20 of file FacBomManager.cpp.

References stdair::YIELD_BASED_NESTING_STRUCTURE_CODE.

8.65.3.11 static void stdair::FacBomManager::setAirlineFeature (Inventory & *iInventory*, AirlineFeature & *iAirlineFeature*) [inline, static]

Set the airline feature object of an inventory.

Definition at line 205 of file FacBomManager.hpp.

References stdair::Inventory::setAirlineFeature().

8.65.3.12 static void stdair::FacBomManager::linkWithOperating (SegmentDate & *iSegmentDate*, SegmentDate & *ioOperatingSegmentDate*) [inline, static]

Link the segment date with its operating segment date.

Definition at line 213 of file FacBomManager.hpp.

References stdair::SegmentDate::linkWithOperating().

8.65.3.13 template<> void stdair::FacBomManager::addToList (SegmentDate & *ioSegmentDate*, SegmentDate & *ioMarketingSegmentDate*) [inline]

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

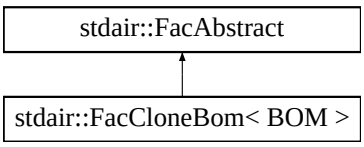
- stdair/factory/FacBomManager.hpp
- stdair/factory/FacBomManager.cpp

8.66 stdair::FacCloneBom< BOM > Class Template Reference

Base class for Factory layer.

`#include <stdair/factory/FacCloneBom.hpp>`

Inheritance diagram for stdair::FacCloneBom< BOM >::



Public Member Functions

- BOM & clone (const BOM &)
- ~FacCloneBom ()
- void clean ()

Static Public Member Functions

- static FacCloneBom & instance ()

Protected Member Functions

- FacCloneBom ()

8.66.1 Detailed Description

template<typename BOM> class stdair::FacCloneBom< BOM >

Base class for Factory layer.

Definition at line 22 of file FacCloneBom.hpp.

8.66.2 Constructor & Destructor Documentation

8.66.2.1 template<typename BOM> stdair::FacCloneBom< BOM >::FacCloneBom () [inline, protected]

Default Constructor.

Definition at line 48 of file FacCloneBom.hpp.

Referenced by stdair::FacCloneBom< BOM >::instance().

8.66.2.2 template<typename BOM> stdair::FacCloneBom< BOM >::~FacCloneBom () [inline]

Destructor.

Definition at line 54 of file FacCloneBom.hpp.

References stdair::FacCloneBom< BOM >::clean().

8.66.3 Member Function Documentation

8.66.3.1 template<typename BOM> FacCloneBom< BOM > & stdair::FacCloneBom< BOM >::instance () [static]

Provide the unique instance.

The singleton is instantiated when first used.

Returns:

FacCloneBom&

Definition at line 82 of file FacCloneBom.hpp.

References stdair::FacCloneBom< BOM >::FacCloneBom(), stdair::FacSupervisor::instance(), and stdair::FacSupervisor::registerCloneBomFactory().

8.66.3.2 template<typename BOM> BOM & stdair::FacCloneBom< BOM >::clone (const BOM &)

Clone a BOM object.

Definition at line 110 of file FacCloneBom.hpp.

8.66.3.3 template<typename BOM> void stdair::FacCloneBom< BOM >::clean ()

Destroyed all the object instantiated by this factory.

Definition at line 93 of file FacCloneBom.hpp.

Referenced by stdair::FacCloneBom< BOM >::~FacCloneBom().

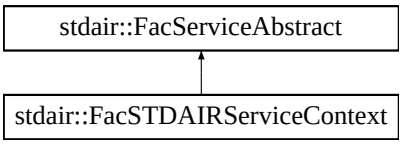
The documentation for this class was generated from the following file:

- stdair/factory/FacCloneBom.hpp

8.67 stdair::FacServiceAbstract Class Reference

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

Inheritance diagram for stdair::FacServiceAbstract::



Public Types

- typedef std::vector< ServiceAbstract * > ServicePool_T

Public Member Functions

- virtual ~FacServiceAbstract ()
- void clean ()

Protected Member Functions

- FacServiceAbstract ()

Protected Attributes

- ServicePool_T _pool

8.67.1 Detailed Description

Base class for the (Service) Factory layer.

Definition at line 16 of file FacServiceAbstract.hpp.

8.67.2 Member Typedef Documentation

8.67.2.1 typedef std::vector<ServiceAbstract*> stdair::FacServiceAbstract::ServicePool_T

Define the list (pool) of Service objects.

Definition at line 20 of file FacServiceAbstract.hpp.

8.67.3 Constructor & Destructor Documentation

8.67.3.1 stdair::FacServiceAbstract::~FacServiceAbstract () [virtual]

Destructor.

Definition at line 13 of file FacServiceAbstract.cpp.

References clean().

8.67.3.2 stdair::FacServiceAbstract::FacServiceAbstract () [inline, protected]

Default Constructor.

This constructor is protected to ensure the class is abstract.

Definition at line 31 of file FacServiceAbstract.hpp.

8.67.4 Member Function Documentation

8.67.4.1 void stdair::FacServiceAbstract::clean ()

Destroyed all the object instantiated by this factory.

Definition at line 18 of file FacServiceAbstract.cpp.

References _pool.

Referenced by ~FacServiceAbstract().

8.67.5 Member Data Documentation

8.67.5.1 ServicePool_T stdair::FacServiceAbstract::_pool [protected]

List of instantiated Business Objects

Definition at line 34 of file FacServiceAbstract.hpp.

Referenced by clean(), and stdair::FacSTDAIRServiceContext::create().

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

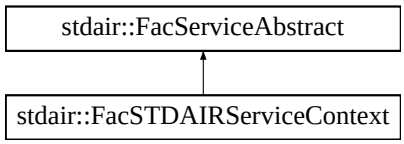
- stdair/service/FacServiceAbstract.hpp
- stdair/service/FacServiceAbstract.cpp

8.68 stdair::FacSTDAIRServiceContext Class Reference

Factory for Bucket.

`#include <stdair/service/FacSTDAIRServiceContext.hpp>`

Inheritance diagram for stdair::FacSTDAIRServiceContext::



Public Types

- typedef std::vector< ServiceAbstract * > ServicePool_T

Public Member Functions

- ~FacSTDAIRServiceContext ()
- STDAIR_ServiceContext & create ()
- void clean ()

Static Public Member Functions

- static FacSTDAIRServiceContext & instance ()

Protected Member Functions

- FacSTDAIRServiceContext ()

Protected Attributes

- ServicePool_T _pool

8.68.1 Detailed Description

Factory for Bucket.

Definition at line 18 of file FacSTDAIRServiceContext.hpp.

8.68.2 Member Typedef Documentation

8.68.2.1 typedef std::vector<ServiceAbstract*> stdair::FacServiceAbstract::ServicePool_T [inherited]

Define the list (pool) of Service objects.

Definition at line 20 of file FacServiceAbstract.hpp.

8.68.3 Constructor & Destructor Documentation

8.68.3.1 stdair::FacSTDAIRServiceContext::~FacSTDAIRServiceContext ()

Destructor.

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

Definition at line 16 of file FacSTDAIRServiceContext.cpp.

8.68.3.2 stdair::FacSTDAIRServiceContext::FacSTDAIRServiceContext () [inline, protected]

Default Constructor.

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

Definition at line 54 of file FacSTDAIRServiceContext.hpp.

Referenced by instance().

8.68.4 Member Function Documentation

8.68.4.1 FacSTDAIRServiceContext & stdair::FacSTDAIRServiceContext::instance () [static]

Provide the unique instance.

The singleton is instantiated when first used.

Returns:

FacSTDAIRServiceContext&

Definition at line 21 of file FacSTDAIRServiceContext.cpp.

References FacSTDAIRServiceContext(), stdair::FacSupervisor::instance(), and stdair::FacSupervisor::registerServiceFactory().

8.68.4.2 STDAIR_ServiceContext & stdair::FacSTDAIRServiceContext::create ()

Create a new STDAIR_ServiceContext object.

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

Returns:

STDAIR_ServiceContext& The newly created object.

Definition at line 33 of file FacSTDAIRServiceContext.cpp.

References stdair::FacServiceAbstract::_pool.

8.68.4.3 void stdair::FacServiceAbstract::clean () [inherited]

Destroyed all the object instantiated by this factory.

Definition at line 18 of file FacServiceAbstract.cpp.

References stdair::FacServiceAbstract::_pool.

Referenced by stdair::FacServiceAbstract::~FacServiceAbstract().

8.68.5 Member Data Documentation

8.68.5.1 ServicePool_T stdair::FacServiceAbstract::_pool [protected, inherited]

List of instantiated Business Objects

Definition at line 34 of file FacServiceAbstract.hpp.

Referenced by stdair::FacServiceAbstract::clean(), and create().

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

- stdair/service/FacSTDAIRServiceContext.hpp
- stdair/service/FacSTDAIRServiceContext.cpp

8.69 stdair::FacSupervisor Class Reference

`#include <stdair/service/FacSupervisor.hpp>`

Public Types

- typedef std::list< FacAbstract * > PersistentBomFactoryPool_T
- typedef std::list< FacAbstract * > CloneBomFactoryPool_T
- typedef std::list< FacServiceAbstract * > ServiceFactoryPool_T

Public Member Functions

- void registerPersistentBomFactory (FacAbstract *)
- void registerCloneBomFactory (FacAbstract *)
- void registerServiceFactory (FacServiceAbstract *)
- void cleanPersistentBomLayer ()
- void cleanCloneBomLayer ()
- void cleanServiceLayer ()
- ~FacSupervisor ()

Static Public Member Functions

- static FacSupervisor & instance ()
- static void cleanLoggerService ()
- static void cleanDBSessionManager ()
- static void cleanAll ()

Protected Member Functions

- FacSupervisor ()
- FacSupervisor (const FacSupervisor &)

8.69.1 Detailed Description

Singleton class to register and clean all Factories.

Definition at line 20 of file FacSupervisor.hpp.

8.69.2 Member Typedef Documentation

8.69.2.1 typedef std::list<FacAbstract*> stdair::FacSupervisor::PersistentBomFactoryPool_T

Define the pool (list) of factories.

Definition at line 25 of file FacSupervisor.hpp.

8.69.2.2 typedef std::list<FacAbstract*> stdair::FacSupervisor::CloneBomFactoryPool_T

Definition at line 26 of file FacSupervisor.hpp.

8.69.2.3 typedef std::list<FacServiceAbstract*> stdair::FacSupervisor::ServiceFactoryPool_T

Definition at line 27 of file FacSupervisor.hpp.

8.69.3 Constructor & Destructor Documentation

8.69.3.1 stdair::FacSupervisor::~FacSupervisor ()

Destructor.

That destructors is applied on the static instance. It then deletes in turn all the other registered objects.

Definition at line 27 of file FacSupervisor.cpp.

References cleanCloneBomLayer(), cleanPersistentBomLayer(), and cleanServiceLayer().

8.69.3.2 stdair::FacSupervisor::FacSupervisor () [inline, protected]

Default Constructor.

This constructor is protected to ensure the singleton pattern.

Definition at line 120 of file FacSupervisor.hpp.

Referenced by instance().

8.69.3.3 stdair::FacSupervisor::FacSupervisor (const FacSupervisor &) [inline, protected]

Definition at line 121 of file FacSupervisor.hpp.

8.69.4 Member Function Documentation

8.69.4.1 FacSupervisor & stdair::FacSupervisor::instance () [static]

Provide the unique (static) instance of the FacSupervisor object.

The singleton is instantiated when first used.

Returns:

FacSupervisor&

Definition at line 18 of file FacSupervisor.cpp.

References FacSupervisor().

Referenced by stdair::STDAIR_Service::clonePersistentBom(), stdair::FacSTDAIRServiceContext::instance(), stdair::FacCloneBom< BOM >::instance(), and stdair::FacBom< BOM >::instance().

8.69.4.2 void stdair::FacSupervisor::registerPersistentBomFactory (FacAbstract *)

Register a newly instantiated persistent factory for the Bom layer.

When a concrete Factory is firstly instantiated this factory have to register itself to the FacSupervisor

Parameters:

FacAbstract* The concrete Factory to register.

Definition at line 34 of file FacSupervisor.cpp.

Referenced by stdair::FacBom< BOM >::instance().

8.69.4.3 void stdair::FacSupervisor::registerCloneBomFactory (FacAbstract *)

Register a newly instantiated concrete factory for the Bom layer.

When a concrete Factory is firstly instantiated this factory have to register itself to the FacSupervisor

Parameters:

FacAbstract* The concrete Factory to register.

Definition at line 39 of file FacSupervisor.cpp.

Referenced by stdair::FacCloneBom< BOM >::instance().

8.69.4.4 void stdair::FacSupervisor::registerServiceFactory (FacServiceAbstract *)

Register a newly instantiated concrete factory for the Service layer.

When a concrete Factory is firstly instantiated this factory have to register itself to the FacSupervisor.

Parameters:

FacServiceAbstract& the concrete Factory to register.

Definition at line 44 of file FacSupervisor.cpp.

Referenced by stdair::FacSTDAIRServiceContext::instance().

8.69.4.5 void stdair::FacSupervisor::cleanPersistentBomLayer ()

Clean all the persistent registered object.

Call the clean method of all the instantiated persistent factories for the BomStructure layer.

Definition at line 49 of file FacSupervisor.cpp.

Referenced by ~FacSupervisor().

8.69.4.6 void stdair::FacSupervisor::cleanCloneBomLayer ()

Clean all the clone registered object.

Call the clean method of all the instantiated factories for the BomStructure layer.

Definition at line 62 of file FacSupervisor.cpp.

Referenced by stdair::STDAIR_Service::clonePersistentBom(), and ~FacSupervisor().

8.69.4.7 void stdair::FacSupervisor::cleanServiceLayer ()

Clean all Service created object.

Call the clean method of all the instantiated factories for the Service layer.

Definition at line 76 of file FacSupervisor.cpp.

Referenced by ~FacSupervisor().

8.69.4.8 void stdair::FacSupervisor::cleanLoggerService () [static]

Delete the static instance of the Logger object.

Definition at line 90 of file FacSupervisor.cpp.

Referenced by cleanAll().

8.69.4.9 void stdair::FacSupervisor::cleanDBSessionManager () [static]

Delete the static instance of the DBSessionManager object.

Definition at line 96 of file FacSupervisor.cpp.

Referenced by cleanAll().

8.69.4.10 void stdair::FacSupervisor::cleanAll () [static]

Clean the static instance. As the static instance (singleton) is deleted, all the other registered objects will be deleted in turn.

Definition at line 102 of file FacSupervisor.cpp.

References cleanDBSessionManager(), and cleanLoggerService().

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

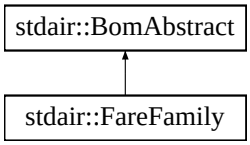
- stdair/service/FacSupervisor.hpp
- stdair/service/FacSupervisor.cpp

8.70 stdair::FareFamily Class Reference

Class representing the actual attributes for a family fare.

`#include <stdair/bom/FareFamily.hpp>`

Inheritance diagram for stdair::FareFamily::



Public Types

- typedef FareFamilyKey Key_T

Public Member Functions

- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const FamilyCode_T & getFamilyCode () const
- const HolderMap_T & getHolderMap () const
- const FRAT5Curve_T & getFrat5Curve () const
- const FFDisutilityCurve_T & getDisutilityCurve () const
- const MeanValue_T & getMean () const
- const StdDevValue_T & getStdDev () const
- const MeanStdDevPairVector_T & getMeanStdDev () const
- void setFrat5Curve (const FRAT5Curve_T &iFRAT5Curve)
- void setDisutilityCurve (const FFDisutilityCurve_T &iDisutilityCurve)
- void setMean (const MeanValue_T &iMean)
- void setStdDev (const StdDevValue_T &iStdDev)
- void setMeanStdDev (const MeanStdDevPairVector_T &iMeanStdDev)
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- template<class Archive> void serialize (Archive &ar, const unsigned int iFileVersion)

Public Attributes

- Key_T _key
- BomAbstract * _parent
- HolderMap_T _holderMap
- FRAT5Curve_T _frat5Curve
- FFDisutilityCurve_T _disutilityCurve
- MeanValue_T _mean
- StdDevValue_T _stdDev
- MeanStdDevPairVector_T _meanStdDev

Protected Member Functions

- FareFamily (const Key_T &)
- virtual ~FareFamily ()

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager
- class boost::serialization::access

8.70.1 Detailed Description

Class representing the actual attributes for a family fare.

Definition at line 28 of file FareFamily.hpp.

8.70.2 Member Typedef Documentation

8.70.2.1 typedef FareFamilyKey stdair::FareFamily::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 39 of file FareFamily.hpp.

8.70.3 Constructor & Destructor Documentation

8.70.3.1 stdair::FareFamily::FareFamily (const Key_T &) [protected]

Constructor.

Definition at line 32 of file FareFamily.cpp.

8.70.3.2 stdair::FareFamily::~FareFamily () [protected, virtual]

Destructor.

Definition at line 36 of file FareFamily.cpp.

8.70.4 Member Function Documentation

8.70.4.1 const Key_T& stdair::FareFamily::getKey () const [inline]

Get the family fare key.

Definition at line 45 of file FareFamily.hpp.

References _key.

8.70.4.2 BomAbstract* const stdair::FareFamily::getParent () const [inline]

Get the parent object.

Definition at line 50 of file FareFamily.hpp.

References _parent.

8.70.4.3 const FamilyCode_T& stdair::FareFamily::getFamilyCode () const [inline]

Get the family fare code (part of the primary key).

Definition at line 55 of file FareFamily.hpp.

References _key, and stdair::FareFamilyKey::getFamilyCode().

8.70.4.4 const HolderMap_T& stdair::FareFamily::getHolderMap () const [inline]

Get the map of children holders.

Definition at line 60 of file FareFamily.hpp.

References _holderMap.

8.70.4.5 const FRAT5Curve_T& stdair::FareFamily::getFrat5Curve () const [inline]

Get the FRAT5 Curve.

Definition at line 65 of file FareFamily.hpp.

References _frat5Curve.

8.70.4.6 const FFDisutilityCurve_T& stdair::FareFamily::getDisutilityCurve () const [inline]

Get the Disutility Curve.

Definition at line 70 of file FareFamily.hpp.

References _disutilityCurve.

8.70.4.7 const MeanValue_T& stdair::FareFamily::getMean () const [inline]

Demand distribution.

Definition at line 75 of file FareFamily.hpp.

References _mean.

8.70.4.8 const StdDevValue_T& stdair::FareFamily::getStdDev () const [inline]

Definition at line 76 of file FareFamily.hpp.

References _stdDev.

8.70.4.9 const MeanStdDevPairVector_T& stdair::FareFamily::getMeanStdDev () const [inline]

Demand distribution.

Definition at line 79 of file FareFamily.hpp.

References _meanStdDev.

8.70.4.10 void stdair::FareFamily::setFrat5Curve (const FRAT5Curve_T & *iFRAT5Curve*) [inline]

FRAT5 Curve.

Definition at line 85 of file FareFamily.hpp.

References _frat5Curve.

8.70.4.11 void stdair::FareFamily::setDisutilityCurve (const FFDisutilityCurve_T & *iDisutilityCurve*) [inline]

Disutility Curve.

Definition at line 90 of file FareFamily.hpp.

References _disutilityCurve.

8.70.4.12 void stdair::FareFamily::setMean (const MeanValue_T & *iMean*) [inline]

Demand distribution.

Definition at line 95 of file FareFamily.hpp.

References _mean.

8.70.4.13 void stdair::FareFamily::setStdDev (const StdDevValue_T & *iStdDev*) [inline]

Definition at line 96 of file FareFamily.hpp.

References _stdDev.

8.70.4.14 void stdair::FareFamily::setMeanStdDev (const MeanStdDevPairVector_T & *iMeanStdDev*) [inline]

Demand distribution.

Definition at line 99 of file FareFamily.hpp.

References _meanStdDev.

8.70.4.15 void stdair::FareFamily::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 111 of file FareFamily.hpp.

References toString().

8.70.4.16 void stdair::FareFamily::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 120 of file FareFamily.hpp.

8.70.4.17 std::string stdair::FareFamily::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 40 of file FareFamily.cpp.

References describeKey().

Referenced by toStream().

8.70.4.18 const std::string stdair::FareFamily::describeKey () const [inline]

Get a string describing the key.

Definition at line 131 of file FareFamily.hpp.

References _key, and stdair::FareFamilyKey::toString().

Referenced by toString().

8.70.4.19 template<class Archive> void stdair::FareFamily::serialize (Archive & *ar*, const unsigned int *iFileVersion*)

Serialisation.

Definition at line 62 of file FareFamily.cpp.

References _key.

8.70.5 Friends And Related Function Documentation

8.70.5.1 friend class FacBom [friend]

Definition at line 29 of file FareFamily.hpp.

8.70.5.2 friend class FacCloneBom [friend]

Definition at line 30 of file FareFamily.hpp.

8.70.5.3 friend class FacBomManager [friend]

Definition at line 31 of file FareFamily.hpp.

8.70.5.4 friend class boost::serialization::access [friend]

Definition at line 32 of file FareFamily.hpp.

8.70.6 Member Data Documentation

8.70.6.1 Key_T stdair::FareFamily::_key

Primary key (fare family code).

Definition at line 184 of file FareFamily.hpp.

Referenced by describeKey(), getFamilyCode(), getKey(), and serialize().

8.70.6.2 BomAbstract* stdair::FareFamily::_parent

Pointer on the parent class (SegmentCabin).

Definition at line 189 of file FareFamily.hpp.

Referenced by getParent().

8.70.6.3 HolderMap_T stdair::FareFamily::_holderMap

Map holding the children (BookingClass objects).

Definition at line 194 of file FareFamily.hpp.

Referenced by getHolderMap().

8.70.6.4 FRAT5Curve_T stdair::FareFamily::_frat5Curve

The associated FRAT5 curve.

Definition at line 199 of file FareFamily.hpp.

Referenced by getFrat5Curve(), and setFrat5Curve().

8.70.6.5 FFDisutilityCurve_T stdair::FareFamily::_disutilityCurve

The associated disutility for the next higher fare family.

Definition at line 204 of file FareFamily.hpp.

Referenced by getDisutilityCurve(), and setDisutilityCurve().

8.70.6.6 MeanValue_T stdair::FareFamily::_mean

Demand distribution forecast.

Definition at line 207 of file FareFamily.hpp.

Referenced by getMean(), and setMean().

8.70.6.7 StdDevValue_T stdair::FareFamily::_stdDev

Definition at line 208 of file FareFamily.hpp.

Referenced by getStdDev(), and setStdDev().

8.70.6.8 MeanStdDevPairVector_T stdair::FareFamily::_meanStdDev

Achievable demand distribution forecast.

Definition at line 213 of file FareFamily.hpp.

Referenced by getMeanStdDev(), and setMeanStdDev().

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

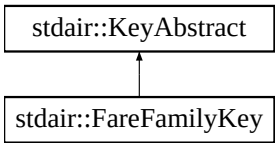
- stdair/bom/FareFamily.hpp
- stdair/bom/FareFamily.cpp

8.71 stdair::FareFamilyKey Struct Reference

Key of a given fare family, made of a fare family code.

`#include <stdair/bom/FareFamilyKey.hpp>`

Inheritance diagram for stdair::FareFamilyKey::



Public Member Functions

- FareFamilyKey (const FamilyCode_T &iFamilyCode)
- FareFamilyKey (const FareFamilyKey &)
- ~FareFamilyKey ()
- const FamilyCode_T & getFamilyCode () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const
- template<class Archive> void serialize (Archive &ar, const unsigned int iFileVersion)

Friends

- class boost::serialization::access

8.71.1 Detailed Description

Key of a given fare family, made of a fare family code.

Definition at line 26 of file FareFamilyKey.hpp.

8.71.2 Constructor & Destructor Documentation

8.71.2.1 stdair::FareFamilyKey::FareFamilyKey (const FamilyCode_T & *iFamilyCode*)

Constructor.

Definition at line 28 of file FareFamilyKey.cpp.

8.71.2.2 stdair::FareFamilyKey::FareFamilyKey (const FareFamilyKey &)

Copy constructor.

Definition at line 23 of file FareFamilyKey.cpp.

8.71.2.3 stdair::FareFamilyKey::~FareFamilyKey ()

Destructor.

Definition at line 33 of file FareFamilyKey.cpp.

8.71.3 Member Function Documentation

8.71.3.1 const FamilyCode_T& stdair::FareFamilyKey::getFamilyCode () const [inline]

Get the family code.

Definition at line 56 of file FareFamilyKey.hpp.

Referenced by stdair::FareFamily::getFamilyCode().

8.71.3.2 void stdair::FareFamilyKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 37 of file FareFamilyKey.cpp.

References toString().

8.71.3.3 void stdair::FareFamilyKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 42 of file FareFamilyKey.cpp.

8.71.3.4 const std::string stdair::FareFamilyKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from stdair::KeyAbstract.

Definition at line 46 of file FareFamilyKey.cpp.

Referenced by stdair::FareFamily::describeKey(), and toStream().

8.71.3.5 template<class Archive> void stdair::FareFamilyKey::serialize (Archive & *ar*, const unsigned int *iFileVersion*)

Serialisation.

Definition at line 68 of file FareFamilyKey.cpp.

8.71.4 Friends And Related Function Documentation

8.71.4.1 friend class boost::serialization::access [friend]

Definition at line 27 of file FareFamilyKey.hpp.

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

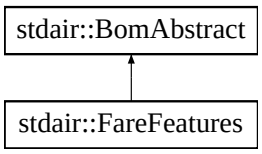
- stdair/bom/FareFamilyKey.hpp
- stdair/bom/FareFamilyKey.cpp

8.72 stdair::FareFeatures Class Reference

Class representing the actual attributes for a fare date-period.

`#include <stdair/bom/FareFeatures.hpp>`

Inheritance diagram for stdair::FareFeatures::



Public Types

- typedef FareFeaturesKey Key_T

Public Member Functions

- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const HolderMap_T & getHolderMap () const
- const TripType_T & getTripType () const
- const DayDuration_T & getAdvancePurchase () const
- const SaturdayStay_T & getSaturdayStay () const
- const ChangeFees_T & getChangeFees () const
- const NonRefundable_T & getRefundableOption () const
- const DayDuration_T & getMinimumStay () const
- bool isTripTypeValid (const TripType_T &) const

- bool isStayDurationValid (const DayDuration_T &) const
- bool isAdvancePurchaseValid (const DateTime_T &iBookingRequestDateTime, const DateTime_T &iFlightDateTime) const

Protected Member Functions

- FareFeatures (const Key_T &)
- virtual ~FareFeatures ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent
- HolderMap_T _holderMap

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager

8.72.1 Detailed Description

Class representing the actual attributes for a fare date-period.

Definition at line 18 of file FareFeatures.hpp.

8.72.2 Member Typedef Documentation

8.72.2.1 typedef FareFeaturesKey stdair::FareFeatures::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 28 of file FareFeatures.hpp.

8.72.3 Constructor & Destructor Documentation

8.72.3.1 stdair::FareFeatures::FareFeatures (const Key_T &) [protected]

Main constructor.

Definition at line 33 of file FareFeatures.cpp.

8.72.3.2 stdair::FareFeatures::~FareFeatures () [protected, virtual]

Destructor.

Definition at line 38 of file FareFeatures.cpp.

8.72.4 Member Function Documentation

8.72.4.1 void stdair::FareFeatures::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 37 of file FareFeatures.hpp.

References toString().

8.72.4.2 void stdair::FareFeatures::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 46 of file FareFeatures.hpp.

8.72.4.3 std::string stdair::FareFeatures::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 42 of file FareFeatures.cpp.

References describeKey().

Referenced by toStream().

8.72.4.4 const std::string stdair::FareFeatures::describeKey () const [inline]

Get a string describing the key.

Definition at line 57 of file FareFeatures.hpp.

References _key, and stdair::FareFeaturesKey::toString().

Referenced by toString().

8.72.4.5 const Key_T& stdair::FareFeatures::getKey () const [inline]

Get the primary key (trip type, advance purchase,... ,cabin code).

Definition at line 67 of file FareFeatures.hpp.

References _key.

8.72.4.6 BomAbstract* const stdair::FareFeatures::getParent () const [inline]

Get a reference on the parent object instance.

Definition at line 74 of file FareFeatures.hpp.

References _parent.

8.72.4.7 const HolderMap_T& stdair::FareFeatures::getHolderMap () const [inline]

Get a reference on the children holder.

Definition at line 81 of file FareFeatures.hpp.

References _holderMap.

8.72.4.8 const TripType_T& stdair::FareFeatures::getTripType () const [inline]

Get the trip type.

Definition at line 88 of file FareFeatures.hpp.

References _key, and stdair::FareFeaturesKey::getTripType().

Referenced by isTripTypeValid().

8.72.4.9 const DayDuration_T& stdair::FareFeatures::getAdvancePurchase () const [inline]

Get the fare day duration.

Definition at line 95 of file FareFeatures.hpp.

References _key, and stdair::FareFeaturesKey::getAdvancePurchase().

Referenced by isAdvancePurchaseValid().

8.72.4.10 const SaturdayStay_T& stdair::FareFeatures::getSaturdayStay () const [inline]

Get the fare saturday stay option.

Definition at line 102 of file FareFeatures.hpp.

References _key, and stdair::FareFeaturesKey::getSaturdayStay().

8.72.4.11 const ChangeFees_T& stdair::FareFeatures::getChangeFees () const [inline]

Get the change fees criterion.

Definition at line 109 of file FareFeatures.hpp.

References _key, and stdair::FareFeaturesKey::getChangeFees().

8.72.4.12 const NonRefundable_T& stdair::FareFeatures::getRefundableOption () const [inline]

Get the refundable option.

Definition at line 116 of file FareFeatures.hpp.

References _key, and stdair::FareFeaturesKey::getRefundableOption().

8.72.4.13 const DayDuration_T& stdair::FareFeatures::getMinimumStay () const [inline]

Get the minimum stay.

Definition at line 123 of file FareFeatures.hpp.

References _key, and stdair::FareFeaturesKey::getMinimumStay().

Referenced by isStayDurationValid().

8.72.4.14 bool stdair::FareFeatures::isTripTypeValid (const TripType_T &) const

Check whether the fare rule trip type corresponds to the booking request trip type.

Definition at line 50 of file FareFeatures.cpp.

References getTripType(), stdair::TRIP_TYPE_INBOUND, stdair::TRIP_TYPE_OUTBOUND, and stdair::TRIP_TYPE_ROUND_TRIP.

8.72.4.15 bool stdair::FareFeatures::isStayDurationValid (const DayDuration_T &) const

Check whether a given stay duration is greater or equal to the minimum stay of the fare rule.

Definition at line 75 of file FareFeatures.cpp.

References getMinimumStay().

8.72.4.16 bool stdair::FareFeatures::isAdvancePurchaseValid (const DateTime_T & *iBookingRequestDateTime*, const DateTime_T & *iFlightDateTime*) const

Check whether a booking request date is valid compared the required advance purchase number of days of the fare rule.

Definition at line 88 of file FareFeatures.cpp.

References getAdvancePurchase().

8.72.5 Friends And Related Function Documentation

8.72.5.1 friend class FacBom [friend]

Definition at line 19 of file FareFeatures.hpp.

8.72.5.2 friend class FacCloneBom [friend]

Definition at line 20 of file FareFeatures.hpp.

8.72.5.3 friend class FacBomManager [friend]

Definition at line 21 of file FareFeatures.hpp.

8.72.6 Member Data Documentation

8.72.6.1 Key_T stdair::FareFeatures::_key [protected]

Primary key (flight number and departure date).

Definition at line 176 of file FareFeatures.hpp.

Referenced by describeKey(), getAdvancePurchase(), getChangeFees(), getKey(), getMinimumStay(), getRefundableOption(), getSaturdayStay(), and getTripType().

8.72.6.2 BomAbstract* stdair::FareFeatures::_parent [protected]

Pointer on the parent class.

Definition at line 181 of file FareFeatures.hpp.

Referenced by getParent().

8.72.6.3 HolderMap_T stdair::FareFeatures::_holderMap [protected]

Map holding the children.

Definition at line 186 of file FareFeatures.hpp.

Referenced by getHolderMap().

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

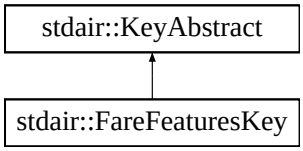
- stdair/bom/FareFeatures.hpp
- stdair/bom/FareFeatures.cpp

8.73 stdair::FareFeaturesKey Struct Reference

Key of date-period.

`#include <stdair/bom/FareFeaturesKey.hpp>`

Inheritance diagram for stdair::FareFeaturesKey::



Public Member Functions

- FareFeaturesKey (const TripType_T &, const DayDuration_T &, const SaturdayStay_T &, const ChangeFees_T &, const NonRefundable_T &, const DayDuration_T &)
- FareFeaturesKey (const FareFeaturesKey &)
- ~FareFeaturesKey ()
- const TripType_T & getTripType () const
- const DayDuration_T & getAdvancePurchase () const
- const SaturdayStay_T & getSaturdayStay () const
- const ChangeFees_T & getChangeFees () const
- const NonRefundable_T & getRefundableOption () const
- const DayDuration_T & getMinimumStay () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const

8.73.1 Detailed Description

Key of date-period.

Definition at line 18 of file FareFeaturesKey.hpp.

8.73.2 Constructor & Destructor Documentation

8.73.2.1 stdair::FareFeaturesKey::FareFeaturesKey (const TripType_T &, const DayDuration_T &, const SaturdayStay_T &, const ChangeFees_T &, const NonRefundable_T &, const DayDuration_T &)

Main constructor.

Definition at line 26 of file FareFeaturesKey.cpp.

8.73.2.2 stdair::FareFeaturesKey::FareFeaturesKey (const FareFeaturesKey &)

Copy constructor.

Definition at line 38 of file FareFeaturesKey.cpp.

8.73.2.3 stdair::FareFeaturesKey::~FareFeaturesKey ()

Destructor.

Definition at line 48 of file FareFeaturesKey.cpp.

8.73.3 Member Function Documentation

8.73.3.1 const TripType_T& stdair::FareFeaturesKey::getTripType () const [inline]

Get the fare trip type.

Definition at line 39 of file FareFeaturesKey.hpp.

Referenced by stdair::FareFeatures::getTripType().

8.73.3.2 const DayDuration_T& stdair::FareFeaturesKey::getAdvancePurchase () const [inline]

Get the fare day duration.

Definition at line 46 of file FareFeaturesKey.hpp.

Referenced by stdair::FareFeatures::getAdvancePurchase().

8.73.3.3 const SaturdayStay_T& stdair::FareFeaturesKey::getSaturdayStay () const [inline]

Get the fare saturday stay option.

Definition at line 53 of file FareFeaturesKey.hpp.

Referenced by stdair::FareFeatures::getSaturdayStay().

8.73.3.4 const ChangeFees_T& stdair::FareFeaturesKey::getChangeFees () const [inline]

Get the change fees criterion.

Definition at line 60 of file FareFeaturesKey.hpp.

Referenced by stdair::FareFeatures::getChangeFees().

8.73.3.5 const NonRefundable_T& stdair::FareFeaturesKey::getRefundableOption () const [inline]

Get the refundable option.

Definition at line 67 of file FareFeaturesKey.hpp.

Referenced by stdair::FareFeatures::getRefundableOption().

8.73.3.6 const DayDuration_T& stdair::FareFeaturesKey::getMinimumStay () const [inline]

Get the minimum stay.

Definition at line 74 of file FareFeaturesKey.hpp.

Referenced by stdair::FareFeatures::getMinimumStay().

8.73.3.7 void stdair::FareFeaturesKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 52 of file FareFeaturesKey.cpp.

References toString().

8.73.3.8 void stdair::FareFeaturesKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 57 of file FareFeaturesKey.cpp.

8.73.3.9 const std::string stdair::FareFeaturesKey::toString () const [virtual]

Get the serialised version of the Business Object Key. That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

Reimplemented from stdair::KeyAbstract.

Definition at line 61 of file FareFeaturesKey.cpp.

Referenced by stdair::FareFeatures::describeKey(), and toStream().

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

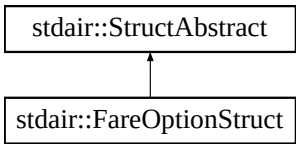
- stdair/bom/FareFeaturesKey.hpp
- stdair/bom/FareFeaturesKey.cpp

8.74 stdair::FareOptionStruct Struct Reference

Structure holding the elements of a fare option.

`#include <stdair/bom/FareOptionStruct.hpp>`

Inheritance diagram for stdair::FareOptionStruct::



Public Member Functions

- const ClassList_StringList_T & getClassPath () const
- const Fare_T & getFare () const
- const Availability_T & getAvailability () const
- const ChangeFees_T getChangeFees () const
- const NonRefundable_T getNonRefundable () const
- const SaturdayStay_T getSaturdayStay () const
- void addClassList (const std::string)
- void emptyClassList ()
- void setFare (const Fare_T &iFare)
- void setAvailability (const Availability_T &iAvl)
- void setChangeFees (const ChangeFees_T iRes)
- void setNonRefundable (const NonRefundable_T iRes)
- void setSaturdayStay (const SaturdayStay_T iRes)
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string describe () const
- const std::string display () const
- FareOptionStruct ()
- FareOptionStruct (const std::string &iClassPath, const Fare_T &, const ChangeFees_T &, const NonRefundable_T &, const SaturdayStay_T &)
- FareOptionStruct (const FareOptionStruct &)
- ~FareOptionStruct ()

8.74.1 Detailed Description

Structure holding the elements of a fare option.

Definition at line 20 of file FareOptionStruct.hpp.

8.74.2 Constructor & Destructor Documentation

8.74.2.1 stdair::FareOptionStruct::FareOptionStruct ()

Default constructor.

Definition at line 14 of file FareOptionStruct.cpp.

8.74.2.2 stdair::FareOptionStruct::FareOptionStruct (const std::string & *iClassPath*, const Fare_T &, const ChangeFees_T &, const NonRefundable_T &, const SaturdayStay_T &)

Main constructor.

Definition at line 26 of file FareOptionStruct.cpp.

8.74.2.3 stdair::FareOptionStruct::FareOptionStruct (const FareOptionStruct &)

Copy constructor.

Definition at line 19 of file FareOptionStruct.cpp.

8.74.2.4 stdair::FareOptionStruct::~FareOptionStruct ()

Destructor.

Definition at line 38 of file FareOptionStruct.cpp.

8.74.3 Member Function Documentation

8.74.3.1 const ClassList_StringList_T& stdair::FareOptionStruct::getClassPath () const [inline]

Get the class-path.

Definition at line 24 of file FareOptionStruct.hpp.

8.74.3.2 const Fare_T& stdair::FareOptionStruct::getFare () const [inline]

Get the fare value.

Definition at line 29 of file FareOptionStruct.hpp.

8.74.3.3 const Availability_T& stdair::FareOptionStruct::getAvailability () const [inline]

Get the availability.

Definition at line 34 of file FareOptionStruct.hpp.

8.74.3.4 const ChangeFees_T stdair::FareOptionStruct::getChangeFees () const [inline]

Get the change fees.

Definition at line 39 of file FareOptionStruct.hpp.

8.74.3.5 const NonRefundable_T stdair::FareOptionStruct::getNonRefundable () const [inline]

State whether the ticket is refundable.

Definition at line 44 of file FareOptionStruct.hpp.

8.74.3.6 const SaturdayStay_T stdair::FareOptionStruct::getSaturdayStay () const [inline]

State whether there is a condition on the saturday night stay.

Definition at line 49 of file FareOptionStruct.hpp.

8.74.3.7 void stdair::FareOptionStruct::addClassList (const std::string)

Set the class-path.

Definition at line 93 of file FareOptionStruct.cpp.

8.74.3.8 void stdair::FareOptionStruct::emptyClassList ()

Empty the class-path.

Definition at line 98 of file FareOptionStruct.cpp.

8.74.3.9 void stdair::FareOptionStruct::setFare (const Fare_T & *iFare*) [inline]

Set the fare value.

Definition at line 63 of file FareOptionStruct.hpp.

8.74.3.10 void stdair::FareOptionStruct::setAvailability (const Availability_T & *iAvl*) [inline]

Set the availability.

Definition at line 68 of file FareOptionStruct.hpp.

8.74.3.11 void stdair::FareOptionStruct::setChangeFees (const ChangeFees_T *iRes*) [inline]

Set the change fees.

Definition at line 73 of file FareOptionStruct.hpp.

8.74.3.12 void stdair::FareOptionStruct::setNonRefundable (const NonRefundable_T *iRes*) [inline]

Set the flag for the ticket refundability.

Definition at line 78 of file FareOptionStruct.hpp.

8.74.3.13 void stdair::FareOptionStruct::setSaturdayStay (const SaturdayStay_T *iRes*) [inline]

Set the flag for the saturday night stay condition.

Definition at line 83 of file FareOptionStruct.hpp.

8.74.3.14 void stdair::FareOptionStruct::toStream (std::ostream & *ioOut*) const

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::StructAbstract.

Definition at line 42 of file FareOptionStruct.cpp.

References describe().

8.74.3.15 void stdair::FareOptionStruct::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::StructAbstract.

Definition at line 47 of file FareOptionStruct.cpp.

8.74.3.16 const std::string stdair::FareOptionStruct::describe () const [virtual]

Display of the structure.

Implements stdair::StructAbstract.

Definition at line 51 of file FareOptionStruct.cpp.

Referenced by stdair::TravelSolutionStruct::describe(), and toStream().

8.74.3.17 const std::string stdair::FareOptionStruct::display () const

Display of the structure.

Definition at line 73 of file FareOptionStruct.cpp.

Referenced by stdair::TravelSolutionStruct::display().

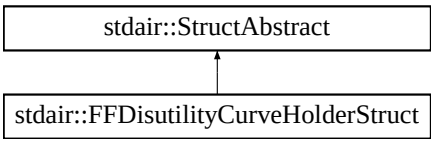
The documentation for this struct was generated from the following files:

- stdair/bom/FareOptionStruct.hpp
- stdair/bom/FareOptionStruct.cpp

8.75 stdair::FFDisutilityCurveHolderStruct Struct Reference

`#include <stdair/bom/FFDisutilityCurveHolderStruct.hpp>`

Inheritance diagram for stdair::FFDisutilityCurveHolderStruct::



Public Member Functions

- const FFDisutilityCurve_T & getFFDisutilityCurve (const std::string &) const
- void addCurve (const std::string &, const FFDisutilityCurve_T &)
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string describe () const
- FFDisutilityCurveHolderStruct ()
- FFDisutilityCurveHolderStruct (const FFDisutilityCurveHolderStruct &)
- ~FFDisutilityCurveHolderStruct ()

8.75.1 Detailed Description

Structure holding the elements of a snapshot .

Definition at line 19 of file FFDisutilityCurveHolderStruct.hpp.

8.75.2 Constructor & Destructor Documentation

8.75.2.1 stdair::FFDisutilityCurveHolderStruct::FFDisutilityCurveHolderStruct ()

Constructor.

Definition at line 14 of file FFDisutilityCurveHolderStruct.cpp.

8.75.2.2 stdair::FFDisutilityCurveHolderStruct::FFDisutilityCurveHolderStruct (const FFDisutilityCurveHolderStruct &)

Copy constructor.

Definition at line 19 of file FFDisutilityCurveHolderStruct.cpp.

8.75.2.3 stdair::FFDisutilityCurveHolderStruct::~FFDisutilityCurveHolderStruct ()

Destructor.

Definition at line 24 of file FFDisutilityCurveHolderStruct.cpp.

8.75.3 Member Function Documentation

8.75.3.1 const FFDisutilityCurve_T & stdair::FFDisutilityCurveHolderStruct::getFFDisutilityCurve (const std::string &) const

Get the FFDisutility curve corresponding to the given key.

Definition at line 29 of file FFDisutilityCurveHolderStruct.cpp.

References STDAIR_LOG_DEBUG.

Referenced by stdair::BomRoot::getFFDisutilityCurve().

8.75.3.2 void stdair::FFDisutilityCurveHolderStruct::addCurve (const std::string &, const FFDisutilityCurve_T &)

Add a new curve to the holder.

Definition at line 42 of file FFDisutilityCurveHolderStruct.cpp.

References STDAIR_LOG_DEBUG.

Referenced by stdair::BomRoot::addFFDisutilityCurve().

8.75.3.3 void stdair::FFDisutilityCurveHolderStruct::toStream (std::ostream & *ioOut*) const

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::StructAbstract.

Definition at line 53 of file FFDisutilityCurveHolderStruct.cpp.

References describe().

8.75.3.4 void stdair::FFDisutilityCurveHolderStruct::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::StructAbstract.

Definition at line 58 of file FFDisutilityCurveHolderStruct.cpp.

8.75.3.5 const std::string stdair::FFDisutilityCurveHolderStruct::describe () const [virtual]

Display of the structure.

Implements stdair::StructAbstract.

Definition at line 62 of file FFDisutilityCurveHolderStruct.cpp.

Referenced by toStream().

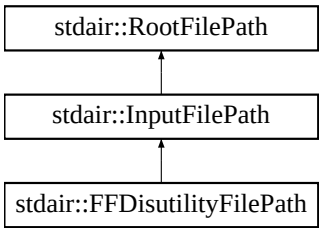
The documentation for this struct was generated from the following files:

- stdair/bom/FFDisutilityCurveHolderStruct.hpp
- stdair/bom/FFDisutilityCurveHolderStruct.cpp

8.76 stdair::FFDisutilityFilePath Class Reference

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

Inheritance diagram for stdair::FFDisutilityFilePath::



Public Member Functions

- FFDisutilityFilePath (const Filename_T &iFilename)
- const char * name () const

Protected Attributes

- const Filename_T _filename

8.76.1 Detailed Description

FFDisutility input file.

Definition at line 100 of file stdair_file.hpp.

8.76.2 Constructor & Destructor Documentation

8.76.2.1 stdair::FFDisutilityFilePath::FFDisutilityFilePath (const Filename_T & *iFilename*) [inline, explicit]

Constructor.

Definition at line 105 of file stdair_file.hpp.

8.76.3 Member Function Documentation

8.76.3.1 const char* stdair::RootFilePath::name () const [inline, inherited]

Give the details of the exception.

Definition at line 42 of file stdair_file.hpp.

References stdair::RootFilePath::_filename.

Referenced by stdair::BomINIImport::importINIConfig().

8.76.4 Member Data Documentation

8.76.4.1 const Filename_T stdair::RootFilePath::_filename [protected, inherited]

Name of the file.

Definition at line 50 of file stdair_file.hpp.

Referenced by stdair::RootFilePath::name().

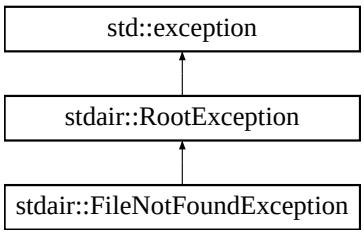
The documentation for this class was generated from the following file:

- stdair/stdair_file.hpp

8.77 stdair::FileNotFoundException Class Reference

`#include <stdair/stdair_exceptions.hpp>`

Inheritance diagram for stdair::FileNotFoundException::



Public Member Functions

- FileNotFoundException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

8.77.1 Detailed Description

File not found.

Definition at line 50 of file stdair_exceptions.hpp.

8.77.2 Constructor & Destructor Documentation

8.77.2.1 stdair::FileNotFoundException::FileNotFoundException (const std::string & *iWhat*) `[inline]`

Constructor.

Definition at line 53 of file stdair_exceptions.hpp.

8.77.3 Member Function Documentation

8.77.3.1 const char* stdair::RootException::what () const throw () `[inline, inherited]`

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

8.77.4 Member Data Documentation

8.77.4.1 std::string stdair::RootException::_what [protected, inherited]

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

Referenced by stdair::RootException::what().

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

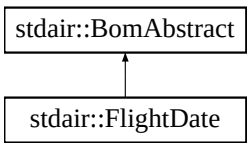
- stdair/stdair_exceptions.hpp

8.78 stdair::FlightDate Class Reference

Class representing the actual attributes for an airline flight-date.

`#include <stdair/bom/FlightDate.hpp>`

Inheritance diagram for stdair::FlightDate::



Public Types

- typedef FlightDateKey Key_T

Public Member Functions

- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const FlightNumber_T & getFlightNumber () const
- const Date_T & getDepartureDate () const
- const AirlineCode_T & getAirlineCode () const
- const HolderMap_T & getHolderMap () const
- LegDate * getLegDate (const std::string &iLegDateKeyStr) const
- LegDate * getLegDate (const LegDateKey &) const
- SegmentDate * getSegmentDate (const std::string &iSegmentDateKeyStr) const
- SegmentDate * getSegmentDate (const SegmentDateKey &) const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- template<class Archive> void serialize (Archive &ar, const unsigned int iFileVersion)

Protected Member Functions

- FlightDate (const Key_T &)
- virtual ~FlightDate ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent
- HolderMap_T _holderMap

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager
- class boost::serialization::access

8.78.1 Detailed Description

Class representing the actual attributes for an airline flight-date.

Definition at line 35 of file FlightDate.hpp.

8.78.2 Member Typedef Documentation

8.78.2.1 typedef FlightDateKey stdair::FlightDate::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 46 of file FlightDate.hpp.

8.78.3 Constructor & Destructor Documentation

8.78.3.1 stdair::FlightDate::FlightDate (const Key_T &) [protected]

Main constructor.

Definition at line 29 of file FlightDate.cpp.

8.78.3.2 stdair::FlightDate::~FlightDate () [protected, virtual]

Destructor.

Definition at line 33 of file FlightDate.cpp.

8.78.4 Member Function Documentation

8.78.4.1 const Key_T& stdair::FlightDate::getKey () const [inline]

Get the flight-date key.

Definition at line 52 of file FlightDate.hpp.

References _key.

8.78.4.2 BomAbstract* const stdair::FlightDate::getParent () const [inline]

Get the parent object.

Definition at line 57 of file FlightDate.hpp.

References _parent.

Referenced by getAirlineCode().

8.78.4.3 const FlightNumber_T& stdair::FlightDate::getFlightNumber () const [inline]

Get the flight number (part of the primary key).

Definition at line 62 of file FlightDate.hpp.

References _key, and stdair::FlightDateKey::getFlightNumber().

Referenced by stdair::BomJSONExport::jsonExportFlightDateObjects().

8.78.4.4 const Date_T& stdair::FlightDate::getDepartureDate () const [inline]

Get the flight date (part of the primary key).

Definition at line 67 of file FlightDate.hpp.

References _key, and stdair::FlightDateKey::getDepartureDate().

Referenced by stdair::LegDate::describeRoutingKey(), and stdair::BomJSONExport::jsonExportFlightDateObjects().

8.78.4.5 const AirlineCode_T & stdair::FlightDate::getAirlineCode () const

Get the airline code (key of the parent object).

Note:

That method assumes that the parent object derives from the Inventory class, as it needs to have access to the getAirlineCode() method.

Definition at line 37 of file FlightDate.cpp.

References stdair::Inventory::getAirlineCode(), and getParent().

Referenced by stdair::LegDate::getAirlineCode(), and stdair::BomJSONExport::jsonExportFlightDateObjects().

8.78.4.6 const HolderMap_T& stdair::FlightDate::getHolderMap () const [inline]

Get the map of children holders.

Definition at line 83 of file FlightDate.hpp.

References _holderMap.

8.78.4.7 LegDate * stdair::FlightDate::getLegDate (const std::string & *iLegDateKeyStr*) const

Get a pointer on the LegDate object corresponding to the given key.

Note:

The LegDate object can be inherited from, if needed. In that case, a dynamic_cast<> may be needed.

Parameters:

const std::string& The leg-date key.

Returns:

LegDate* Found LegDate object. NULL if not found.

Definition at line 52 of file FlightDate.cpp.

Referenced by getLegDate(), and stdair::BomRetriever::retrieveOperatingLegDateFromLongKey().

8.78.4.8 LegDate * stdair::FlightDate::getLegDate (const LegDateKey &) const

Get a pointer on the LegDate object corresponding to the given key.

Note:

The LegDate object can be inherited from, if needed. In that case, a dynamic_cast<> may be needed.

Parameters:

const LegDateKey& The leg-date key

Returns:

LegDate* Found LegDate object. NULL if not found.

Definition at line 59 of file FlightDate.cpp.

References getLegDate(), and stdair::LegDateKey::toString().

8.78.4.9 SegmentDate * stdair::FlightDate::getSegmentDate (const std::string & *iSegmentDateKeyStr*) const

Get a pointer on the SegmentDate object corresponding to the given key.

Note:

The SegmentDate object can be inherited from, if needed. In that case, a dynamic_cast<> may be needed.

Parameters:

const std::string& The segment-date key.

Returns:

SegmentDate* Found SegmentDate object. NULL if not found.

Definition at line 65 of file FlightDate.cpp.

Referenced by getSegmentDate(), stdair::BomRetriever::retrieveSegmentDateFromKey(), and stdair::BomRetriever::retrieveSegmentDateFromLongKey().

8.78.4.10 SegmentDate * stdair::FlightDate::getSegmentDate (const SegmentDateKey &) const

Get a pointer on the SegmentDate object corresponding to the given key.

Note:

The SegmentDate object can be inherited from, if needed. In that case, a dynamic_cast<> may be needed.

Parameters:

const SegmentDateKey& The segment-date key

Returns:

SegmentDate* Found SegmentDate object. NULL if not found.

Definition at line 73 of file FlightDate.cpp.

References getSegmentDate(), and stdair::SegmentDateKey::toString().

8.78.4.11 void stdair::FlightDate::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 142 of file FlightDate.hpp.

References toString().

8.78.4.12 void stdair::FlightDate::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 151 of file FlightDate.hpp.

8.78.4.13 std::string stdair::FlightDate::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 45 of file FlightDate.cpp.

References describeKey().

Referenced by toStream().

8.78.4.14 const std::string stdair::FlightDate::describeKey () const [inline]

Get a string describing the key.

Definition at line 162 of file FlightDate.hpp.

References _key, and stdair::FlightDateKey::toString().

Referenced by toString().

8.78.4.15 template<class Archive> void stdair::FlightDate::serialize (Archive & *ar*, const unsigned int *iFileVersion*)

Serialisation.

Definition at line 187 of file CmdBomSerialiser.cpp.

References _key.

8.78.5 Friends And Related Function Documentation

8.78.5.1 friend class FacBom [friend]

Definition at line 36 of file FlightDate.hpp.

8.78.5.2 friend class FacCloneBom [friend]

Definition at line 37 of file FlightDate.hpp.

8.78.5.3 friend class FacBomManager [friend]

Definition at line 38 of file FlightDate.hpp.

8.78.5.4 friend class boost::serialization::access [friend]

Definition at line 39 of file FlightDate.hpp.

8.78.6 Member Data Documentation

8.78.6.1 Key_T stdair::FlightDate::_key [protected]

Primary key (flight number and departure date).

Definition at line 216 of file FlightDate.hpp.

Referenced by describeKey(), getDepartureDate(), getFlightNumber(), getKey(), and serialize().

8.78.6.2 BomAbstract* stdair::FlightDate::_parent [protected]

Pointer on the parent class (Inventory).

Definition at line 221 of file FlightDate.hpp.

Referenced by getParent().

8.78.6.3 HolderMap_T stdair::FlightDate::_holderMap [protected]

Map holding the children (SegmentDate and LegDate objects).

Definition at line 226 of file FlightDate.hpp.

Referenced by getHolderMap().

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

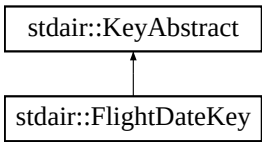
- stdair/bom/FlightDate.hpp
- stdair/bom/FlightDate.cpp
- stdair/command/CmdBomSerialiser.cpp

8.79 stdair::FlightDateKey Struct Reference

Key of a given flight-date, made of a flight number and a departure date.

`#include <stdair/bom/FlightDateKey.hpp>`

Inheritance diagram for stdair::FlightDateKey::



Public Member Functions

- FlightDateKey (const FlightNumber_T &, const Date_T &)
- FlightDateKey (const FlightDateKey &)
- ~FlightDateKey ()
- const FlightNumber_T & getFlightNumber () const
- const Date_T & getDepartureDate () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const
- template<class Archive> void serialize (Archive &ar, const unsigned int iFileVersion)

Friends

- class boost::serialization::access

8.79.1 Detailed Description

Key of a given flight-date, made of a flight number and a departure date.

Definition at line 28 of file FlightDateKey.hpp.

8.79.2 Constructor & Destructor Documentation

8.79.2.1 stdair::FlightDateKey::FlightDateKey (const FlightNumber_T &, const Date_T &)

Constructor.

Definition at line 28 of file FlightDateKey.cpp.

8.79.2.2 stdair::FlightDateKey::FlightDateKey (const FlightDateKey &)

Copy constructor.

Definition at line 34 of file FlightDateKey.cpp.

8.79.2.3 stdair::FlightDateKey::~FlightDateKey ()

Destructor.

Definition at line 39 of file FlightDateKey.cpp.

8.79.3 Member Function Documentation

8.79.3.1 const FlightNumber_T& stdair::FlightDateKey::getFlightNumber () const [inline]

Get the flight number.

Definition at line 58 of file FlightDateKey.hpp.

Referenced by stdair::FlightDate::getFlightNumber().

8.79.3.2 const Date_T& stdair::FlightDateKey::getDepartureDate () const [inline]

Get the departure date of the (first leg of the) flight.

Definition at line 63 of file FlightDateKey.hpp.

Referenced by stdair::OnDDateKey::getDate(), and stdair::FlightDate::getDepartureDate().

8.79.3.3 void stdair::FlightDateKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 43 of file FlightDateKey.cpp.

References toString().

8.79.3.4 void stdair::FlightDateKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 48 of file FlightDateKey.cpp.

8.79.3.5 const std::string stdair::FlightDateKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from stdair::KeyAbstract.

Definition at line 52 of file FlightDateKey.cpp.

References stdair::DEFAULT_KEY_SUB_FLD_DELIMITER.

Referenced by stdair::FlightDate::describeKey(), stdair::Inventory::getFlightDate(), stdair::BomRetriever::retrieveSegmentDateFromLongKey(), and toStream().

8.79.3.6 template<class Archive> void stdair::FlightDateKey::serialize (Archive & *ar*, const unsigned int *iFileVersion*)

Serialisation.

Definition at line 77 of file FlightDateKey.cpp.

8.79.4 Friends And Related Function Documentation

8.79.4.1 friend class boost::serialization::access [friend]

Definition at line 29 of file FlightDateKey.hpp.

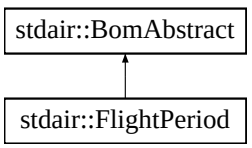
The documentation for this struct was generated from the following files:

- stdair/bom/FlightDateKey.hpp
- stdair/bom/FlightDateKey.cpp

8.80 stdair::FlightPeriod Class Reference

`#include <stdair/bom/FlightPeriod.hpp>`

Inheritance diagram for stdair::FlightPeriod::



Public Types

- typedef FlightPeriodKey Key_T

Public Member Functions

- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const FlightNumber_T & getFlightNumber () const
- const PeriodStruct & getPeriod () const
- const HolderMap_T & getHolderMap () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const

Protected Member Functions

- FlightPeriod (const Key_T &)
- ~FlightPeriod ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent
- HolderMap_T _holderMap

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager

8.80.1 Detailed Description

Class representing the actual attributes for an airline flight-period.

Definition at line 15 of file FlightPeriod.hpp.

8.80.2 Member Typedef Documentation

8.80.2.1 typedef FlightPeriodKey stdair::FlightPeriod::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 23 of file FlightPeriod.hpp.

8.80.3 Constructor & Destructor Documentation

8.80.3.1 stdair::FlightPeriod::FlightPeriod (const Key_T &) [protected]

Main constructor.

Definition at line 12 of file FlightPeriod.cpp.

8.80.3.2 stdair::FlightPeriod::~FlightPeriod () [protected]

Destructor.

Definition at line 22 of file FlightPeriod.cpp.

8.80.4 Member Function Documentation

8.80.4.1 const Key_T& stdair::FlightPeriod::getKey () const [inline]

Get the flight-period key.

Definition at line 28 of file FlightPeriod.hpp.

References _key.

8.80.4.2 BomAbstract* const stdair::FlightPeriod::getParent () const [inline]

Get the parent object.

Definition at line 31 of file FlightPeriod.hpp.

References _parent.

8.80.4.3 const FlightNumber_T& stdair::FlightPeriod::getFlightNumber () const [inline]

Get the flight number (part of the primary key).

Definition at line 34 of file FlightPeriod.hpp.

References _key, and stdair::FlightPeriodKey::getFlightNumber().

8.80.4.4 const PeriodStruct& stdair::FlightPeriod::getPeriod () const [inline]

Get the departure period (part of the key).

Definition at line 39 of file FlightPeriod.hpp.

References _key, and stdair::FlightPeriodKey::getPeriod().

8.80.4.5 const HolderMap_T& stdair::FlightPeriod::getHolderMap () const [inline]

Get the map of children holders.

Definition at line 42 of file FlightPeriod.hpp.

References _holderMap.

8.80.4.6 void stdair::FlightPeriod::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 49 of file FlightPeriod.hpp.

References toString().

8.80.4.7 void stdair::FlightPeriod::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 53 of file FlightPeriod.hpp.

8.80.4.8 std::string stdair::FlightPeriod::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 26 of file FlightPeriod.cpp.

References describeKey().

Referenced by toStream().

8.80.4.9 const std::string stdair::FlightPeriod::describeKey () const [inline]

Get a string describing the key.

Definition at line 59 of file FlightPeriod.hpp.

References _key, and stdair::FlightPeriodKey::toString().

Referenced by toString().

8.80.5 Friends And Related Function Documentation

8.80.5.1 friend class FacBom [friend]

Definition at line 16 of file FlightPeriod.hpp.

8.80.5.2 friend class FacCloneBom [friend]

Definition at line 17 of file FlightPeriod.hpp.

8.80.5.3 friend class FacBomManager [friend]

Definition at line 18 of file FlightPeriod.hpp.

8.80.6 Member Data Documentation

8.80.6.1 Key_T stdair::FlightPeriod::_key [protected]

Definition at line 86 of file FlightPeriod.hpp.

Referenced by describeKey(), getFlightNumber(), getKey(), and getPeriod().

8.80.6.2 BomAbstract* stdair::FlightPeriod::_parent [protected]

Definition at line 87 of file FlightPeriod.hpp.

Referenced by getParent().

8.80.6.3 HolderMap_T stdair::FlightPeriod::_holderMap [protected]

Definition at line 88 of file FlightPeriod.hpp.

Referenced by getHolderMap().

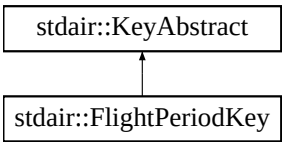
The documentation for this class was generated from the following files:

- stdair/bom/FlightPeriod.hpp
- stdair/bom/FlightPeriod.cpp

8.81 stdair::FlightPeriodKey Struct Reference

`#include <stdair/bom/FlightPeriodKey.hpp>`

Inheritance diagram for stdair::FlightPeriodKey::



Public Member Functions

- FlightPeriodKey (const FlightNumber_T &, const PeriodStruct &)
- FlightPeriodKey (const FlightPeriodKey &)
- ~FlightPeriodKey ()
- const FlightNumber_T & getFlightNumber () const
- const PeriodStruct & getPeriod () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const

8.81.1 Detailed Description

Key of flight-period.

Definition at line 13 of file FlightPeriodKey.hpp.

8.81.2 Constructor & Destructor Documentation

8.81.2.1 stdair::FlightPeriodKey::FlightPeriodKey (const FlightNumber_T &, const PeriodStruct &)

Constructors.

Definition at line 10 of file FlightPeriodKey.cpp.

8.81.2.2 stdair::FlightPeriodKey::FlightPeriodKey (const FlightPeriodKey &)

Definition at line 16 of file FlightPeriodKey.cpp.

8.81.2.3 stdair::FlightPeriodKey::~FlightPeriodKey ()

Destructor.

Definition at line 21 of file FlightPeriodKey.cpp.

8.81.3 Member Function Documentation

8.81.3.1 const FlightNumber_T& stdair::FlightPeriodKey::getFlightNumber () const [inline]

Get the flight number.

Definition at line 28 of file FlightPeriodKey.hpp.

Referenced by stdair::FlightPeriod::getFlightNumber().

8.81.3.2 const PeriodStruct& stdair::FlightPeriodKey::getPeriod () const [inline]

Get the active days-of-week.

Definition at line 33 of file FlightPeriodKey.hpp.

Referenced by stdair::FlightPeriod::getPeriod().

8.81.3.3 void stdair::FlightPeriodKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 25 of file FlightPeriodKey.cpp.

References toString().

8.81.3.4 void stdair::FlightPeriodKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 30 of file FlightPeriodKey.cpp.

8.81.3.5 const std::string stdair::FlightPeriodKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-period.

Reimplemented from stdair::KeyAbstract.

Definition at line 34 of file FlightPeriodKey.cpp.

References stdair::PeriodStruct::describeShort().

Referenced by stdair::FlightPeriod::describeKey(), and toStream().

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

- stdair/bom/FlightPeriodKey.hpp
- stdair/bom/FlightPeriodKey.cpp

8.82 FloatingPoint< RawType > Class Template Reference

`#include <stdair/basic/float_utils_google.hpp>`

Public Types

- typedef TypeWithSize< sizeof(RawType)>::UInt Bits

Public Member Functions

- FloatingPoint (const RawType &x)
- const Bits & bits () const
- Bits exponent_bits () const
- Bits fraction_bits () const
- Bits sign_bit () const
- bool is_nan () const
- bool AlmostEquals (const FloatingPoint &rhs) const

Static Public Member Functions

- static RawType ReinterpretBits (const Bits bits)
- static RawType Infinity ()

Static Public Attributes

- static const size_t kBitCount = 8∗sizeof(RawType)
- static const size_t kFractionBitCount
- static const size_t kExponentBitCount = kBitCount - 1 - kFractionBitCount

- static const Bits kSignBitMask = static_cast<Bits>(1) << (kBitCount - 1)
- static const Bits kFractionBitMask
- static const Bits kExponentBitMask = ~(kSignBitMask | kFractionBitMask)
- static const size_t kMaxUlps = 4

8.82.1 Detailed Description

template<typename RawType> class FloatingPoint< RawType >

Definition at line 117 of file float_utils_google.hpp.

8.82.2 Member Typedef Documentation

8.82.2.1 template<typename RawType> typedef TypeWithSize<sizeof(RawType)>::UInt FloatingPoint< RawType >::Bits

Definition at line 121 of file float_utils_google.hpp.

8.82.3 Constructor & Destructor Documentation

8.82.3.1 template<typename RawType> FloatingPoint< RawType >::FloatingPoint (const RawType & *x*) [inline, explicit]

Definition at line 165 of file float_utils_google.hpp.

8.82.4 Member Function Documentation

8.82.4.1 template<typename RawType> static RawType FloatingPoint< RawType >::ReinterpretBits (const Bits *bits*) [inline, static]

Definition at line 172 of file float_utils_google.hpp.

References FloatingPoint< RawType >::u_.

Referenced by FloatingPoint< RawType >::Infinity().

8.82.4.2 template<typename RawType> static RawType FloatingPoint< RawType >::Infinity () [inline, static]

Definition at line 179 of file float_utils_google.hpp.

References FloatingPoint< RawType >::kExponentBitMask, and FloatingPoint< RawType >::ReinterpretBits().

8.82.4.3 template<typename RawType> const Bits& FloatingPoint< RawType >::bits () const [inline]

Definition at line 186 of file float_utils_google.hpp.

8.82.4.4 template<typename RawType> Bits FloatingPoint< RawType >::exponent_bits () const [inline]

Definition at line 189 of file float_utils_google.hpp.

References FloatingPoint< RawType >::kExponentBitMask.

Referenced by FloatingPoint< RawType >::is_nan().

8.82.4.5 template<typename RawType> Bits FloatingPoint< RawType >::fraction_bits () const [inline]

Definition at line 192 of file float_utils_google.hpp.

References FloatingPoint< RawType >::kFractionBitMask.

Referenced by FloatingPoint< RawType >::is_nan().

8.82.4.6 template<typename RawType> Bits FloatingPoint< RawType >::sign_bit () const [inline]

Definition at line 195 of file float_utils_google.hpp.

References FloatingPoint< RawType >::kSignBitMask.

8.82.4.7 template<typename RawType> bool FloatingPoint< RawType >::is_nan () const [inline]

Definition at line 198 of file float_utils_google.hpp.

References FloatingPoint< RawType >::exponent_bits(), FloatingPoint< RawType >::fraction_bits(), and FloatingPoint< RawType >::kExponentBitMask.

Referenced by FloatingPoint< RawType >::AlmostEquals().

8.82.4.8 template<typename RawType> bool FloatingPoint< RawType >::AlmostEquals (const FloatingPoint< RawType > & *rhs*) const [inline]

Definition at line 210 of file float_utils_google.hpp.

References FloatingPoint< RawType >::is_nan(), FloatingPoint< RawType >::kMaxUlps, and FloatingPoint< RawType >::u_.

8.82.5 Member Data Documentation

8.82.5.1 template<typename RawType> const size_t FloatingPoint< RawType >::kBitCount = 8∗sizeof(RawType) [static]

Definition at line 126 of file float_utils_google.hpp.

8.82.5.2 template<typename RawType> const size_t FloatingPoint< RawType >::kFractionBitCount [static]

Initial value:

```
std::numeric_limits<RawType>::digits - 1
```

Definition at line 129 of file float_utils_google.hpp.

8.82.5.3 template<typename RawType> const size_t FloatingPoint< RawType >::kExponentBitCount = kBitCount - 1 - kFractionBitCount [static]

Definition at line 133 of file float_utils_google.hpp.

8.82.5.4 template<typename RawType> const Bits FloatingPoint< RawType >::kSignBitMask = static_cast<Bits>(1) << (kBitCount - 1) [static]

Definition at line 136 of file float_utils_google.hpp.

Referenced by FloatingPoint< RawType >::sign_bit().

8.82.5.5 template<typename RawType> const Bits FloatingPoint< RawType >::kFractionBitMask [static]

Initial value:

```
~static_cast<Bits>(0) >> (kExponentBitCount + 1)
```

Definition at line 139 of file float_utils_google.hpp.

Referenced by FloatingPoint< RawType >::fraction_bits().

8.82.5.6 template<typename RawType> const Bits FloatingPoint< RawType >::kExponentBitMask = ~(kSignBitMask | kFractionBitMask) [static]

Definition at line 143 of file float_utils_google.hpp.

Referenced by FloatingPoint< RawType >::exponent_bits(), FloatingPoint< RawType >::Infinity(), and FloatingPoint< RawType >::is_nan().

8.82.5.7 template<typename RawType> const size_t FloatingPoint< RawType >::kMaxUlps = 4 [static]

Definition at line 157 of file float_utils_google.hpp.

Referenced by FloatingPoint< RawType >::AlmostEquals().

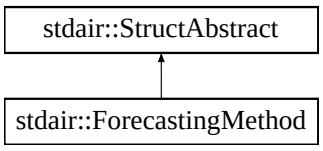
The documentation for this class was generated from the following file:

- stdair/basic/float_utils_google.hpp

8.83 stdair::ForecastingMethod Struct Reference

`#include <stdair/basic/ForecastingMethod.hpp>`

Inheritance diagram for stdair::ForecastingMethod::



Public Types

- Q_FORECASTING = 0
- HYBRID_FORECASTING
- OLD_QFF
- NEW_QFF
- BASED_FORECASTING
- LAST_VALUE
- enum EN_ForecastingMethod {
  Q_FORECASTING = 0, HYBRID_FORECASTING, OLD_QFF, NEW_QFF,
  BASED_FORECASTING, LAST_VALUE }

Public Member Functions

- EN_ForecastingMethod getMethod () const
- std::string getMethodAsString () const
- const std::string describe () const
- bool operator== (const EN_ForecastingMethod &) const
- ForecastingMethod (const EN_ForecastingMethod &)
- ForecastingMethod (const char iMethod)
- ForecastingMethod (const ForecastingMethod &)
- void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

Static Public Member Functions

- static const std::string & getLabel (const EN_ForecastingMethod &)
- static char getMethodLabel (const EN_ForecastingMethod &)
- static std::string getMethodLabelAsString (const EN_ForecastingMethod &)
- static std::string describeLabels ()

8.83.1 Detailed Description

Enumeration of forecasting methods.

Definition at line 15 of file ForecastingMethod.hpp.

8.83.2 Member Enumeration Documentation

8.83.2.1 enum stdair::ForecastingMethod::EN_ForecastingMethod

Enumerator:

Q_FORECASTING
HYBRID_FORECASTING
OLD_QFF
NEW_QFF
BASED_FORECASTING
LAST_VALUE

Definition at line 17 of file ForecastingMethod.hpp.

8.83.3 Constructor & Destructor Documentation

8.83.3.1 stdair::ForecastingMethod::ForecastingMethod (const EN_ForecastingMethod &)

Constructor.

Definition at line 37 of file ForecastingMethod.cpp.

8.83.3.2 stdair::ForecastingMethod::ForecastingMethod (const char *iMethod*)

Constructor.

Definition at line 42 of file ForecastingMethod.cpp.

References BASED_FORECASTING, describeLabels(), HYBRID_FORECASTING, LAST_VALUE, NEW_QFF, OLD_QFF, and Q_FORECASTING.

8.83.3.3 stdair::ForecastingMethod::ForecastingMethod (const ForecastingMethod &)

Default copy constructor.

Definition at line 31 of file ForecastingMethod.cpp.

8.83.4 Member Function Documentation

8.83.4.1 const std::string & stdair::ForecastingMethod::getLabel (const EN_ForecastingMethod &) [static]

Get the label as a string (e.g., "Q Forecasting", "Hybrid Forecasting", "Old QFF" or "New QFF").

Definition at line 63 of file ForecastingMethod.cpp.

8.83.4.2 char stdair::ForecastingMethod::getMethodLabel (const EN_ForecastingMethod &) [static]

Get the label as a single char (e.g., 'Q', 'H', 'O', 'N' or 'B').

Definition at line 68 of file ForecastingMethod.cpp.

8.83.4.3 std::string stdair::ForecastingMethod::getMethodLabelAsString (const EN_ForecastingMethod &) [static]

Get the label as a string of a single char (e.g., "Q", "H", "O", "N" or "B").

Definition at line 74 of file ForecastingMethod.cpp.

8.83.4.4 std::string stdair::ForecastingMethod::describeLabels () [static]

List the labels.

Definition at line 81 of file ForecastingMethod.cpp.

References LAST_VALUE.

Referenced by ForecastingMethod().

8.83.4.5 ForecastingMethod::EN_ForecastingMethod stdair::ForecastingMethod::getMethod () const

Get the enumerated value.

Definition at line 93 of file ForecastingMethod.cpp.

Referenced by stdair::AirlineFeature::getForecastingMethod().

8.83.4.6 std::string stdair::ForecastingMethod::getMethodAsString () const

Get the enumerated value as a short string (e.g., "Q", "H", "O", "N" or "B").

Definition at line 98 of file ForecastingMethod.cpp.

8.83.4.7 const std::string stdair::ForecastingMethod::describe () const [virtual]

Give a description of the structure (e.g., "Q Forecasting", "Hybrid Forecasting, "Old QFF", "New QFF" or "Based Forecasting").

Implements stdair::StructAbstract.

Definition at line 105 of file ForecastingMethod.cpp.

8.83.4.8 bool stdair::ForecastingMethod::operator== (const EN_ForecastingMethod &) const

Comparison operator.

Definition at line 113 of file ForecastingMethod.cpp.

8.83.4.9 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const [inline, inherited]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References stdair::StructAbstract::describe().

8.83.4.10 virtual void stdair::StructAbstract::fromStream (std::istream & *ioIn*) [inline, virtual, inherited]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented in stdair::ProgressStatusSet, stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct,

stdair::EventStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 38 of file StructAbstract.hpp.

Referenced by operator>>().

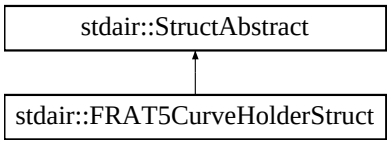
The documentation for this struct was generated from the following files:

- stdair/basic/ForecastingMethod.hpp
- stdair/basic/ForecastingMethod.cpp

8.84 stdair::FRAT5CurveHolderStruct Struct Reference

`#include <stdair/bom/FRAT5CurveHolderStruct.hpp>`

Inheritance diagram for stdair::FRAT5CurveHolderStruct::



Public Member Functions

- const FRAT5Curve_T & getFRAT5Curve (const std::string &) const
- void addCurve (const std::string &, const FRAT5Curve_T &)
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string describe () const
- FRAT5CurveHolderStruct ()
- FRAT5CurveHolderStruct (const FRAT5CurveHolderStruct &)
- ~FRAT5CurveHolderStruct ()

8.84.1 Detailed Description

Structure holding the elements of a snapshot .

Definition at line 19 of file FRAT5CurveHolderStruct.hpp.

8.84.2 Constructor & Destructor Documentation

8.84.2.1 stdair::FRAT5CurveHolderStruct::FRAT5CurveHolderStruct ()

Constructor.

Definition at line 14 of file FRAT5CurveHolderStruct.cpp.

8.84.2.2 stdair::FRAT5CurveHolderStruct::FRAT5CurveHolderStruct (const FRAT5CurveHolderStruct &)

Copy constructor.

Definition at line 19 of file FRAT5CurveHolderStruct.cpp.

8.84.2.3 stdair::FRAT5CurveHolderStruct::~FRAT5CurveHolderStruct ()

Destructor.

Definition at line 24 of file FRAT5CurveHolderStruct.cpp.

8.84.3 Member Function Documentation

8.84.3.1 const FRAT5Curve_T & stdair::FRAT5CurveHolderStruct::getFRAT5Curve (const std::string &) const

Get the FRAT5 curve corresponding to the given key.

Definition at line 29 of file FRAT5CurveHolderStruct.cpp.

References STDAIR_LOG_DEBUG.

Referenced by stdair::BomRoot::getFRAT5Curve().

8.84.3.2 void stdair::FRAT5CurveHolderStruct::addCurve (const std::string &, const FRAT5Curve_T &)

Add a new curve to the holder.

Definition at line 42 of file FRAT5CurveHolderStruct.cpp.

References STDAIR_LOG_DEBUG.

Referenced by stdair::BomRoot::addFRAT5Curve().

8.84.3.3 void stdair::FRAT5CurveHolderStruct::toStream (std::ostream & *ioOut*) const

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::StructAbstract.

Definition at line 53 of file FRAT5CurveHolderStruct.cpp.

References describe().

8.84.3.4 void stdair::FRAT5CurveHolderStruct::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::StructAbstract.

Definition at line 58 of file FRAT5CurveHolderStruct.cpp.

8.84.3.5 const std::string stdair::FRAT5CurveHolderStruct::describe () const [virtual]

Display of the structure.

Implements stdair::StructAbstract.

Definition at line 62 of file FRAT5CurveHolderStruct.cpp.

Referenced by toStream().

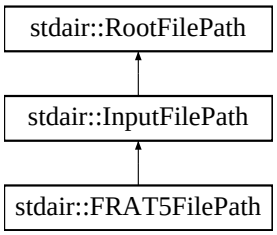
The documentation for this struct was generated from the following files:

- stdair/bom/FRAT5CurveHolderStruct.hpp
- stdair/bom/FRAT5CurveHolderStruct.cpp

8.85 stdair::FRAT5FilePath Class Reference

`#include <stdair/stdair_file.hpp>`

Inheritance diagram for stdair::FRAT5FilePath::



Public Member Functions

- FRAT5FilePath (const Filename_T &iFilename)
- const char * name () const

Protected Attributes

- const Filename_T _filename

8.85.1 Detailed Description

FRAT5 input file.

Definition at line 88 of file stdair_file.hpp.

8.85.2 Constructor & Destructor Documentation

8.85.2.1 stdair::FRAT5FilePath::FRAT5FilePath (const Filename_T & *iFilename*) [inline, explicit]

Constructor.

Definition at line 93 of file stdair_file.hpp.

8.85.3 Member Function Documentation

8.85.3.1 const char* stdair::RootFilePath::name () const [inline, inherited]

Give the details of the exception.

Definition at line 42 of file stdair_file.hpp.

References stdair::RootFilePath::_filename.

Referenced by stdair::BomINIImport::importINIConfig().

8.85.4 Member Data Documentation

8.85.4.1 const Filename_T stdair::RootFilePath::_filename [protected, inherited]

Name of the file.

Definition at line 50 of file stdair_file.hpp.

Referenced by stdair::RootFilePath::name().

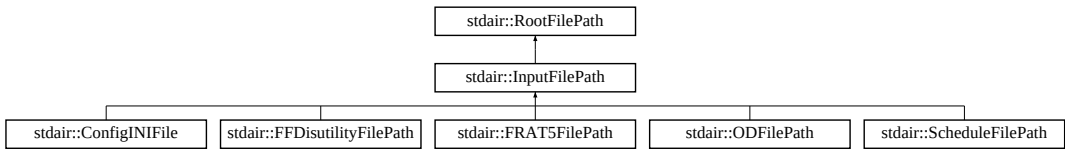
The documentation for this class was generated from the following file:

- stdair/stdair_file.hpp

8.86 stdair::InputFilePath Class Reference

`#include <stdair/stdair_file.hpp>`

Inheritance diagram for stdair::InputFilePath::



Public Member Functions

- InputFilePath (const Filename_T &iFilename)
- const char * name () const

Protected Attributes

- const Filename_T _filename

8.86.1 Detailed Description

Input File.

Definition at line 54 of file stdair_file.hpp.

8.86.2 Constructor & Destructor Documentation

8.86.2.1 stdair::InputFilePath::InputFilePath (const Filename_T & *iFilename*) [inline]

Constructor.

Definition at line 57 of file stdair_file.hpp.

8.86.3 Member Function Documentation

8.86.3.1 const char* stdair::RootFilePath::name () const [inline, inherited]

Give the details of the exception.

Definition at line 42 of file stdair_file.hpp.

References stdair::RootFilePath::_filename.

Referenced by stdair::BomINIImport::importINIConfig().

8.86.4 Member Data Documentation

8.86.4.1 const Filename_T stdair::RootFilePath::_filename [protected, inherited]

Name of the file.

Definition at line 50 of file stdair_file.hpp.

Referenced by stdair::RootFilePath::name().

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

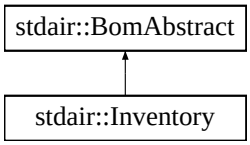
- stdair/stdair_file.hpp

8.87 stdair::Inventory Class Reference

Class representing the actual attributes for an airline inventory.

`#include <stdair/bom/Inventory.hpp>`

Inheritance diagram for stdair::Inventory::



Public Types

- typedef InventoryKey Key_T

Public Member Functions

- const Key_T & getKey () const
- const AirlineCode_T & getAirlineCode () const
- ForecastingMethod::EN_ForecastingMethod getForecastingMethod () const
- UnconstrainingMethod::EN_UnconstrainingMethod getUnconstrainingMethod () const
- PreOptimisationMethod::EN_PreOptimisationMethod getPreOptimisationMethod () const
- OptimisationMethod::EN_OptimisationMethod getOptimisationMethod () const
- PartnershipTechnique::EN_PartnershipTechnique getPartnershipTechnique () const
- BomAbstract *const getParent () const
- const HolderMap_T & getHolderMap () const
- FlightDate * getFlightDate (const std::string &iFlightDateKeyStr) const
- FlightDate * getFlightDate (const FlightDateKey &) const
- AirlineFeature * getAirlineFeature () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- template<class Archive> void serialize (Archive &ar, const unsigned int iFileVersion)

Protected Member Functions

- Inventory (const Key_T &)
- ~Inventory ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent
- AirlineFeature * _airlineFeature
- HolderMap_T _holderMap

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager
- class boost::serialization::access

8.87.1 Detailed Description

Class representing the actual attributes for an airline inventory.

Definition at line 34 of file Inventory.hpp.

8.87.2 Member Typedef Documentation

8.87.2.1 typedef InventoryKey stdair::Inventory::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 45 of file Inventory.hpp.

8.87.3 Constructor & Destructor Documentation

8.87.3.1 stdair::Inventory::Inventory (const Key_T &) [protected]

Constructor.

Definition at line 31 of file Inventory.cpp.

8.87.3.2 stdair::Inventory::~Inventory () [protected]

Destructor.

Definition at line 38 of file Inventory.cpp.

8.87.4 Member Function Documentation

8.87.4.1 const Key_T& stdair::Inventory::getKey () const [inline]

Get the inventory key (airline code).

Definition at line 51 of file Inventory.hpp.

References _key.

8.87.4.2 const AirlineCode_T& stdair::Inventory::getAirlineCode () const [inline]

Get the airline code (inventory/primary key).

Definition at line 56 of file Inventory.hpp.

References _key, and stdair::InventoryKey::getAirlineCode().

Referenced by stdair::OnDDate::getAirlineCode(), stdair::FlightDate::getAirlineCode(), and stdair::BomRetriever::retrieveSegmentDateFromLongKey().

8.87.4.3 ForecastingMethod::EN_ForecastingMethod stdair::Inventory::getForecastingMethod () const

Get the forecasting method.

Definition at line 64 of file Inventory.cpp.

References _airlineFeature, and stdair::AirlineFeature::getForecastingMethod().

8.87.4.4 UnconstrainingMethod::EN_UnconstrainingMethod stdair::Inventory::getUnconstrainingMethod () const

Get the unconstraining method.

Definition at line 71 of file Inventory.cpp.

References _airlineFeature, and stdair::AirlineFeature::getUnconstrainingMethod().

8.87.4.5 PreOptimisationMethod::EN_PreOptimisationMethod stdair::Inventory::getPreOptimisationMethod () const

Get the pre-optimisation method.

Definition at line 78 of file Inventory.cpp.

References _airlineFeature, and stdair::AirlineFeature::getPreOptimisationMethod().

8.87.4.6 OptimisationMethod::EN_OptimisationMethod stdair::Inventory::getOptimisationMethod () const

Get the optimisation method.

Definition at line 85 of file Inventory.cpp.

References _airlineFeature, and stdair::AirlineFeature::getOptimisationMethod().

8.87.4.7 PartnershipTechnique::EN_PartnershipTechnique stdair::Inventory::getPartnershipTechnique () const

Get the partnership technique.

Definition at line 92 of file Inventory.cpp.

References _airlineFeature, and stdair::AirlineFeature::getPartnershipTechnique().

8.87.4.8 BomAbstract* const stdair::Inventory::getParent () const [inline]

Get the parent object.

Definition at line 76 of file Inventory.hpp.

References _parent.

8.87.4.9 const HolderMap_T& stdair::Inventory::getHolderMap () const [inline]

Get the map of children.

Definition at line 81 of file Inventory.hpp.

References _holderMap.

8.87.4.10 FlightDate * stdair::Inventory::getFlightDate (const std::string & *iFlightDateKeyStr*) const

Get a pointer on the FlightDate object corresponding to the given key.

Note:

The FlightDate object can be inherited from, if needed. In that case, a dynamic_cast<> may be needed.

Parameters:

const std::string& The flight-date key.

Returns:

FlightDate* Found FlightDate object. NULL if not found.

Definition at line 50 of file Inventory.cpp.

Referenced by getFlightDate(), stdair::BomRetriever::retrieveFlightDateFromKey(), and stdair::BomRetriever::retrieveFlightDateFromLongKey().

8.87.4.11 FlightDate * stdair::Inventory::getFlightDate (const FlightDateKey &) const

Get a pointer on the FlightDate object corresponding to the given key.

Note:

The FlightDate object can be inherited from, if needed. In that case, a dynamic_cast<> may be needed.

Parameters:

const FlightDateKey& The flight-date key

Returns:

FlightDate* Found FlightDate object. NULL if not found.

Definition at line 58 of file Inventory.cpp.

References getFlightDate(), and stdair::FlightDateKey::toString().

8.87.4.12 AirlineFeature* stdair::Inventory::getAirlineFeature () const [inline]

Get the airline feature.

Definition at line 112 of file Inventory.hpp.

References _airlineFeature.

8.87.4.13 void stdair::Inventory::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 132 of file Inventory.hpp.

References toString().

8.87.4.14 void stdair::Inventory::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 141 of file Inventory.hpp.

8.87.4.15 std::string stdair::Inventory::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 42 of file Inventory.cpp.

References describeKey().

Referenced by toStream().

8.87.4.16 const std::string stdair::Inventory::describeKey () const [inline]

Get a string describing the key.

Definition at line 152 of file Inventory.hpp.

References _key, and stdair::InventoryKey::toString().

Referenced by stdair::BomRetriever::retrieveFullKeyFromSegmentDate(), and toString().

8.87.4.17 template<class Archive> void stdair::Inventory::serialize (Archive & *ar*, const unsigned int *iFileVersion*)

Serialisation.

Definition at line 160 of file CmdBomSerialiser.cpp.

References _key.

8.87.5 Friends And Related Function Documentation

8.87.5.1 friend class FacBom [friend]

Definition at line 35 of file Inventory.hpp.

8.87.5.2 friend class FacCloneBom [friend]

Definition at line 36 of file Inventory.hpp.

8.87.5.3 friend class FacBomManager [friend]

Definition at line 37 of file Inventory.hpp.

8.87.5.4 friend class boost::serialization::access [friend]

Definition at line 38 of file Inventory.hpp.

8.87.6 Member Data Documentation

8.87.6.1 Key_T stdair::Inventory::_key [protected]

Primary key (airline code).

Definition at line 204 of file Inventory.hpp.

Referenced by describeKey(), getAirlineCode(), getKey(), and serialize().

8.87.6.2 BomAbstract* stdair::Inventory::_parent [protected]

Pointer on the parent class (BomRoot).

Definition at line 209 of file Inventory.hpp.

Referenced by getParent().

8.87.6.3 AirlineFeature* stdair::Inventory::_airlineFeature [protected]

Features specific to the airline.

Definition at line 214 of file Inventory.hpp.

Referenced by getAirlineFeature(), getForecastingMethod(), getOptimisationMethod(), getPartnershipTechnique(), getPreOptimisationMethod(), and getUnconstrainingMethod().

8.87.6.4 HolderMap_T stdair::Inventory::_holderMap [protected]

Map holding the children (FlightDate objects).

Definition at line 219 of file Inventory.hpp.

Referenced by getHolderMap().

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

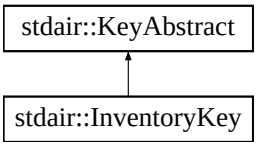
- stdair/bom/Inventory.hpp
- stdair/bom/Inventory.cpp
- stdair/command/CmdBomSerialiser.cpp

8.88 stdair::InventoryKey Struct Reference

Key of a given inventory, made of the airline code.

`#include <stdair/bom/InventoryKey.hpp>`

Inheritance diagram for stdair::InventoryKey::



Public Member Functions

- InventoryKey (const AirlineCode_T &iAirlineCode)
- InventoryKey (const InventoryKey &)
- ~InventoryKey ()
- const AirlineCode_T & getAirlineCode () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const
- template<class Archive> void serialize (Archive &ar, const unsigned int iFileVersion)

Friends

- class boost::serialization::access

8.88.1 Detailed Description

Key of a given inventory, made of the airline code.

Definition at line 26 of file InventoryKey.hpp.

8.88.2 Constructor & Destructor Documentation

8.88.2.1 stdair::InventoryKey::InventoryKey (const AirlineCode_T & *iAirlineCode*)

Constructor.

Definition at line 23 of file InventoryKey.cpp.

8.88.2.2 stdair::InventoryKey::InventoryKey (const InventoryKey &)

Copy constructor.

Definition at line 28 of file InventoryKey.cpp.

8.88.2.3 stdair::InventoryKey::~InventoryKey ()

Destructor.

Definition at line 33 of file InventoryKey.cpp.

8.88.3 Member Function Documentation

8.88.3.1 const AirlineCode_T& stdair::InventoryKey::getAirlineCode () const [inline]

Get the airline code.

Definition at line 58 of file InventoryKey.hpp.

Referenced by stdair::Inventory::getAirlineCode(), stdair::BomRetriever::retrieveInventoryFromLongKey(), and stdair::BomRetriever::retrievePartnerSegmentDateFromLongKey().

8.88.3.2 void stdair::InventoryKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 37 of file InventoryKey.cpp.

References toString().

8.88.3.3 void stdair::InventoryKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 42 of file InventoryKey.cpp.

8.88.3.4 const std::string stdair::InventoryKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from stdair::KeyAbstract.

Definition at line 46 of file InventoryKey.cpp.

Referenced by stdair::Inventory::describeKey(), stdair::BomRoot::getInventory(), and toStream().

8.88.3.5 template<class Archive> void stdair::InventoryKey::serialize (Archive & *ar*, const unsigned int *iFileVersion*)

Serialisation.

Definition at line 68 of file InventoryKey.cpp.

8.88.4 Friends And Related Function Documentation

8.88.4.1 friend class boost::serialization::access [friend]

Definition at line 27 of file InventoryKey.hpp.

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

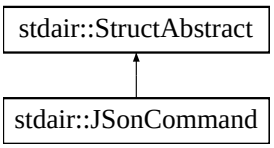
- stdair/bom/InventoryKey.hpp
- stdair/bom/InventoryKey.cpp

8.89 stdair::JSonCommand Struct Reference

Enumeration of json commands.

`#include <stdair/basic/JSonCommand.hpp>`

Inheritance diagram for stdair::JSonCommand::



Public Types

- LIST = 0
- FLIGHT_DATE
- EVENT_LIST
- BREAK_POINT
- RUN
- RESET
- STATUS
- CONFIG
- LAST_VALUE
- enum EN_JSonCommand {
  LIST = 0, FLIGHT_DATE, EVENT_LIST, BREAK_POINT,
  RUN, RESET, STATUS, CONFIG,
  LAST_VALUE }

Public Member Functions

- EN_JSonCommand getCommand () const
- const std::string describe () const
- bool operator== (const EN_JSonCommand &) const
- JSonCommand (const EN_JSonCommand &)
- JSonCommand (const std::string &)
- JSonCommand (const JSonCommand &)
- void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

Static Public Member Functions

- static EN_JSonCommand getCommand (const std::string &iCommandStr)
- static std::string getLabel (const EN_JSonCommand &)
- static std::string describeLabels ()

8.89.1 Detailed Description

Enumeration of json commands.

Definition at line 17 of file JSonCommand.hpp.

8.89.2 Member Enumeration Documentation

8.89.2.1 enum stdair::JSonCommand::EN_JSonCommand

Enumerator:

LIST

FLIGHT_DATE

EVENT_LIST

BREAK_POINT

RUN
RESET
STATUS
CONFIG
LAST_VALUE

Definition at line 19 of file JSonCommand.hpp.

8.89.3 Constructor & Destructor Documentation

8.89.3.1 stdair::JSonCommand::JSonCommand (const EN_JSonCommand &)

Main Constructor.

8.89.3.2 stdair::JSonCommand::JSonCommand (const std::string &)

Alternative constructor.

Definition at line 71 of file JSonCommand.cpp.

References getCommand().

8.89.3.3 stdair::JSonCommand::JSonCommand (const JSonCommand &)

Default copy constructor.

Definition at line 25 of file JSonCommand.cpp.

8.89.4 Member Function Documentation

8.89.4.1 JSonCommand::EN_JSonCommand stdair::JSonCommand::getCommand (const std::string & *iCommandStr*) [static]

Get the command value from parsing a single char (e.g., "list", "flight_date", "event_list", "break_point", "run", "reset, "status" or "config").

Definition at line 31 of file JSonCommand.cpp.

References BREAK_POINT, CONFIG, describeLabels(), EVENT_LIST, FLIGHT_DATE, LAST_-VALUE, LIST, RESET, RUN, and STATUS.

Referenced by stdair::BomJSONImport::jsonImportCommand().

8.89.4.2 std::string stdair::JSonCommand::getLabel (const EN_JSonCommand &) [static]

Get a label of a command

Definition at line 66 of file JSonCommand.cpp.

8.89.4.3 std::string stdair::JSonCommand::describeLabels () [static]

List the labels.

Definition at line 77 of file JSonCommand.cpp.

References LAST_VALUE.

Referenced by getCommand().

8.89.4.4 JSonCommand::EN_JSonCommand stdair::JSonCommand::getCommand () const

Get the enumerated value.

Definition at line 89 of file JSonCommand.cpp.

Referenced by JSonCommand().

8.89.4.5 const std::string stdair::JSonCommand::describe () const [virtual]

Give a description of the structure (e.g., "list", "flight_date", "event_list", "break_point" "run", "reset, "status" or "config").

Implements stdair::StructAbstract.

Definition at line 94 of file JSonCommand.cpp.

8.89.4.6 bool stdair::JSonCommand::operator== (const EN_JSonCommand &) const

Comparison operator.

Definition at line 102 of file JSonCommand.cpp.

8.89.4.7 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const [inline, inherited]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References stdair::StructAbstract::describe().

8.89.4.8 virtual void stdair::StructAbstract::fromStream (std::istream & *ioIn*) [inline, virtual, inherited]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented in stdair::ProgressStatusSet, stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct,

stdair::EventStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 38 of file StructAbstract.hpp.

Referenced by operator>>().

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

- stdair/basic/JSonCommand.hpp
- stdair/basic/JSonCommand.cpp

8.90 stdair::JSONString Class Reference

JSON-formatted string.

`#include <stdair/stdair_json.hpp>`

Public Member Functions

- JSONString (const std::string &iJsonString)
- JSONString ()
- virtual ~JSONString ()
- const std::string & getString () const

Protected Attributes

- std::string _jsonString

8.90.1 Detailed Description

JSON-formatted string.

Definition at line 16 of file stdair_json.hpp.

8.90.2 Constructor & Destructor Documentation

8.90.2.1 stdair::JSONString::JSONString (const std::string & *iJsonString*) [inline, explicit]

Main Constructor.

Definition at line 21 of file stdair_json.hpp.

8.90.2.2 stdair::JSONString::JSONString () [inline, explicit]

Default constructor.

Definition at line 26 of file stdair_json.hpp.

8.90.2.3 virtual stdair::JSONString::~JSONString () [inline, virtual]

Destructor.

Definition at line 31 of file stdair_json.hpp.

8.90.3 Member Function Documentation

8.90.3.1 const std::string& stdair::JSONString::getString () const [inline]

Get the string value.

Definition at line 36 of file stdair_json.hpp.

References _jsonString.

Referenced by stdair::BomJSONImport::jsonImportBreakPoints(), stdair::BomJSONImport::jsonImportCommand(), stdair::BomJSONImport::jsonImportConfig(), stdair::BomJSONImport::jsonImportEventType(), stdair::BomJSONImport::jsonImportFlightDate(), stdair::BomJSONImport::jsonImportFlightNumber(), and stdair::BomJSONImport::jsonImportInventoryKey().

8.90.4 Member Data Documentation

8.90.4.1 std::string stdair::JSONString::_jsonString [protected]

Definition at line 44 of file stdair_json.hpp.

Referenced by getString().

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

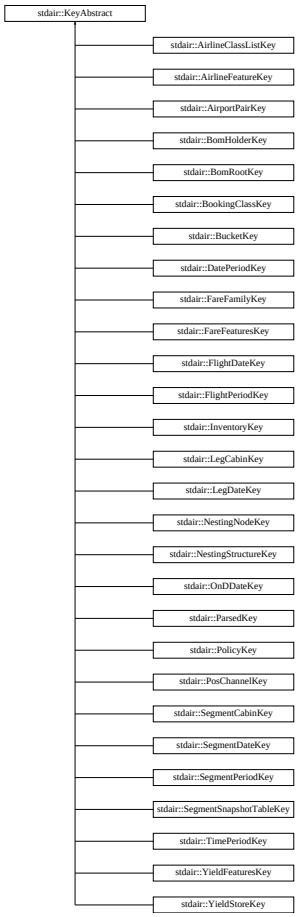
- stdair/stdair_json.hpp

8.91 stdair::KeyAbstract Struct Reference

Base class for the keys of Business Object Model (BOM) layer.

`#include <stdair/bom/KeyAbstract.hpp>`

Inheritance diagram for stdair::KeyAbstract::



Public Member Functions

- virtual void toStream (std::ostream &ioOut) const

  Dump a Business Object Key into an output stream.

- virtual void fromStream (std::istream &ioIn)

  Read a Business Object Key from an input stream.

- virtual const std::string toString () const

  Get the serialised version of the Business Object Key.

- virtual ~KeyAbstract ()

  Default destructor.

8.91.1 Detailed Description

Base class for the keys of Business Object Model (BOM) layer.

Note that that key allows to differentiate two objects at the same level only. For instance, the segment-date key allows to differentiate two segment-dates under a given flight-date, but does not allow to differentiate two segment-dates in general.

Definition at line 27 of file KeyAbstract.hpp.

8.91.2 Constructor & Destructor Documentation

8.91.2.1 virtual stdair::KeyAbstract::~KeyAbstract () [inline, virtual]

Default destructor.

Definition at line 61 of file KeyAbstract.hpp.

8.91.3 Member Function Documentation

8.91.3.1 virtual void stdair::KeyAbstract::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object Key into an output stream.

Parameters:

↔ ***ostream&*** the output stream.

Reimplemented in stdair::AirlineClassListKey, stdair::AirlineFeatureKey, stdair::AirportPairKey, stdair::BomHolderKey, stdair::BomRootKey, stdair::BookingClassKey, stdair::BucketKey, stdair::DatePeriodKey, stdair::FareFamilyKey, stdair::FareFeaturesKey, stdair::FlightDateKey, stdair::FlightPeriodKey, stdair::InventoryKey, stdair::LegCabinKey, stdair::LegDateKey, stdair::NestingNodeKey, stdair::NestingStructureKey, stdair::OnDDateKey, stdair::ParsedKey, stdair::PolicyKey, stdair::PosChannelKey, stdair::SegmentCabinKey, stdair::SegmentDateKey, stdair::SegmentPeriodKey, stdair::SegmentSnapshotTableKey, stdair::TimePeriodKey, stdair::YieldFeaturesKey, and stdair::YieldStoreKey.

Definition at line 36 of file KeyAbstract.hpp.

8.91.3.2 virtual void stdair::KeyAbstract::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object Key from an input stream.

Parameters:

↔ ***istream&*** the input stream.

Reimplemented in stdair::AirlineClassListKey, stdair::AirlineFeatureKey, stdair::AirportPairKey, stdair::BomHolderKey, stdair::BomRootKey, stdair::BookingClassKey, stdair::BucketKey, stdair::DatePeriodKey, stdair::FareFamilyKey, stdair::FareFeaturesKey, stdair::FlightDateKey, stdair::FlightPeriodKey, stdair::InventoryKey, stdair::LegCabinKey, stdair::LegDateKey, stdair::NestingNodeKey, stdair::NestingStructureKey, stdair::OnDDateKey, stdair::ParsedKey, stdair::PolicyKey, stdair::PosChannelKey, stdair::SegmentCabinKey, stdair::SegmentDateKey, stdair::SegmentPeriodKey, stdair::SegmentSnapshotTableKey, stdair::TimePeriodKey, stdair::YieldFeaturesKey, and stdair::YieldStoreKey.

Definition at line 43 of file KeyAbstract.hpp.

Referenced by operator>>().

8.91.3.3 virtual const std::string stdair::KeyAbstract::toString () const [inline, virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Parameters:

→ ***const*** std::string The serialised version of the Business Object Key.

Reimplemented in stdair::AirlineClassListKey, stdair::AirlineFeatureKey, stdair::AirportPairKey, stdair::BomHolderKey, stdair::BomRootKey, stdair::BookingClassKey, stdair::BucketKey, stdair::DatePeriodKey, stdair::FareFamilyKey, stdair::FareFeaturesKey, stdair::FlightDateKey, stdair::FlightPeriodKey, stdair::InventoryKey, stdair::LegCabinKey, stdair::LegDateKey, stdair::NestingNodeKey, stdair::NestingStructureKey, stdair::OnDDateKey, stdair::ParsedKey, stdair::PolicyKey, stdair::PosChannelKey, stdair::SegmentCabinKey, stdair::SegmentDateKey, stdair::SegmentPeriodKey, stdair::SegmentSnapshotTableKey, stdair::TimePeriodKey, stdair::YieldFeaturesKey, and stdair::YieldStoreKey.

Definition at line 56 of file KeyAbstract.hpp.

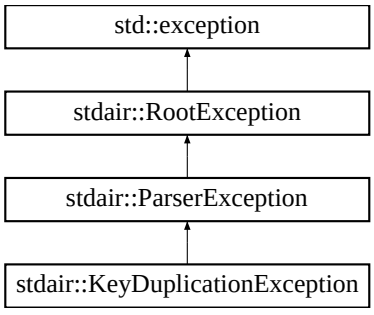
The documentation for this struct was generated from the following file:

- stdair/bom/KeyAbstract.hpp

8.92 stdair::KeyDuplicationException Class Reference

`#include <stdair/stdair_exceptions.hpp>`

Inheritance diagram for stdair::KeyDuplicationException::



Public Member Functions

- KeyDuplicationException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

8.92.1 Detailed Description

Key duplication.

Definition at line 149 of file stdair_exceptions.hpp.

8.92.2 Constructor & Destructor Documentation

8.92.2.1 stdair::KeyDuplicationException::KeyDuplicationException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 152 of file stdair_exceptions.hpp.

8.92.3 Member Function Documentation

8.92.3.1 const char* stdair::RootException::what () const throw () [inline, inherited]

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

8.92.4 Member Data Documentation

8.92.4.1 std::string stdair::RootException::_what [protected, inherited]

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

Referenced by stdair::RootException::what().

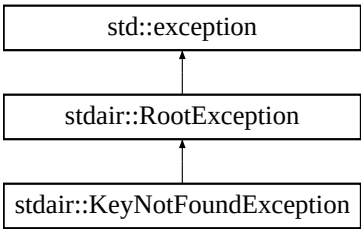
The documentation for this class was generated from the following file:

- stdair/stdair_exceptions.hpp

8.93 stdair::KeyNotFoundException Class Reference

`#include <stdair/stdair_exceptions.hpp>`

Inheritance diagram for stdair::KeyNotFoundException::



Public Member Functions

- KeyNotFoundException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

8.93.1 Detailed Description

Not found key.

Definition at line 126 of file stdair_exceptions.hpp.

8.93.2 Constructor & Destructor Documentation

8.93.2.1 stdair::KeyNotFoundException::KeyNotFoundException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 129 of file stdair_exceptions.hpp.

8.93.3 Member Function Documentation

8.93.3.1 const char* stdair::RootException::what () const throw () [inline, inherited]

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

8.93.4 Member Data Documentation

8.93.4.1 std::string stdair::RootException::_what [protected, inherited]

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

Referenced by stdair::RootException::what().

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

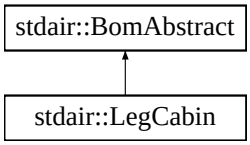
- stdair/stdair_exceptions.hpp

8.94 stdair::LegCabin Class Reference

Class representing the actual attributes for an airline leg-cabin.

`#include <stdair/bom/LegCabin.hpp>`

Inheritance diagram for stdair::LegCabin::



Public Types

- typedef LegCabinKey Key_T

Public Member Functions

- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const CabinCode_T & getCabinCode () const
- const MapKey_T getFullerKey () const
- const HolderMap_T & getHolderMap () const
- const CabinCapacity_T & getOfferedCapacity () const
- const CabinCapacity_T & getPhysicalCapacity () const
- const NbOfSeats_T & getSoldSeat () const
- const CommittedSpace_T & getCommittedSpace () const
- const Availability_T & getAvailabilityPool () const
- const Availability_T & getAvailability () const
- const BidPrice_T & getCurrentBidPrice () const
- const BidPrice_T & getPreviousBidPrice () const
- const BidPriceVector_T & getBidPriceVector () const
- const CapacityAdjustment_T & getRegradeAdjustment () const
- const AuthorizationLevel_T & getAuthorizationLevel () const
- const UPR_T & getUPR () const
- const Availability_T & getNetAvailability () const
- const Availability_T & getGrossAvailability () const
- const OverbookingRate_T & getAvgCancellationPercentage () const
- const NbOfSeats_T & getETB () const
- const NbOfSeats_T & getStaffNbOfSeats () const
- const NbOfSeats_T & getWLNbOfSeats () const
- const NbOfSeats_T & getGroupNbOfSeats () const
- VirtualClassList_T & getVirtualClassList ()
- BidPriceVector_T & getBidPriceVector ()
- const YieldLevelDemandMap_T & getYieldLevelDemandMap ()
- void setCapacities (const CabinCapacity_T &iCapacity)
- void setSoldSeat (const NbOfSeats_T &iSoldSeat)
- void setCommittedSpace (const CommittedSpace_T &iCommittedSpace)
- void setAvailabilityPool (const Availability_T &iAvailabilityPool)
- void setAvailability (const Availability_T &iAvailability)
- void setCurrentBidPrice (const BidPrice_T &iBidPrice)
- void setPreviousBidPrice (const BidPrice_T &iBidPrice)
- void updatePreviousBidPrice ()

- void setRegradeAdjustment (const CapacityAdjustment_T &iRegradeAdjustment)
- void setAuthorizationLevel (const AuthorizationLevel_T &iAU)
- void setUPR (const UPR_T &iUPR)
- void setNetAvailability (const Availability_T &iNAV)
- void setGrossAvailability (const Availability_T &iGAV)
- void setAvgCancellationPercentage (const OverbookingRate_T &iACP)
- void setETB (const NbOfSeats_T &iETB)
- void setStaffNbOfSeats (const NbOfSeats_T &iStaffSeats)
- void setWLNbOfSeats (const NbOfSeats_T &iWLSeats)
- void setGroupNbOfSeats (const NbOfSeats_T &iGroupSeats)
- void updateCurrentBidPrice ()
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- const std::string displayVirtualClassList () const
- void updateFromReservation (const NbOfBookings_T &)
- void addVirtualClass (const VirtualClassStruct &iVC)
- void emptyVirtualClassList ()
- void emptyBidPriceVector ()
- void addDemandInformation (const YieldValue_T &, const MeanValue_T &, const StdDevValue_T &)
- void emptyYieldLevelDemandMap ()

Public Attributes

- CapacityAdjustment_T _dcsRegrade
- AuthorizationLevel_T _au
- UPR_T _upr
- Availability_T _nav
- Availability_T _gav
- OverbookingRate_T _acp
- NbOfSeats_T _etb
- NbOfSeats_T _staffNbOfBookings
- NbOfSeats_T _wlNbOfBookings
- NbOfSeats_T _groupNbOfBookings

Protected Member Functions

- LegCabin (const Key_T &)
- ~LegCabin ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent
- HolderMap_T _holderMap
- CabinCapacity_T _offeredCapacity
- CabinCapacity_T _physicalCapacity

- NbOfSeats_T _soldSeat
- CommittedSpace_T _committedSpace
- Availability_T _availabilityPool
- Availability_T _availability
- BidPrice_T _currentBidPrice
- BidPrice_T _previousBidPrice
- BidPriceVector_T _bidPriceVector
- VirtualClassList_T _virtualClassList
- YieldLevelDemandMap_T _yieldLevelDemandMap

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager

8.94.1 Detailed Description

Class representing the actual attributes for an airline leg-cabin.

Definition at line 25 of file LegCabin.hpp.

8.94.2 Member Typedef Documentation

8.94.2.1 typedef LegCabinKey stdair::LegCabin::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 35 of file LegCabin.hpp.

8.94.3 Constructor & Destructor Documentation

8.94.3.1 stdair::LegCabin::LegCabin (const Key_T &) [protected]

Constructor.

Definition at line 46 of file LegCabin.cpp.

8.94.3.2 stdair::LegCabin::~LegCabin () [protected]

Destructor.

Definition at line 69 of file LegCabin.cpp.

8.94.4 Member Function Documentation

8.94.4.1 const Key_T& stdair::LegCabin::getKey () const [inline]

Get the leg-cabin key (cabin code).

Definition at line 42 of file LegCabin.hpp.

References _key.

8.94.4.2 BomAbstract* const stdair::LegCabin::getParent () const [inline]

Get the parent object.

Definition at line 49 of file LegCabin.hpp.

References _parent.

8.94.4.3 const CabinCode_T& stdair::LegCabin::getCabinCode () const [inline]

Get the cabin code (from key).

Definition at line 56 of file LegCabin.hpp.

References _key, and stdair::LegCabinKey::getCabinCode().

Referenced by getFullerKey().

8.94.4.4 const MapKey_T stdair::LegCabin::getFullerKey () const

Get the (leg-date, leg-cabin) key (board point and cabin code).

Note:

That method assumes that the parent object derives from the SegmentDate class, as it needs to have access to the describeKey() method.

Definition at line 80 of file LegCabin.cpp.

References stdair::DEFAULT_KEY_FLD_DELIMITER, stdair::LegDate::describeKey(), and getCabinCode().

8.94.4.5 const HolderMap_T& stdair::LegCabin::getHolderMap () const [inline]

Get the map of children holders.

Definition at line 72 of file LegCabin.hpp.

References _holderMap.

8.94.4.6 const CabinCapacity_T& stdair::LegCabin::getOfferedCapacity () const [inline]

Get the cabin offered capacity.

Definition at line 77 of file LegCabin.hpp.

References _offeredCapacity.

8.94.4.7 const CabinCapacity_T& stdair::LegCabin::getPhysicalCapacity () const [inline]

Get the cabin physical capacity.

Definition at line 82 of file LegCabin.hpp.

References _physicalCapacity.

8.94.4.8 const NbOfSeats_T& stdair::LegCabin::getSoldSeat () const [inline]

Get the number of sold seat.

Definition at line 87 of file LegCabin.hpp.

References _soldSeat.

8.94.4.9 const CommittedSpace_T& stdair::LegCabin::getCommittedSpace () const [inline]

Get the value of committed space.

Definition at line 92 of file LegCabin.hpp.

References _committedSpace.

8.94.4.10 const Availability_T& stdair::LegCabin::getAvailabilityPool () const [inline]

Get the value of the availability pool.

Definition at line 97 of file LegCabin.hpp.

References _availabilityPool.

8.94.4.11 const Availability_T& stdair::LegCabin::getAvailability () const [inline]

Get the value of the availability.

Definition at line 102 of file LegCabin.hpp.

References _availability.

8.94.4.12 const BidPrice_T& stdair::LegCabin::getCurrentBidPrice () const [inline]

Get the current Bid-Price.

Definition at line 107 of file LegCabin.hpp.

References _currentBidPrice.

8.94.4.13 const BidPrice_T& stdair::LegCabin::getPreviousBidPrice () const [inline]

Get the previous Bid-Price.

Definition at line 112 of file LegCabin.hpp.

References _previousBidPrice.

8.94.4.14 const BidPriceVector_T& stdair::LegCabin::getBidPriceVector () const [inline]

Get the Bid-Price Vector.

Definition at line 117 of file LegCabin.hpp.

References _bidPriceVector.

8.94.4.15 const CapacityAdjustment_T& stdair::LegCabin::getRegradeAdjustment () const [inline]

Get the capacity adjustment due to check-in (DCS) regrade.

Definition at line 122 of file LegCabin.hpp.

References _dcsRegrade.

8.94.4.16 const AuthorizationLevel_T& stdair::LegCabin::getAuthorizationLevel () const [inline]

Authorisation Level (AU).

Definition at line 127 of file LegCabin.hpp.

References _au.

8.94.4.17 const UPR_T& stdair::LegCabin::getUPR () const [inline]

Unsold Protection (UPR).

Definition at line 132 of file LegCabin.hpp.

References _upr.

8.94.4.18 const Availability_T& stdair::LegCabin::getNetAvailability () const [inline]

Net Availability (NAV).

Definition at line 137 of file LegCabin.hpp.

References _nav.

8.94.4.19 const Availability_T& stdair::LegCabin::getGrossAvailability () const [inline]

Gross Availability (GAV).

Definition at line 142 of file LegCabin.hpp.

References _gav.

8.94.4.20 const OverbookingRate_T& stdair::LegCabin::getAvgCancellationPercentage () const [inline]

Average Cancellation Percentage (ACP).

Definition at line 147 of file LegCabin.hpp.

References _acp.

8.94.4.21 const NbOfSeats_T& stdair::LegCabin::getETB () const [inline]

Expected to Board (ETB).

Definition at line 152 of file LegCabin.hpp.

References _etb.

8.94.4.22 const NbOfSeats_T& stdair::LegCabin::getStaffNbOfSeats () const [inline]

Number of staff bookings.

Definition at line 157 of file LegCabin.hpp.

References _staffNbOfBookings.

8.94.4.23 const NbOfSeats_T& stdair::LegCabin::getWLNbOfSeats () const [inline]

Number of wait-listed bookings.

Definition at line 162 of file LegCabin.hpp.

References _wlNbOfBookings.

8.94.4.24 const NbOfSeats_T& stdair::LegCabin::getGroupNbOfSeats () const [inline]

Number of group bookings.

Definition at line 167 of file LegCabin.hpp.

References _groupNbOfBookings.

8.94.4.25 VirtualClassList_T& stdair::LegCabin::getVirtualClassList () [inline]

The virtual class list.

Definition at line 172 of file LegCabin.hpp.

References _virtualClassList.

8.94.4.26 BidPriceVector_T& stdair::LegCabin::getBidPriceVector () [inline]

Reset the bid price vector and return it.

Definition at line 177 of file LegCabin.hpp.

References _bidPriceVector.

8.94.4.27 const YieldLevelDemandMap_T& stdair::LegCabin::getYieldLevelDemandMap () [inline]

Get the yield-demand map.

Definition at line 183 of file LegCabin.hpp.

References _yieldLevelDemandMap.

8.94.4.28 void stdair::LegCabin::setCapacities (const CabinCapacity_T & *iCapacity*)

Set the offered and physical capacities.

Definition at line 73 of file LegCabin.cpp.

References _committedSpace, _offeredCapacity, _physicalCapacity, and setAvailabilityPool().

8.94.4.29 void stdair::LegCabin::setSoldSeat (const NbOfSeats_T & *iSoldSeat*) [inline]

Set the number of sold seat.

Definition at line 194 of file LegCabin.hpp.

References _soldSeat.

8.94.4.30 void stdair::LegCabin::setCommittedSpace (const CommittedSpace_T & *iCommitted-Space*) [inline]

Set the value of committed space.

Definition at line 199 of file LegCabin.hpp.

References _committedSpace.

8.94.4.31 void stdair::LegCabin::setAvailabilityPool (const Availability_T & *iAvailabilityPool*) [inline]

Set the value of availability pool.

Definition at line 204 of file LegCabin.hpp.

References _availabilityPool.

Referenced by setCapacities().

8.94.4.32 void stdair::LegCabin::setAvailability (const Availability_T & *iAvailability*) [inline]

Set the value of availability.

Definition at line 209 of file LegCabin.hpp.

References _availability.

8.94.4.33 void stdair::LegCabin::setCurrentBidPrice (const BidPrice_T & *iBidPrice*) [inline]

Set the current Bid-Price.

Definition at line 214 of file LegCabin.hpp.

References _currentBidPrice.

8.94.4.34 void stdair::LegCabin::setPreviousBidPrice (const BidPrice_T & *iBidPrice*) [inline]

Set the previous Bid-Price.

Definition at line 219 of file LegCabin.hpp.

References _previousBidPrice.

8.94.4.35 void stdair::LegCabin::updatePreviousBidPrice () [inline]

Update the previous bid price value with the current one.

Definition at line 224 of file LegCabin.hpp.

References _currentBidPrice, and _previousBidPrice.

8.94.4.36 void stdair::LegCabin::setRegradeAdjustment (const CapacityAdjustment_T & *iRegradeAdjustment*) [inline]

Get the capacity adjustment due to check-in (DCS) regrade.

Definition at line 229 of file LegCabin.hpp.

References _dcsRegrade.

8.94.4.37 void stdair::LegCabin::setAuthorizationLevel (const AuthorizationLevel_T & *iAU*) [inline]

Set the Authorisation Level (AU).

Definition at line 234 of file LegCabin.hpp.

References _au.

8.94.4.38 void stdair::LegCabin::setUPR (const UPR_T & *iUPR*) [inline]

Set the Unsold Protection (UPR).

Definition at line 239 of file LegCabin.hpp.

References _upr.

8.94.4.39 void stdair::LegCabin::setNetAvailability (const Availability_T & *iNAV*) [inline]

Set the Net Availability (NAV).

Definition at line 244 of file LegCabin.hpp.

References _nav.

8.94.4.40 void stdair::LegCabin::setGrossAvailability (const Availability_T & *iGAV*) [inline]

Set the Gross Availability (GAV).

Definition at line 249 of file LegCabin.hpp.

References _gav.

8.94.4.41 void stdair::LegCabin::setAvgCancellationPercentage (const OverbookingRate_T & *iACP*) [inline]

Set the Average Cancellation Percentage (ACP).

Definition at line 254 of file LegCabin.hpp.

References _acp.

8.94.4.42 void stdair::LegCabin::setETB (const NbOfSeats_T & *iETB*) [inline]

Set the Expected to Board (ETB).

Definition at line 259 of file LegCabin.hpp.

References _etb.

8.94.4.43 void stdair::LegCabin::setStaffNbOfSeats (const NbOfSeats_T & *iStaffSeats*) [inline]

Set the Number of staff sold seats.

Definition at line 264 of file LegCabin.hpp.

References _staffNbOfBookings.

8.94.4.44 void stdair::LegCabin::setWLNbOfSeats (const NbOfSeats_T & *iWLSeats*) [inline]

Set the Number of wait-listed sold seats.

Definition at line 269 of file LegCabin.hpp.

References _wlNbOfBookings.

8.94.4.45 void stdair::LegCabin::setGroupNbOfSeats (const NbOfSeats_T & *iGroupSeats*) [inline]

Set the Number of group sold seats.

Definition at line 274 of file LegCabin.hpp.

References _groupNbOfBookings.

8.94.4.46 void stdair::LegCabin::updateCurrentBidPrice ()

Update the bid price (from bid price vector if not empty).

Definition at line 120 of file LegCabin.cpp.

References _availabilityPool, _bidPriceVector, and _currentBidPrice.

8.94.4.47 void stdair::LegCabin::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 288 of file LegCabin.hpp.

References toString().

8.94.4.48 void stdair::LegCabin::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 296 of file LegCabin.hpp.

8.94.4.49 std::string stdair::LegCabin::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 89 of file LegCabin.cpp.

References describeKey().

Referenced by toStream().

8.94.4.50 const std::string stdair::LegCabin::describeKey () const [inline]

Get a string describing the key.

Definition at line 307 of file LegCabin.hpp.

References _key, and stdair::LegCabinKey::toString().

Referenced by toString().

8.94.4.51 const std::string stdair::LegCabin::displayVirtualClassList () const

Display the virtual class list content.

Definition at line 96 of file LegCabin.cpp.

References _virtualClassList.

8.94.4.52 void stdair::LegCabin::updateFromReservation (const NbOfBookings_T &)

Register a sale.

Definition at line 114 of file LegCabin.cpp.

References _availabilityPool, _committedSpace, and _offeredCapacity.

8.94.4.53 void stdair::LegCabin::addVirtualClass (const VirtualClassStruct & *iVC*) [inline]

Add a virtual class to the list.

Definition at line 327 of file LegCabin.hpp.

References _virtualClassList.

8.94.4.54 void stdair::LegCabin::emptyVirtualClassList () [inline]

Empty the virtual class list.

Definition at line 334 of file LegCabin.hpp.

References _virtualClassList.

8.94.4.55 void stdair::LegCabin::emptyBidPriceVector () [inline]

Empty the bid price vector.

Definition at line 341 of file LegCabin.hpp.

References _bidPriceVector.

8.94.4.56 void stdair::LegCabin::addDemandInformation (const YieldValue_T &, const MeanValue_T &, const StdDevValue_T &)

Add demand information.

Definition at line 133 of file LegCabin.cpp.

References _yieldLevelDemandMap.

8.94.4.57 void stdair::LegCabin::emptyYieldLevelDemandMap () [inline]

Reset the (yield level,demand) map.

Definition at line 354 of file LegCabin.hpp.

References _yieldLevelDemandMap.

8.94.5 Friends And Related Function Documentation

8.94.5.1 friend class FacBom [friend]

Definition at line 26 of file LegCabin.hpp.

8.94.5.2 friend class FacCloneBom [friend]

Definition at line 27 of file LegCabin.hpp.

8.94.5.3 friend class FacBomManager [friend]

Definition at line 28 of file LegCabin.hpp.

8.94.6 Member Data Documentation

8.94.6.1 Key_T stdair::LegCabin::_key [protected]

Primary key (cabin code).

Definition at line 387 of file LegCabin.hpp.

Referenced by describeKey(), getCabinCode(), and getKey().

8.94.6.2 BomAbstract* stdair::LegCabin::_parent [protected]

Pointer on the parent class (LegDate).

Definition at line 392 of file LegCabin.hpp.

Referenced by getParent().

8.94.6.3 HolderMap_T stdair::LegCabin::_holderMap [protected]

Map holding the children (Bucket objects).

Definition at line 397 of file LegCabin.hpp.

Referenced by getHolderMap().

8.94.6.4 CabinCapacity_T stdair::LegCabin::_offeredCapacity [protected]

Saleable capacity of the cabin.

Definition at line 400 of file LegCabin.hpp.

Referenced by getOfferedCapacity(), setCapacities(), and updateFromReservation().

8.94.6.5 CabinCapacity_T stdair::LegCabin::_physicalCapacity [protected]

Physical capacity of the cabin.

Definition at line 403 of file LegCabin.hpp.

Referenced by getPhysicalCapacity(), and setCapacities().

8.94.6.6 NbOfSeats_T stdair::LegCabin::_soldSeat [protected]

Aggregated number of sold seats.

Definition at line 406 of file LegCabin.hpp.

Referenced by getSoldSeat(), and setSoldSeat().

8.94.6.7 CommittedSpace_T stdair::LegCabin::_committedSpace [protected]

Definition at line 409 of file LegCabin.hpp.

Referenced by getCommittedSpace(), setCapacities(), setCommittedSpace(), and updateFromReservation().

8.94.6.8 Availability_T stdair::LegCabin::_availabilityPool [protected]

Availability pool.

Definition at line 412 of file LegCabin.hpp.

Referenced by getAvailabilityPool(), setAvailabilityPool(), updateCurrentBidPrice(), and updateFromReservation().

8.94.6.9 Availability_T stdair::LegCabin::_availability [protected]

Availability.

Definition at line 415 of file LegCabin.hpp.

Referenced by getAvailability(), and setAvailability().

8.94.6.10 BidPrice_T stdair::LegCabin::_currentBidPrice [protected]

Current Bid-Price (BP).

Definition at line 418 of file LegCabin.hpp.

Referenced by getCurrentBidPrice(), setCurrentBidPrice(), updateCurrentBidPrice(), and updatePreviousBidPrice().

8.94.6.11 BidPrice_T stdair::LegCabin::_previousBidPrice [protected]

Previous Bid-Price (BP).

Definition at line 421 of file LegCabin.hpp.

Referenced by getPreviousBidPrice(), setPreviousBidPrice(), and updatePreviousBidPrice().

8.94.6.12 BidPriceVector_T stdair::LegCabin::_bidPriceVector [protected]

Bid-Price Vector (BPV).

Definition at line 424 of file LegCabin.hpp.

Referenced by emptyBidPriceVector(), getBidPriceVector(), and updateCurrentBidPrice().

8.94.6.13 VirtualClassList_T stdair::LegCabin::_virtualClassList [protected]

List of virtual classes (for revenue management optimisation).

Definition at line 427 of file LegCabin.hpp.

Referenced by addVirtualClass(), displayVirtualClassList(), emptyVirtualClassList(), and getVirtualClassList().

8.94.6.14 YieldLevelDemandMap_T stdair::LegCabin::_yieldLevelDemandMap [protected]

Map holding the demand information indexed by yield.

Definition at line 430 of file LegCabin.hpp.

Referenced by addDemandInformation(), emptyYieldLevelDemandMap(), and getYieldLevelDemandMap().

8.94.6.15 CapacityAdjustment_T stdair::LegCabin::_dcsRegrade

Capacity adjustment of the cabin, due to check-in (DCS) regrade.

Definition at line 435 of file LegCabin.hpp.

Referenced by getRegradeAdjustment(), and setRegradeAdjustment().

8.94.6.16 AuthorizationLevel_T stdair::LegCabin::_au

Authorisation Level (AU).

Definition at line 438 of file LegCabin.hpp.

Referenced by getAuthorizationLevel(), and setAuthorizationLevel().

8.94.6.17 UPR_T stdair::LegCabin::_upr

Unsold Protection (UPR).

Definition at line 441 of file LegCabin.hpp.

Referenced by getUPR(), and setUPR().

8.94.6.18 Availability_T stdair::LegCabin::_nav

Net Availability (NAV).

Definition at line 444 of file LegCabin.hpp.

Referenced by getNetAvailability(), and setNetAvailability().

8.94.6.19 Availability_T stdair::LegCabin::_gav

Gross Availability (GAV).

Definition at line 447 of file LegCabin.hpp.

Referenced by getGrossAvailability(), and setGrossAvailability().

8.94.6.20 OverbookingRate_T stdair::LegCabin::_acp

Average Cancellation Percentage (ACP).

Definition at line 450 of file LegCabin.hpp.

Referenced by getAvgCancellationPercentage(), and setAvgCancellationPercentage().

8.94.6.21 NbOfSeats_T stdair::LegCabin::_etb

Expected to Board (ETB).

Definition at line 453 of file LegCabin.hpp.

Referenced by getETB(), and setETB().

8.94.6.22 NbOfSeats_T stdair::LegCabin::_staffNbOfBookings

Number of staff bookings.

Definition at line 456 of file LegCabin.hpp.

Referenced by getStaffNbOfSeats(), and setStaffNbOfSeats().

8.94.6.23 NbOfSeats_T stdair::LegCabin::_wlNbOfBookings

Number of wait-listed bookings.

Definition at line 459 of file LegCabin.hpp.

Referenced by getWLNbOfSeats(), and setWLNbOfSeats().

8.94.6.24 NbOfSeats_T stdair::LegCabin::_groupNbOfBookings

Number of group bookings.

Definition at line 462 of file LegCabin.hpp.

Referenced by getGroupNbOfSeats(), and setGroupNbOfSeats().

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

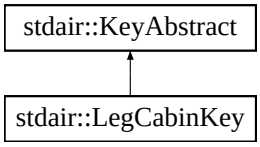
- stdair/bom/LegCabin.hpp
- stdair/bom/LegCabin.cpp

8.95 stdair::LegCabinKey Struct Reference

Key of a given leg-cabin, made of a cabin code (only).

`#include <stdair/bom/LegCabinKey.hpp>`

Inheritance diagram for stdair::LegCabinKey::



Public Member Functions

- LegCabinKey (const CabinCode_T &iCabinCode)
- LegCabinKey (const LegCabinKey &)
- ~LegCabinKey ()
- const CabinCode_T & getCabinCode () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const
- template<class Archive> void serialize (Archive &ar, const unsigned int iFileVersion)

Friends

- class boost::serialization::access

8.95.1 Detailed Description

Key of a given leg-cabin, made of a cabin code (only).

Definition at line 26 of file LegCabinKey.hpp.

8.95.2 Constructor & Destructor Documentation

8.95.2.1 stdair::LegCabinKey::LegCabinKey (const CabinCode_T & *iCabinCode*)

Constructor.

Definition at line 23 of file LegCabinKey.cpp.

8.95.2.2 stdair::LegCabinKey::LegCabinKey (const LegCabinKey &)

Copy constructor.

Definition at line 28 of file LegCabinKey.cpp.

8.95.2.3 stdair::LegCabinKey::~LegCabinKey ()

Destructor.

Definition at line 33 of file LegCabinKey.cpp.

8.95.3 Member Function Documentation

8.95.3.1 const CabinCode_T& stdair::LegCabinKey::getCabinCode () const [inline]

Get the cabin code.

Definition at line 56 of file LegCabinKey.hpp.

Referenced by stdair::LegCabin::getCabinCode().

8.95.3.2 void stdair::LegCabinKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 37 of file LegCabinKey.cpp.

References toString().

8.95.3.3 void stdair::LegCabinKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 42 of file LegCabinKey.cpp.

8.95.3.4 const std::string stdair::LegCabinKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from stdair::KeyAbstract.

Definition at line 46 of file LegCabinKey.cpp.

Referenced by stdair::LegCabin::describeKey(), stdair::LegDate::getLegCabin(), and toStream().

8.95.3.5 template<class Archive> void stdair::LegCabinKey::serialize (Archive & *ar*, const unsigned int *iFileVersion*)

Serialisation.

Definition at line 68 of file LegCabinKey.cpp.

8.95.4 Friends And Related Function Documentation

8.95.4.1 friend class boost::serialization::access [friend]

Definition at line 27 of file LegCabinKey.hpp.

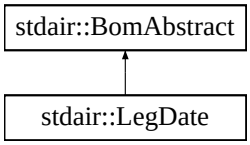
The documentation for this struct was generated from the following files:

- stdair/bom/LegCabinKey.hpp
- stdair/bom/LegCabinKey.cpp

8.96 stdair::LegDate Class Reference

`#include <stdair/bom/LegDate.hpp>`

Inheritance diagram for stdair::LegDate::



Public Types

- typedef LegDateKey Key_T

Public Member Functions

- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const AirportCode_T & getBoardingPoint () const
- const AirlineCode_T & getAirlineCode () const
- const HolderMap_T & getHolderMap () const
- LegCabin * getLegCabin (const std::string &iLegCabinKeyStr) const
- LegCabin * getLegCabin (const LegCabinKey &) const
- const AirportCode_T & getOffPoint () const
- const Date_T & getBoardingDate () const
- const Duration_T & getBoardingTime () const
- const Date_T & getOffDate () const
- const Duration_T & getOffTime () const
- const Duration_T & getElapsedTime () const
- const Distance_T & getDistance () const
- const CabinCapacity_T & getCapacity () const
- const DateOffset_T getDateOffset () const
- const Duration_T getTimeOffset () const
- void setOffPoint (const AirportCode_T &iOffPoint)
- void setBoardingDate (const Date_T &iBoardingDate)
- void setBoardingTime (const Duration_T &iBoardingTime)
- void setOffDate (const Date_T &iOffDate)
- void setOffTime (const Duration_T &iOffTime)
- void setElapsedTime (const Duration_T &)
- void setOperatingAirlineCode (const AirlineCode_T &iAirlineCode)
- void setOperatingFlightNumber (const FlightNumber_T &iFlightNumber)
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- const std::string describeRoutingKey () const

Protected Member Functions

- LegDate (const Key_T &)
- virtual ~LegDate ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent
- HolderMap_T _holderMap
- AirportCode_T _offPoint
- Date_T _boardingDate
- Duration_T _boardingTime
- Date_T _offDate
- Duration_T _offTime
- Duration_T _elapsedTime
- Distance_T _distance
- CabinCapacity_T _capacity
- AirlineCode_T _operatingAirlineCode
- FlightNumber_T _operatingFlightNumber

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager

8.96.1 Detailed Description

Class representing the actual attributes for an airline leg-date.

Definition at line 25 of file LegDate.hpp.

8.96.2 Member Typedef Documentation

8.96.2.1 typedef LegDateKey stdair::LegDate::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 33 of file LegDate.hpp.

8.96.3 Constructor & Destructor Documentation

8.96.3.1 stdair::LegDate::LegDate (const Key_T &) [protected]

Constructor.

Definition at line 38 of file LegDate.cpp.

8.96.3.2 stdair::LegDate::~LegDate () [protected, virtual]

Destructor.

Definition at line 44 of file LegDate.cpp.

8.96.4 Member Function Documentation

8.96.4.1 const Key_T& stdair::LegDate::getKey () const [inline]

Get the leg-date key.

Definition at line 39 of file LegDate.hpp.

References _key.

8.96.4.2 BomAbstract* const stdair::LegDate::getParent () const [inline]

Get the parent object.

Definition at line 44 of file LegDate.hpp.

References _parent.

Referenced by describeRoutingKey(), and getAirlineCode().

8.96.4.3 const AirportCode_T& stdair::LegDate::getBoardingPoint () const [inline]

Get the boarding point (part of the primary key).

Definition at line 49 of file LegDate.hpp.

References _key, and stdair::LegDateKey::getBoardingPoint().

8.96.4.4 const AirlineCode_T & stdair::LegDate::getAirlineCode () const

Get the airline code (key of the parent object).

Note:

That method assumes that the parent object derives from the Inventory class, as it needs to have access to the getAirlineCode() method.

Definition at line 48 of file LegDate.cpp.

References stdair::FlightDate::getAirlineCode(), and getParent().

8.96.4.5 const HolderMap_T& stdair::LegDate::getHolderMap () const [inline]

Get the map of children holders.

Definition at line 65 of file LegDate.hpp.

References _holderMap.

8.96.4.6 LegCabin * stdair::LegDate::getLegCabin (const std::string & *iLegCabinKeyStr*) const

Get a pointer on the LegCabin object corresponding to the given key.

Note:

The LegCabin object can be inherited from, if needed. In that case, a dynamic_cast<> may be needed.

Parameters:

const std::string& The leg-cabin key.

Returns:

LegCabin* Found LegCabin object. NULL if not found.

Definition at line 76 of file LegDate.cpp.

Referenced by getLegCabin(), and stdair::BomRetriever::retrieveDummyLegCabin().

8.96.4.7 LegCabin * stdair::LegDate::getLegCabin (const LegCabinKey &) const

Get a pointer on the LegCabin object corresponding to the given key.

Note:

The LegCabin object can be inherited from, if needed. In that case, a dynamic_cast<> may be needed.

Parameters:

const LegCabinKey& The leg-cabin key

Returns:

LegCabin* Found LegCabin object. NULL if not found.

Definition at line 83 of file LegDate.cpp.

References getLegCabin(), and stdair::LegCabinKey::toString().

8.96.4.8 const AirportCode_T& stdair::LegDate::getOffPoint () const [inline]

Get the off point.

Definition at line 94 of file LegDate.hpp.

References _offPoint.

8.96.4.9 const Date_T& stdair::LegDate::getBoardingDate () const [inline]

Get the boarding date.

Definition at line 99 of file LegDate.hpp.

References _boardingDate.

8.96.4.10 const Duration_T& stdair::LegDate::getBoardingTime () const [inline]

Get the boarding time.

Definition at line 104 of file LegDate.hpp.

References _boardingTime.

8.96.4.11 const Date_T& stdair::LegDate::getOffDate () const [inline]

Get the off date.

Definition at line 109 of file LegDate.hpp.

References _offDate.

8.96.4.12 const Duration_T& stdair::LegDate::getOffTime () const [inline]

Get the off time.

Definition at line 114 of file LegDate.hpp.

References _offTime.

8.96.4.13 const Duration_T& stdair::LegDate::getElapsedTime () const [inline]

Get the elapsed time.

Definition at line 119 of file LegDate.hpp.

References _elapsedTime.

8.96.4.14 const Distance_T& stdair::LegDate::getDistance () const [inline]

Get the distance.

Definition at line 124 of file LegDate.hpp.

References _distance.

8.96.4.15 const CabinCapacity_T& stdair::LegDate::getCapacity () const [inline]

Get the leg capacity.

Definition at line 129 of file LegDate.hpp.

References _capacity.

8.96.4.16 const DateOffset_T stdair::LegDate::getDateOffset () const [inline]

Get the date offset (off date - boarding date).

Definition at line 134 of file LegDate.hpp.

References _boardingDate, and _offDate.

Referenced by getTimeOffset().

8.96.4.17 const Duration_T stdair::LegDate::getTimeOffset () const

Get the time off set between boarding and off points.

It is defined as being: TimeOffset = (OffTime - BoardingTime) + (OffDate - BoardingDate) $*$ 24

- ElapsedTime.

Definition at line 88 of file LegDate.cpp.

References _boardingTime, _elapsedTime, _offTime, and getDateOffset().

8.96.4.18 void stdair::LegDate::setOffPoint (const AirportCode_T & *iOffPoint*) [inline]

Set the off point.

Definition at line 148 of file LegDate.hpp.

References _offPoint.

8.96.4.19 void stdair::LegDate::setBoardingDate (const Date_T & *iBoardingDate*) [inline]

Set the boarding date.

Definition at line 153 of file LegDate.hpp.

References _boardingDate.

8.96.4.20 void stdair::LegDate::setBoardingTime (const Duration_T & *iBoardingTime*) [inline]

Set the boarding time.

Definition at line 158 of file LegDate.hpp.

References _boardingTime.

8.96.4.21 void stdair::LegDate::setOffDate (const Date_T & *iOffDate*) [inline]

Set the off date.

Definition at line 163 of file LegDate.hpp.

References _offDate.

8.96.4.22 void stdair::LegDate::setOffTime (const Duration_T & *iOffTime*) [inline]

Set the off time.

Definition at line 168 of file LegDate.hpp.

References _offTime.

8.96.4.23 void stdair::LegDate::setElapsedTime (const Duration_T &)

Set the elapsed time.

Definition at line 103 of file LegDate.cpp.

References _elapsedTime.

8.96.4.24 void stdair::LegDate::setOperatingAirlineCode (const AirlineCode_T & *iAirlineCode*) [inline]

Set the operating airline code.

Definition at line 176 of file LegDate.hpp.

References _operatingAirlineCode.

8.96.4.25 void stdair::LegDate::setOperatingFlightNumber (const FlightNumber_T & *iFlightNumber*) [inline]

Set the operating flight number.

Definition at line 181 of file LegDate.hpp.

References _operatingFlightNumber.

8.96.4.26 void stdair::LegDate::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 194 of file LegDate.hpp.

References toString().

8.96.4.27 void stdair::LegDate::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 200 of file LegDate.hpp.

8.96.4.28 std::string stdair::LegDate::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 56 of file LegDate.cpp.

References describeKey().

Referenced by toStream().

8.96.4.29 const std::string stdair::LegDate::describeKey () const [inline]

Get a string describing the key.

Definition at line 207 of file LegDate.hpp.

References _key, and stdair::LegDateKey::toString().

Referenced by describeRoutingKey(), stdair::LegCabin::getFullerKey(), and toString().

8.96.4.30 const std::string stdair::LegDate::describeRoutingKey () const

Get a string describing the routing key.

Definition at line 63 of file LegDate.cpp.

References _operatingAirlineCode, _operatingFlightNumber, stdair::DEFAULT_KEY_FLD_-DELIMITER, describeKey(), stdair::FlightDate::getDepartureDate(), and getParent().

8.96.5 Friends And Related Function Documentation

8.96.5.1 friend class FacBom [friend]

Definition at line 26 of file LegDate.hpp.

8.96.5.2 friend class FacCloneBom [friend]

Definition at line 27 of file LegDate.hpp.

8.96.5.3 friend class FacBomManager [friend]

Definition at line 28 of file LegDate.hpp.

8.96.6 Member Data Documentation

8.96.6.1 Key_T stdair::LegDate::_key [protected]

Primary key (origin airport).

Definition at line 231 of file LegDate.hpp.

Referenced by describeKey(), getBoardingPoint(), and getKey().

8.96.6.2 BomAbstract* stdair::LegDate::_parent [protected]

Pointer on the parent class (FlightDate).

Definition at line 234 of file LegDate.hpp.

Referenced by getParent().

8.96.6.3 HolderMap_T stdair::LegDate::_holderMap [protected]

Map holding the children (LegCabin objects).

Definition at line 237 of file LegDate.hpp.

Referenced by getHolderMap().

8.96.6.4 AirportCode_T stdair::LegDate::_offPoint [protected]

Landing airport.

Definition at line 240 of file LegDate.hpp.

Referenced by getOffPoint(), and setOffPoint().

8.96.6.5 Date_T stdair::LegDate::_boardingDate [protected]

Boarding date.

Definition at line 243 of file LegDate.hpp.

Referenced by getBoardingDate(), getDateOffset(), and setBoardingDate().

8.96.6.6 Duration_T stdair::LegDate::_boardingTime [protected]

Boarding time.

Definition at line 246 of file LegDate.hpp.

Referenced by getBoardingTime(), getTimeOffset(), and setBoardingTime().

8.96.6.7 Date_T stdair::LegDate::_offDate [protected]

Landing date.

Definition at line 249 of file LegDate.hpp.

Referenced by getDateOffset(), getOffDate(), and setOffDate().

8.96.6.8 Duration_T stdair::LegDate::_offTime [protected]

Landing time.

Definition at line 252 of file LegDate.hpp.

Referenced by getOffTime(), getTimeOffset(), and setOffTime().

8.96.6.9 Duration_T stdair::LegDate::_elapsedTime [protected]

Trip elapsed time.

Definition at line 255 of file LegDate.hpp.

Referenced by getElapsedTime(), getTimeOffset(), and setElapsedTime().

8.96.6.10 Distance_T stdair::LegDate::_distance [protected]

Trip distance.

Definition at line 258 of file LegDate.hpp.

Referenced by getDistance().

8.96.6.11 CabinCapacity_T stdair::LegDate::_capacity [protected]

Aggregated capacity for all the leg-cabins.

Definition at line 261 of file LegDate.hpp.

Referenced by getCapacity().

8.96.6.12 AirlineCode_T stdair::LegDate::_operatingAirlineCode [protected]

Operating airline code.

Definition at line 264 of file LegDate.hpp.

Referenced by describeRoutingKey(), and setOperatingAirlineCode().

8.96.6.13 FlightNumber_T stdair::LegDate::_operatingFlightNumber [protected]

Operating flight number.

Definition at line 267 of file LegDate.hpp.

Referenced by describeRoutingKey(), and setOperatingFlightNumber().

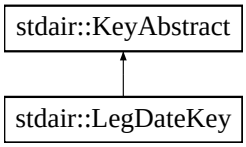
The documentation for this class was generated from the following files:

- stdair/bom/LegDate.hpp
- stdair/bom/LegDate.cpp

8.97 stdair::LegDateKey Struct Reference

`#include <stdair/bom/LegDateKey.hpp>`

Inheritance diagram for stdair::LegDateKey::



Public Member Functions

- LegDateKey (const AirportCode_T &iBoardingPoint)
- LegDateKey (const LegDateKey &)
- ~LegDateKey ()
- const AirportCode_T & getBoardingPoint () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const

8.97.1 Detailed Description

Key of a given leg-date, made of an origin airport.

Definition at line 16 of file LegDateKey.hpp.

8.97.2 Constructor & Destructor Documentation

8.97.2.1 stdair::LegDateKey::LegDateKey (const AirportCode_T & *iBoardingPoint*)

Constructor.

Definition at line 19 of file LegDateKey.cpp.

8.97.2.2 stdair::LegDateKey::LegDateKey (const LegDateKey &)

Default copy constructor.

Definition at line 24 of file LegDateKey.cpp.

8.97.2.3 stdair::LegDateKey::~LegDateKey ()

Destructor.

Definition at line 29 of file LegDateKey.cpp.

8.97.3 Member Function Documentation

8.97.3.1 const AirportCode_T& stdair::LegDateKey::getBoardingPoint () const [inline]

Get the boarding point.

Definition at line 34 of file LegDateKey.hpp.

Referenced by stdair::LegDate::getBoardingPoint().

8.97.3.2 void stdair::LegDateKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 33 of file LegDateKey.cpp.

References toString().

8.97.3.3 void stdair::LegDateKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 38 of file LegDateKey.cpp.

8.97.3.4 const std::string stdair::LegDateKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same leg-date.

Reimplemented from stdair::KeyAbstract.

Definition at line 42 of file LegDateKey.cpp.

Referenced by stdair::LegDate::describeKey(), stdair::FlightDate::getLegDate(), and toStream().

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

- stdair/bom/LegDateKey.hpp
- stdair/bom/LegDateKey.cpp

8.98 stdair::Logger Class Reference

`#include <stdair/service/Logger.hpp>`

Public Member Functions

- template<typename T> void log (const LOG::EN_LogLevel iLevel, const int iLineNumber, const std::string &iFileName, const T &iToBeLogged)

Static Public Member Functions

- static Logger & instance ()

Friends

- class FacSupervisor

  Friend classes.

- class STDAIR_Service

8.98.1 Detailed Description

Class holding the stream for logs.

Note that the error logs are seen as standard output logs, but with a higher level of visibility.

Definition at line 48 of file Logger.hpp.

8.98.2 Member Function Documentation

8.98.2.1 template<typename T> void stdair::Logger::log (const LOG::EN_LogLevel *iLevel*, const int *iLineNumber*, const std::string & *iFileName*, const T & *iToBeLogged*) [inline]

Main log entry.

Definition at line 59 of file Logger.hpp.

References stdair::LOG::_logLevels.

8.98.2.2 Logger & stdair::Logger::instance () [static]

Return the static Logger instance.

Definition at line 48 of file Logger.cpp.

8.98.3 Friends And Related Function Documentation

8.98.3.1 friend class FacSupervisor [friend]

Friend classes.

Definition at line 50 of file Logger.hpp.

8.98.3.2 friend class STDAIR_Service [friend]

Definition at line 51 of file Logger.hpp.

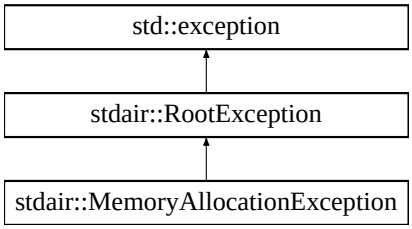
The documentation for this class was generated from the following files:

- stdair/service/Logger.hpp
- stdair/service/Logger.cpp

8.99 stdair::MemoryAllocationException Class Reference

`#include <stdair/stdair_exceptions.hpp>`

Inheritance diagram for stdair::MemoryAllocationException::



Public Member Functions

- MemoryAllocationException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

8.99.1 Detailed Description

Memory allocation.

Definition at line 89 of file stdair_exceptions.hpp.

8.99.2 Constructor & Destructor Documentation

8.99.2.1 stdair::MemoryAllocationException::MemoryAllocationException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 92 of file stdair_exceptions.hpp.

8.99.3 Member Function Documentation

8.99.3.1 const char* stdair::RootException::what () const throw () [inline, inherited]

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

8.99.4 Member Data Documentation

8.99.4.1 std::string stdair::RootException::_what [protected, inherited]

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

Referenced by stdair::RootException::what().

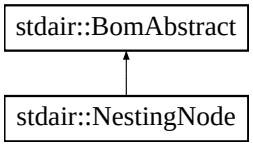
The documentation for this class was generated from the following file:

- stdair/stdair_exceptions.hpp

8.100 stdair::NestingNode Class Reference

`#include <stdair/bom/NestingNode.hpp>`

Inheritance diagram for stdair::NestingNode::



Public Types

- typedef NestingNodeKey Key_T

Public Member Functions

- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const HolderMap_T & getHolderMap () const
- const Yield_T & getYield () const
- void setYield (const Yield_T &iYield)
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- template<class Archive> void serialize (Archive &ar, const unsigned int iFileVersion)

Protected Member Functions

- NestingNode (const Key_T &)
- virtual ~NestingNode ()

Friends

- class FacBom
- class FacBomManager
- class boost::serialization::access

8.100.1 Detailed Description

Structure holding the elements of a nesting node. A nesting node is a set of booking classes.

Definition at line 29 of file NestingNode.hpp.

8.100.2 Member Typedef Documentation

8.100.2.1 typedef NestingNodeKey stdair::NestingNode::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 39 of file NestingNode.hpp.

8.100.3 Constructor & Destructor Documentation

8.100.3.1 stdair::NestingNode::NestingNode (const Key_T &) [protected]

Main constructor.

Definition at line 31 of file NestingNode.cpp.

8.100.3.2 stdair::NestingNode::~NestingNode () [protected, virtual]

Destructor.

Definition at line 35 of file NestingNode.cpp.

8.100.4 Member Function Documentation

8.100.4.1 const Key_T& stdair::NestingNode::getKey () const [inline]

Get the policy key.

Definition at line 44 of file NestingNode.hpp.

8.100.4.2 BomAbstract* const stdair::NestingNode::getParent () const [inline]

Get the parent object.

Definition at line 49 of file NestingNode.hpp.

8.100.4.3 const HolderMap_T& stdair::NestingNode::getHolderMap () const [inline]

Get the map of children holders.

Definition at line 56 of file NestingNode.hpp.

8.100.4.4 const Yield_T& stdair::NestingNode::getYield () const [inline]

Getter for the yield.

Definition at line 61 of file NestingNode.hpp.

8.100.4.5 void stdair::NestingNode::setYield (const Yield_T & *iYield*) [inline]

Setter for the yield.

Definition at line 68 of file NestingNode.hpp.

8.100.4.6 void stdair::NestingNode::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 80 of file NestingNode.hpp.

References toString().

8.100.4.7 void stdair::NestingNode::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 89 of file NestingNode.hpp.

8.100.4.8 std::string stdair::NestingNode::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 39 of file NestingNode.cpp.

References describeKey().

Referenced by toStream().

8.100.4.9 const std::string stdair::NestingNode::describeKey () const [inline]

Get a string describing the key.

Definition at line 100 of file NestingNode.hpp.

References stdair::NestingNodeKey::toString().

Referenced by toString().

8.100.4.10 template<class Archive> void stdair::NestingNode::serialize (Archive & *ar*, const unsigned int *iFileVersion*)

Serialisation.

8.100.5 Friends And Related Function Documentation

8.100.5.1 friend class FacBom [friend]

Definition at line 30 of file NestingNode.hpp.

8.100.5.2 friend class FacBomManager [friend]

Definition at line 31 of file NestingNode.hpp.

8.100.5.3 friend class boost::serialization::access [friend]

Definition at line 32 of file NestingNode.hpp.

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

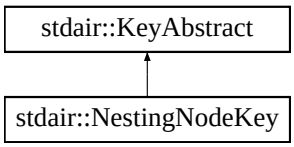
- stdair/bom/NestingNode.hpp
- stdair/bom/NestingNode.cpp

8.101 stdair::NestingNodeKey Struct Reference

Key of a given policy, made of a policy code.

`#include <stdair/bom/NestingNodeKey.hpp>`

Inheritance diagram for stdair::NestingNodeKey::



Public Member Functions

- NestingNodeKey (const NestingNodeCode_T &iNestingNodeCode)
- NestingNodeKey (const NestingNodeKey &)
- ~NestingNodeKey ()
- const NestingNodeCode_T & getNestingNodeCode () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const
- template<class Archive> void serialize (Archive &ar, const unsigned int iFileVersion)

Friends

- class boost::serialization::access

8.101.1 Detailed Description

Key of a given policy, made of a policy code.

Definition at line 26 of file NestingNodeKey.hpp.

8.101.2 Constructor & Destructor Documentation

8.101.2.1 stdair::NestingNodeKey::NestingNodeKey (const NestingNodeCode_T & *iNestingNodeCode*)

Constructor.

Definition at line 28 of file NestingNodeKey.cpp.

8.101.2.2 stdair::NestingNodeKey::NestingNodeKey (const NestingNodeKey &)

Copy constructor.

Definition at line 23 of file NestingNodeKey.cpp.

8.101.2.3 stdair::NestingNodeKey::~NestingNodeKey ()

Destructor.

Definition at line 33 of file NestingNodeKey.cpp.

8.101.3 Member Function Documentation

8.101.3.1 const NestingNodeCode_T& stdair::NestingNodeKey::getNestingNodeCode () const [inline]

Get the policy code.

Definition at line 56 of file NestingNodeKey.hpp.

8.101.3.2 void stdair::NestingNodeKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 37 of file NestingNodeKey.cpp.

References toString().

8.101.3.3 void stdair::NestingNodeKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 42 of file NestingNodeKey.cpp.

8.101.3.4 const std::string stdair::NestingNodeKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from stdair::KeyAbstract.

Definition at line 46 of file NestingNodeKey.cpp.

Referenced by stdair::NestingNode::describeKey(), and toStream().

8.101.3.5 template<class Archive> void stdair::NestingNodeKey::serialize (Archive & *ar*, const unsigned int *iFileVersion*)

Serialisation.

Definition at line 68 of file NestingNodeKey.cpp.

8.101.4 Friends And Related Function Documentation

8.101.4.1 friend class boost::serialization::access [friend]

Definition at line 27 of file NestingNodeKey.hpp.

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

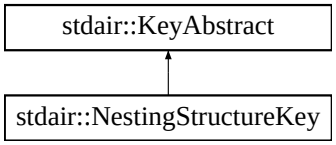
- stdair/bom/NestingNodeKey.hpp
- stdair/bom/NestingNodeKey.cpp

8.102 stdair::NestingStructureKey Struct Reference

Key of a given policy, made of a policy code.

`#include <stdair/bom/NestingStructureKey.hpp>`

Inheritance diagram for stdair::NestingStructureKey::



Public Member Functions

- NestingStructureKey (const NestingStructureCode_T &iNestingStructureCode)
- NestingStructureKey (const NestingStructureKey &)
- ~NestingStructureKey ()
- const NestingStructureCode_T & getNestingStructureCode () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const
- template<class Archive> void serialize (Archive &ar, const unsigned int iFileVersion)

Friends

- class boost::serialization::access

8.102.1 Detailed Description

Key of a given policy, made of a policy code.

Definition at line 26 of file NestingStructureKey.hpp.

8.102.2 Constructor & Destructor Documentation

8.102.2.1 stdair::NestingStructureKey::NestingStructureKey (const NestingStructureCode_T & *iNestingStructureCode*)

Constructor.

Definition at line 28 of file NestingStructureKey.cpp.

8.102.2.2 stdair::NestingStructureKey::NestingStructureKey (const NestingStructureKey &)

Copy constructor.

Definition at line 23 of file NestingStructureKey.cpp.

8.102.2.3 stdair::NestingStructureKey::~NestingStructureKey ()

Destructor.

Definition at line 33 of file NestingStructureKey.cpp.

8.102.3 Member Function Documentation

8.102.3.1 const NestingStructureCode_T& stdair::NestingStructureKey::getNestingStructureCode () const [inline]

Get the nesting structure code.

Definition at line 56 of file NestingStructureKey.hpp.

8.102.3.2 void stdair::NestingStructureKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 37 of file NestingStructureKey.cpp.

References toString().

8.102.3.3 void stdair::NestingStructureKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 42 of file NestingStructureKey.cpp.

8.102.3.4 const std::string stdair::NestingStructureKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from stdair::KeyAbstract.

Definition at line 46 of file NestingStructureKey.cpp.

Referenced by stdair::SimpleNestingStructure::describeKey(), and toStream().

8.102.3.5 template<class Archive> void stdair::NestingStructureKey::serialize (Archive & *ar*, const unsigned int *iFileVersion*)

Serialisation.

Definition at line 68 of file NestingStructureKey.cpp.

8.102.4 Friends And Related Function Documentation

8.102.4.1 friend class boost::serialization::access [friend]

Definition at line 27 of file NestingStructureKey.hpp.

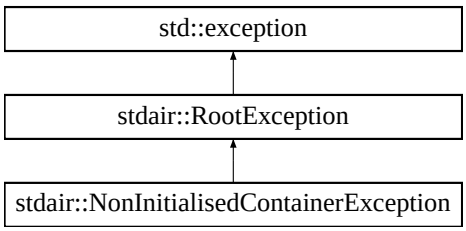
The documentation for this struct was generated from the following files:

- stdair/bom/NestingStructureKey.hpp
- stdair/bom/NestingStructureKey.cpp

8.103 stdair::NonInitialisedContainerException Class Reference

`#include <stdair/stdair_exceptions.hpp>`

Inheritance diagram for stdair::NonInitialisedContainerException::



Public Member Functions

- NonInitialisedContainerException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

8.103.1 Detailed Description

Non initialised container.

Definition at line 73 of file stdair_exceptions.hpp.

8.103.2 Constructor & Destructor Documentation

8.103.2.1 stdair::NonInitialisedContainerException::NonInitialisedContainerException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 76 of file stdair_exceptions.hpp.

8.103.3 Member Function Documentation

8.103.3.1 const char* stdair::RootException::what () const throw () [inline, inherited]

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

8.103.4 Member Data Documentation

8.103.4.1 std::string stdair::RootException::_what [protected, inherited]

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

Referenced by stdair::RootException::what().

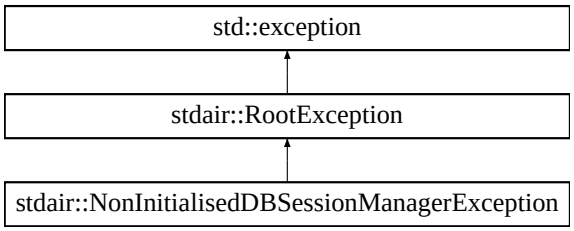
The documentation for this class was generated from the following file:

- stdair/stdair_exceptions.hpp

8.104 stdair::NonInitialisedDBSessionManagerException Class Reference

`#include <stdair/stdair_exceptions.hpp>`

Inheritance diagram for stdair::NonInitialisedDBSessionManagerException::



Public Member Functions

- NonInitialisedDBSessionManagerException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

8.104.1 Detailed Description

Non initialised database session.

Definition at line 188 of file stdair_exceptions.hpp.

8.104.2 Constructor & Destructor Documentation

8.104.2.1 stdair::NonInitialisedDBSessionManagerException::NonInitialisedDBSessionManagerException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 191 of file stdair_exceptions.hpp.

8.104.3 Member Function Documentation

8.104.3.1 const char* stdair::RootException::what () const throw () [inline, inherited]

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

8.104.4 Member Data Documentation

8.104.4.1 std::string stdair::RootException::_what [protected, inherited]

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

Referenced by stdair::RootException::what().

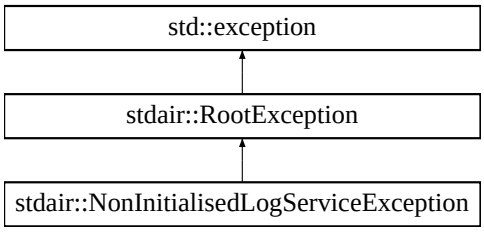
The documentation for this class was generated from the following file:

- stdair/stdair_exceptions.hpp

8.105 stdair::NonInitialisedLogServiceException Class Reference

`#include <stdair/stdair_exceptions.hpp>`

Inheritance diagram for stdair::NonInitialisedLogServiceException::



Public Member Functions

- NonInitialisedLogServiceException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

8.105.1 Detailed Description

Non initialised log service.

Definition at line 57 of file stdair_exceptions.hpp.

8.105.2 Constructor & Destructor Documentation

8.105.2.1 stdair::NonInitialisedLogServiceException::NonInitialisedLogServiceException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 60 of file stdair_exceptions.hpp.

8.105.3 Member Function Documentation

8.105.3.1 const char* stdair::RootException::what () const throw () [inline, inherited]

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

8.105.4 Member Data Documentation

8.105.4.1 std::string stdair::RootException::_what [protected, inherited]

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

Referenced by stdair::RootException::what().

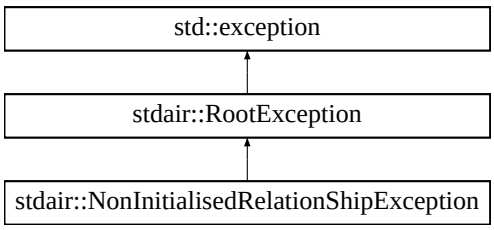
The documentation for this class was generated from the following file:

- stdair/stdair_exceptions.hpp

8.106 stdair::NonInitialisedRelationShipException Class Reference

`#include <stdair/stdair_exceptions.hpp>`

Inheritance diagram for stdair::NonInitialisedRelationShipException::



Public Member Functions

- NonInitialisedRelationShipException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

8.106.1 Detailed Description

Non initialised relationship.

Definition at line 81 of file stdair_exceptions.hpp.

8.106.2 Constructor & Destructor Documentation

8.106.2.1 stdair::NonInitialisedRelationShipException::NonInitialisedRelationShipException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 84 of file stdair_exceptions.hpp.

8.106.3 Member Function Documentation

8.106.3.1 const char* stdair::RootException::what () const throw () [inline, inherited]

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

8.106.4 Member Data Documentation

8.106.4.1 std::string stdair::RootException::_what [protected, inherited]

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

Referenced by stdair::RootException::what().

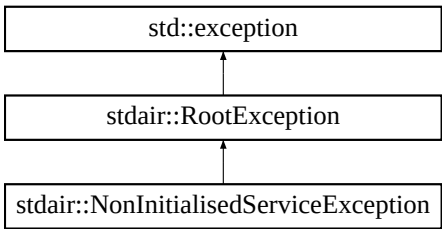
The documentation for this class was generated from the following file:

- stdair/stdair_exceptions.hpp

8.107 stdair::NonInitialisedServiceException Class Reference

`#include <stdair/stdair_exceptions.hpp>`

Inheritance diagram for stdair::NonInitialisedServiceException::



Public Member Functions

- NonInitialisedServiceException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

8.107.1 Detailed Description

Non initialised service.

Definition at line 65 of file stdair_exceptions.hpp.

8.107.2 Constructor & Destructor Documentation

8.107.2.1 stdair::NonInitialisedServiceException::NonInitialisedServiceException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 68 of file stdair_exceptions.hpp.

8.107.3 Member Function Documentation

8.107.3.1 const char* stdair::RootException::what () const throw () [inline, inherited]

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

8.107.4 Member Data Documentation

8.107.4.1 std::string stdair::RootException::_what [protected, inherited]

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

Referenced by stdair::RootException::what().

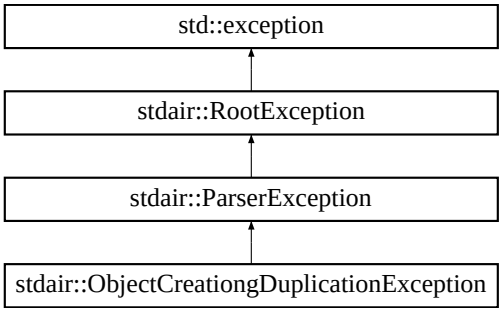
The documentation for this class was generated from the following file:

- stdair/stdair_exceptions.hpp

8.108 stdair::ObjectCreationgDuplicationException Class Reference

`#include <stdair/stdair_exceptions.hpp>`

Inheritance diagram for stdair::ObjectCreationgDuplicationException::



Public Member Functions

- ObjectCreationgDuplicationException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

8.108.1 Detailed Description

Duplicated object.

Definition at line 157 of file stdair_exceptions.hpp.

8.108.2 Constructor & Destructor Documentation

8.108.2.1 stdair::ObjectCreationgDuplicationException::ObjectCreationgDuplicationException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 160 of file stdair_exceptions.hpp.

8.108.3 Member Function Documentation

8.108.3.1 const char* stdair::RootException::what () const throw () [inline, inherited]

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

8.108.4 Member Data Documentation

8.108.4.1 std::string stdair::RootException::_what [protected, inherited]

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

Referenced by stdair::RootException::what().

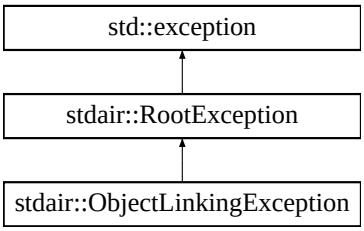
The documentation for this class was generated from the following file:

- stdair/stdair_exceptions.hpp

8.109 stdair::ObjectLinkingException Class Reference

`#include <stdair/stdair_exceptions.hpp>`

Inheritance diagram for stdair::ObjectLinkingException::



Public Member Functions

- ObjectLinkingException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

8.109.1 Detailed Description

Object link.

Definition at line 97 of file stdair_exceptions.hpp.

8.109.2 Constructor & Destructor Documentation

8.109.2.1 stdair::ObjectLinkingException::ObjectLinkingException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 100 of file stdair_exceptions.hpp.

8.109.3 Member Function Documentation

8.109.3.1 const char* stdair::RootException::what () const throw () [inline, inherited]

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

8.109.4 Member Data Documentation

8.109.4.1 std::string stdair::RootException::_what [protected, inherited]

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

Referenced by stdair::RootException::what().

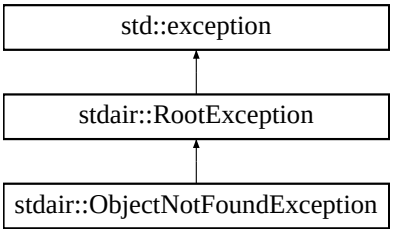
The documentation for this class was generated from the following file:

- stdair/stdair_exceptions.hpp

8.110 stdair::ObjectNotFoundException Class Reference

`#include <stdair/stdair_exceptions.hpp>`

Inheritance diagram for stdair::ObjectNotFoundException::



Public Member Functions

- ObjectNotFoundException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

8.110.1 Detailed Description

Not found object.

Definition at line 165 of file stdair_exceptions.hpp.

8.110.2 Constructor & Destructor Documentation

8.110.2.1 stdair::ObjectNotFoundException::ObjectNotFoundException (const std::string & *iWhat*) `[inline]`

Constructor.

Definition at line 168 of file stdair_exceptions.hpp.

8.110.3 Member Function Documentation

8.110.3.1 const char* stdair::RootException::what () const throw () `[inline, inherited]`

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

8.110.4 Member Data Documentation

8.110.4.1 std::string stdair::RootException::_what [protected, inherited]

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

Referenced by stdair::RootException::what().

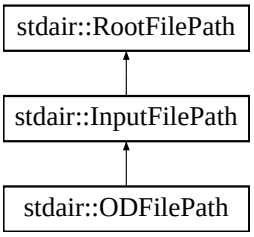
The documentation for this class was generated from the following file:

- stdair/stdair_exceptions.hpp

8.111 stdair::ODFilePath Class Reference

`#include <stdair/stdair_file.hpp>`

Inheritance diagram for stdair::ODFilePath::



Public Member Functions

- ODFilePath (const Filename_T &iFilename)
- const char * name () const

Protected Attributes

- const Filename_T _filename

8.111.1 Detailed Description

OD input file.

Definition at line 76 of file stdair_file.hpp.

8.111.2 Constructor & Destructor Documentation

8.111.2.1 stdair::ODFilePath::ODFilePath (const Filename_T & *iFilename*) [inline, explicit]

Constructor.

Definition at line 81 of file stdair_file.hpp.

8.111.3 Member Function Documentation

8.111.3.1 const char* stdair::RootFilePath::name () const [inline, inherited]

Give the details of the exception.

Definition at line 42 of file stdair_file.hpp.

References stdair::RootFilePath::_filename.

Referenced by stdair::BomINIImport::importINIConfig().

8.111.4 Member Data Documentation

8.111.4.1 const Filename_T stdair::RootFilePath::_filename [protected, inherited]

Name of the file.

Definition at line 50 of file stdair_file.hpp.

Referenced by stdair::RootFilePath::name().

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

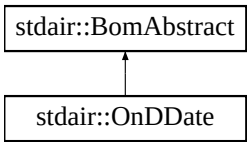
- stdair/stdair_file.hpp

8.112 stdair::OnDDate Class Reference

Class representing the actual attributes for an airline flight-date.

`#include <stdair/bom/OnDDate.hpp>`

Inheritance diagram for stdair::OnDDate::



Public Types

- typedef OnDDateKey Key_T

Public Member Functions

- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const AirlineCode_T & getAirlineCode () const
- const stdair::Date_T getDate () const
- const stdair::AirportCode_T getOrigin () const
- const stdair::AirportCode_T getDestination () const
- const HolderMap_T & getHolderMap () const
- const StringDemandStructMap_T & getDemandInfoMap () const
- const CabinForecastMap_T & getTotalForecastMap () const

- const WTPDemandPair_T & getTotalForecast (const CabinCode_T &iCC) const
- const CabinClassPairList_T & getCabinClassPairList (const std::string &iStr) const
- const short getNbOfSegments () const
- void setDemandInformation (const CabinClassPairList_T &, const YieldDemandPair_T &)
- void setTotalForecast (const CabinCode_T &, const WTPDemandPair_T &)
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- template<class Archive> void serialize (Archive &ar, const unsigned int iFileVersion)

Protected Member Functions

- OnDDate (const Key_T &)
- virtual ~OnDDate ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent
- HolderMap_T _holderMap
- StringDemandStructMap_T _classPathDemandMap
- StringCabinClassPairListMap_T _stringCabinClassPairListMap
- CabinForecastMap_T _cabinForecastMap

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager
- class boost::serialization::access

8.112.1 Detailed Description

Class representing the actual attributes for an airline flight-date.

Definition at line 33 of file OnDDate.hpp.

8.112.2 Member Typedef Documentation

8.112.2.1 typedef OnDDateKey stdair::OnDDate::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 44 of file OnDDate.hpp.

8.112.3 Constructor & Destructor Documentation

8.112.3.1 stdair::OnDDate::OnDDate (const Key_T &) [protected]

Main constructor.

Definition at line 28 of file OnDDate.cpp.

8.112.3.2 stdair::OnDDate::~OnDDate () [protected, virtual]

Destructor.

Definition at line 33 of file OnDDate.cpp.

8.112.4 Member Function Documentation

8.112.4.1 const Key_T& stdair::OnDDate::getKey () const [inline]

Get the O&D date key.

Definition at line 50 of file OnDDate.hpp.

References _key.

8.112.4.2 BomAbstract* const stdair::OnDDate::getParent () const [inline]

Get the parent object.

Definition at line 55 of file OnDDate.hpp.

References _parent.

Referenced by getAirlineCode().

8.112.4.3 const AirlineCode_T & stdair::OnDDate::getAirlineCode () const

Get the airline code (key of the parent object).

Note:

That method assumes that the parent object derives from the Inventory class, as it needs to have access to the getAirlineCode() method.

Definition at line 44 of file OnDDate.cpp.

References stdair::Inventory::getAirlineCode(), and getParent().

8.112.4.4 const stdair::Date_T stdair::OnDDate::getDate () const [inline]

Get the boarding date.

Definition at line 70 of file OnDDate.hpp.

References _key, and stdair::OnDDateKey::getDate().

8.112.4.5 const stdair::AirportCode_T stdair::OnDDate::getOrigin () const [inline]

Get the origin.

Definition at line 75 of file OnDDate.hpp.

References _key, and stdair::OnDDateKey::getOrigin().

8.112.4.6 const stdair::AirportCode_T stdair::OnDDate::getDestination () const [inline]

Get the destination.

Definition at line 80 of file OnDDate.hpp.

References _key, and stdair::OnDDateKey::getDestination().

8.112.4.7 const HolderMap_T& stdair::OnDDate::getHolderMap () const [inline]

Get the map of children holders.

Definition at line 87 of file OnDDate.hpp.

References _holderMap.

8.112.4.8 const StringDemandStructMap_T& stdair::OnDDate::getDemandInfoMap () const [inline]

Get the map of demand information.

Definition at line 94 of file OnDDate.hpp.

References _classPathDemandMap.

8.112.4.9 const CabinForecastMap_T& stdair::OnDDate::getTotalForecastMap () const [inline]

Get the map of total forecast.

Definition at line 101 of file OnDDate.hpp.

References _cabinForecastMap.

8.112.4.10 const WTPDemandPair_T& stdair::OnDDate::getTotalForecast (const CabinCode_T & *iCC*) const [inline]

Get the total forecast for a given cabin.

Definition at line 108 of file OnDDate.hpp.

References _cabinForecastMap.

8.112.4.11 const CabinClassPairList_T& stdair::OnDDate::getCabinClassPairList (const std::string & *iStr*) const [inline]

Get the cabin-class pair out of a string.

Definition at line 116 of file OnDDate.hpp.

References _stringCabinClassPairListMap.

8.112.4.12 const short stdair::OnDDate::getNbOfSegments () const [inline]

Get the number of segments of the O&D.

Definition at line 124 of file OnDDate.hpp.

References _key, and stdair::OnDDateKey::getNbOfSegments().

8.112.4.13 void stdair::OnDDate::setDemandInformation (const CabinClassPairList_T &, const YieldDemandPair_T &)

Set demand information.

Definition at line 53 of file OnDDate.cpp.

References _classPathDemandMap, and _stringCabinClassPairListMap.

8.112.4.14 void stdair::OnDDate::setTotalForecast (const CabinCode_T &, const WTPDemandPair_T &)

Set forecast information per cabin.

Definition at line 76 of file OnDDate.cpp.

References _cabinForecastMap.

8.112.4.15 void stdair::OnDDate::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 147 of file OnDDate.hpp.

References toString().

8.112.4.16 void stdair::OnDDate::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 156 of file OnDDate.hpp.

8.112.4.17 std::string stdair::OnDDate::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 37 of file OnDDate.cpp.

References describeKey().

Referenced by toStream().

8.112.4.18 const std::string stdair::OnDDate::describeKey () const [inline]

Get a string describing the key.

Definition at line 167 of file OnDDate.hpp.

References _key, and stdair::OnDDateKey::toString().

Referenced by toString().

8.112.4.19 template<class Archive> void stdair::OnDDate::serialize (Archive & *ar*, const unsigned int *iFileVersion*)

Serialisation.

8.112.5 Friends And Related Function Documentation

8.112.5.1 friend class FacBom [friend]

Definition at line 34 of file OnDDate.hpp.

8.112.5.2 friend class FacCloneBom [friend]

Definition at line 35 of file OnDDate.hpp.

8.112.5.3 friend class FacBomManager [friend]

Definition at line 36 of file OnDDate.hpp.

8.112.5.4 friend class boost::serialization::access [friend]

Definition at line 37 of file OnDDate.hpp.

8.112.6 Member Data Documentation

8.112.6.1 Key_T stdair::OnDDate::_key [protected]

Primary key (list of OnD string keys).

Definition at line 217 of file OnDDate.hpp.

Referenced by describeKey(), getDate(), getDestination(), getKey(), getNbOfSegments(), and getOrigin().

8.112.6.2 BomAbstract* stdair::OnDDate::_parent [protected]

Pointer on the parent class (Inventory).

Definition at line 222 of file OnDDate.hpp.

Referenced by getParent().

8.112.6.3 HolderMap_T stdair::OnDDate::_holderMap [protected]

Map holding the children (SegmentDate and LegDate objects).

Definition at line 227 of file OnDDate.hpp.

Referenced by getHolderMap().

8.112.6.4 StringDemandStructMap_T stdair::OnDDate::_classPathDemandMap [protected]

O&D demand information.

Definition at line 232 of file OnDDate.hpp.

Referenced by getDemandInfoMap(), and setDemandInformation().

8.112.6.5 StringCabinClassPairListMap_T stdair::OnDDate::_stringCabinClassPairListMap [protected]

O&D cabin and associated class map.

Definition at line 237 of file OnDDate.hpp.

Referenced by getCabinClassPairList(), and setDemandInformation().

8.112.6.6 CabinForecastMap_T stdair::OnDDate::_cabinForecastMap [protected]

O&D demand total forecast.

Definition at line 242 of file OnDDate.hpp.

Referenced by getTotalForecast(), getTotalForecastMap(), and setTotalForecast().

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

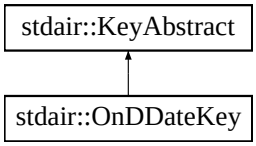
- stdair/bom/OnDDate.hpp
- stdair/bom/OnDDate.cpp

8.113 stdair::OnDDateKey Struct Reference

Key of a given O&D-date, made of a list of OnD strings. a OnD string contains the airline code, the flight number, the date and the segment (origin and destination).

`#include <stdair/bom/OnDDateKey.hpp>`

Inheritance diagram for stdair::OnDDateKey::



Public Member Functions

- OnDDateKey (const OnDStringList_T &)
- OnDDateKey (const OnDDateKey &)
- ~OnDDateKey ()
- const Date_T getDate () const
- const AirportCode_T getOrigin () const
- const AirportCode_T getDestination () const
- const short getNbOfSegments () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const
- template<class Archive> void serialize (Archive &ar, const unsigned int iFileVersion)

Friends

- class boost::serialization::access

8.113.1 Detailed Description

Key of a given O&D-date, made of a list of OnD strings. a OnD string contains the airline code, the flight number, the date and the segment (origin and destination).

Definition at line 23 of file OnDDateKey.hpp.

8.113.2 Constructor & Destructor Documentation

8.113.2.1 stdair::OnDDateKey::OnDDateKey (const OnDStringList_T &)

Constructor.

Definition at line 33 of file OnDDateKey.cpp.

8.113.2.2 stdair::OnDDateKey::OnDDateKey (const OnDDateKey &)

Copy constructor.

Definition at line 38 of file OnDDateKey.cpp.

8.113.2.3 stdair::OnDDateKey::~OnDDateKey ()

Destructor.

Definition at line 43 of file OnDDateKey.cpp.

8.113.3 Member Function Documentation

8.113.3.1 const Date_T stdair::OnDDateKey::getDate () const

Get the boarding date.

Definition at line 47 of file OnDDateKey.cpp.

References stdair::BomKeyManager::extractFlightDateKey(), and stdair::FlightDateKey::getDepartureDate().

Referenced by stdair::OnDDate::getDate().

8.113.3.2 const AirportCode_T stdair::OnDDateKey::getOrigin () const

Get the origin.

Definition at line 54 of file OnDDateKey.cpp.

References stdair::BomKeyManager::extractSegmentDateKey(), and stdair::SegmentDateKey::getBoardingPoint().

Referenced by stdair::OnDDate::getOrigin().

8.113.3.3 const AirportCode_T stdair::OnDDateKey::getDestination () const

Get the destination.

Definition at line 61 of file OnDDateKey.cpp.

References stdair::BomKeyManager::extractSegmentDateKey(), and stdair::SegmentDateKey::getOffPoint().

Referenced by stdair::OnDDate::getDestination().

8.113.3.4 const short stdair::OnDDateKey::getNbOfSegments () const [inline]

Get the number of segments.

Definition at line 70 of file OnDDateKey.hpp.

Referenced by stdair::OnDDate::getNbOfSegments().

8.113.3.5 void stdair::OnDDateKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 68 of file OnDDateKey.cpp.

References toString().

8.113.3.6 void stdair::OnDDateKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 73 of file OnDDateKey.cpp.

8.113.3.7 const std::string stdair::OnDDateKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from stdair::KeyAbstract.

Definition at line 77 of file OnDDateKey.cpp.

Referenced by stdair::OnDDate::describeKey(), and toStream().

8.113.3.8 template<class Archive> void stdair::OnDDateKey::serialize (Archive & *ar*, const unsigned int *iFileVersion*)

Serialisation.

Definition at line 102 of file OnDDateKey.cpp.

8.113.4 Friends And Related Function Documentation

8.113.4.1 friend class boost::serialization::access [friend]

Definition at line 24 of file OnDDateKey.hpp.

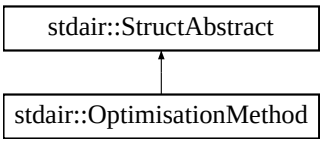
The documentation for this struct was generated from the following files:

- stdair/bom/OnDDateKey.hpp
- stdair/bom/OnDDateKey.cpp

8.114 stdair::OptimisationMethod Struct Reference

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

Inheritance diagram for stdair::OptimisationMethod::



Public Types

- LEG_BASED_MC = 0
- LEG_BASED_EMSR_B
- LAST_VALUE
- enum EN_OptimisationMethod { LEG_BASED_MC = 0, LEG_BASED_EMSR_B, LAST_VALUE }

Public Member Functions

- EN_OptimisationMethod getMethod () const
- std::string getMethodAsString () const
- const std::string describe () const
- bool operator== (const EN_OptimisationMethod &) const
- OptimisationMethod (const EN_OptimisationMethod &)
- OptimisationMethod (const char iMethod)
- OptimisationMethod (const OptimisationMethod &)
- void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

Static Public Member Functions

- static const std::string & getLabel (const EN_OptimisationMethod &)
- static char getMethodLabel (const EN_OptimisationMethod &)
- static std::string getMethodLabelAsString (const EN_OptimisationMethod &)
- static std::string describeLabels ()

8.114.1 Detailed Description

Enumeration of Optimisation methods.

Definition at line 15 of file OptimisationMethod.hpp.

8.114.2 Member Enumeration Documentation

8.114.2.1 enum stdair::OptimisationMethod::EN_OptimisationMethod

Enumerator:

LEG_BASED_MC

LEG_BASED_EMSR_B

LAST_VALUE

Definition at line 17 of file OptimisationMethod.hpp.

8.114.3 Constructor & Destructor Documentation

8.114.3.1 stdair::OptimisationMethod::OptimisationMethod (const EN_OptimisationMethod &)

Constructor.

Definition at line 36 of file OptimisationMethod.cpp.

8.114.3.2 stdair::OptimisationMethod::OptimisationMethod (const char *iMethod*)

Constructor.

Definition at line 41 of file OptimisationMethod.cpp.

References describeLabels(), LAST_VALUE, LEG_BASED_EMSR_B, and LEG_BASED_MC.

8.114.3.3 stdair::OptimisationMethod::OptimisationMethod (const OptimisationMethod &)

Default copy constructor.

Definition at line 30 of file OptimisationMethod.cpp.

8.114.4 Member Function Documentation

8.114.4.1 const std::string & stdair::OptimisationMethod::getLabel (const EN_OptimisationMethod &) [static]

Get the label as a string (e.g., "Leg based Monte Carlo" or "Leg based EMSRb").

Definition at line 59 of file OptimisationMethod.cpp.

8.114.4.2 char stdair::OptimisationMethod::getMethodLabel (const EN_OptimisationMethod &) [static]

Get the label as a single char (e.g., 'M' or 'E').

Definition at line 64 of file OptimisationMethod.cpp.

8.114.4.3 std::string stdair::OptimisationMethod::getMethodLabelAsString (const EN_OptimisationMethod &) [static]

Get the label as a string of a single char (e.g., "M" or "E").

Definition at line 70 of file OptimisationMethod.cpp.

8.114.4.4 std::string stdair::OptimisationMethod::describeLabels () [static]

List the labels.

Definition at line 77 of file OptimisationMethod.cpp.

References LAST_VALUE.

Referenced by OptimisationMethod().

8.114.4.5 OptimisationMethod::EN_OptimisationMethod stdair::OptimisationMethod::getMethod () const

Get the enumerated value.

Definition at line 89 of file OptimisationMethod.cpp.

Referenced by stdair::AirlineFeature::getOptimisationMethod().

8.114.4.6 std::string stdair::OptimisationMethod::getMethodAsString () const

Get the enumerated value as a short string (e.g., "M" or "E").

Definition at line 94 of file OptimisationMethod.cpp.

8.114.4.7 const std::string stdair::OptimisationMethod::describe () const [virtual]

Give a description of the structure (e.g., "Leg based Monte Carlo" or "Leg based EMSRb").

Implements stdair::StructAbstract.

Definition at line 101 of file OptimisationMethod.cpp.

8.114.4.8 bool stdair::OptimisationMethod::operator== (const EN_OptimisationMethod &) const

Comparaison operator.

Definition at line 109 of file OptimisationMethod.cpp.

8.114.4.9 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const [inline, inherited]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct,

stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References stdair::StructAbstract::describe().

8.114.4.10 virtual void stdair::StructAbstract::fromStream (std::istream & *ioIn*) [inline, virtual, inherited]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented in stdair::ProgressStatusSet, stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::EventStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 38 of file StructAbstract.hpp.

Referenced by operator>>().

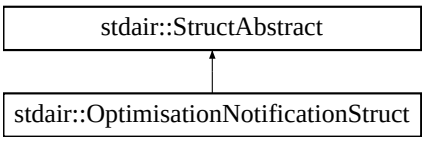
The documentation for this struct was generated from the following files:

- stdair/basic/OptimisationMethod.hpp
- stdair/basic/OptimisationMethod.cpp

8.115 stdair::OptimisationNotificationStruct Struct Reference

`#include <stdair/bom/OptimisationNotificationStruct.hpp>`

Inheritance diagram for stdair::OptimisationNotificationStruct::



Public Member Functions

- const AirportCode_T & getOrigin () const
- const AirportCode_T & getDestination () const
- const CityCode_T & getPOS () const
- const Date_T & getPreferedDepartureDate () const
- const DateTime_T & getNotificationDateTime () const
- const CabinCode_T & getPreferredCabin () const
- const NbOfSeats_T & getPartySize () const
- const ChannelLabel_T & getOptimisationChannel () const

- const TripType_T & getTripType () const
- const DayDuration_T & getStayDuration () const
- const FrequentFlyer_T & getFrequentFlyerType () const
- const Duration_T & getPreferredDepartureTime () const
- const WTP_T & getWTP () const
- const PriceValue_T & getValueOfTime () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string describe () const
- OptimisationNotificationStruct (const AirportCode_T &iOrigin, const AirportCode_T &iDestination, const CityCode_T &iPOS, const Date_T &iDepartureDate, const DateTime_T &iNotificationDateTime, const CabinCode_T &iPreferredCabin, const NbOfSeats_T &iPartySize, const ChannelLabel_T &iChannel, const TripType_T &iTripType, const DayDuration_T &iStayDuration, const FrequentFlyer_T &iFrequentFlyerType, const Duration_T &iPreferredDepartureTime, const WTP_T &iWTP, const PriceValue_T &iValueOfTime)
- OptimisationNotificationStruct (const OptimisationNotificationStruct &)
- ~OptimisationNotificationStruct ()

8.115.1 Detailed Description

Structure holding the elements of a optimisation notification.

Definition at line 19 of file OptimisationNotificationStruct.hpp.

8.115.2 Constructor & Destructor Documentation

8.115.2.1 stdair::OptimisationNotificationStruct::OptimisationNotificationStruct (const AirportCode_T & *iOrigin*, const AirportCode_T & *iDestination*, const CityCode_T & *iPOS*, const Date_T & *iDepartureDate*, const DateTime_T & *iNotificationDateTime*, const CabinCode_T & *iPreferredCabin*, const NbOfSeats_T & *iPartySize*, const ChannelLabel_T & *iChannel*, const TripType_T & *iTripType*, const DayDuration_T & *iStayDuration*, const FrequentFlyer_T & *iFrequentFlyerType*, const Duration_T & *iPreferredDepartureTime*, const WTP_T & *iWTP*, const PriceValue_T & *iValueOfTime*)

Constructor.

Definition at line 39 of file OptimisationNotificationStruct.cpp.

8.115.2.2 stdair::OptimisationNotificationStruct::OptimisationNotificationStruct (const OptimisationNotificationStruct &)

Copy constructor.

Definition at line 20 of file OptimisationNotificationStruct.cpp.

8.115.2.3 stdair::OptimisationNotificationStruct::~OptimisationNotificationStruct ()

Destructor.

Definition at line 64 of file OptimisationNotificationStruct.cpp.

8.115.3 Member Function Documentation

8.115.3.1 const AirportCode_T& stdair::OptimisationNotificationStruct::getOrigin () const [inline]

Get the notificationed origin.

Definition at line 23 of file OptimisationNotificationStruct.hpp.

8.115.3.2 const AirportCode_T& stdair::OptimisationNotificationStruct::getDestination () const [inline]

Get the notificationed destination.

Definition at line 28 of file OptimisationNotificationStruct.hpp.

8.115.3.3 const CityCode_T& stdair::OptimisationNotificationStruct::getPOS () const [inline]

Get the point-of-sale.

Definition at line 33 of file OptimisationNotificationStruct.hpp.

8.115.3.4 const Date_T& stdair::OptimisationNotificationStruct::getPreferedDepartureDate () const [inline]

Get the notificationed departure date.

Definition at line 38 of file OptimisationNotificationStruct.hpp.

8.115.3.5 const DateTime_T& stdair::OptimisationNotificationStruct::getNotificationDateTime () const [inline]

Get the notification datetime.

Definition at line 43 of file OptimisationNotificationStruct.hpp.

8.115.3.6 const CabinCode_T& stdair::OptimisationNotificationStruct::getPreferredCabin () const [inline]

Get the preferred cabin.

Definition at line 48 of file OptimisationNotificationStruct.hpp.

8.115.3.7 const NbOfSeats_T& stdair::OptimisationNotificationStruct::getPartySize () const [inline]

Get the party size.

Definition at line 53 of file OptimisationNotificationStruct.hpp.

8.115.3.8 const ChannelLabel_T& stdair::OptimisationNotificationStruct::getOptimisationChannel () const [inline]

Get the reservation channel.

Definition at line 58 of file OptimisationNotificationStruct.hpp.

8.115.3.9 const TripType_T& stdair::OptimisationNotificationStruct::getTripType () const [inline]

Get the trip type.

Definition at line 63 of file OptimisationNotificationStruct.hpp.

8.115.3.10 const DayDuration_T& stdair::OptimisationNotificationStruct::getStayDuration () const [inline]

Get the duration of stay.

Definition at line 68 of file OptimisationNotificationStruct.hpp.

8.115.3.11 const FrequentFlyer_T& stdair::OptimisationNotificationStruct::getFrequentFlyerType () const [inline]

Get the frequent flyer type.

Definition at line 73 of file OptimisationNotificationStruct.hpp.

8.115.3.12 const Duration_T& stdair::OptimisationNotificationStruct::getPreferredDepartureTime () const [inline]

Get the preferred departure time.

Definition at line 78 of file OptimisationNotificationStruct.hpp.

8.115.3.13 const WTP_T& stdair::OptimisationNotificationStruct::getWTP () const [inline]

Get the willingness-to-pay.

Definition at line 83 of file OptimisationNotificationStruct.hpp.

8.115.3.14 const PriceValue_T& stdair::OptimisationNotificationStruct::getValueOfTime () const [inline]

Get the value of time.

Definition at line 88 of file OptimisationNotificationStruct.hpp.

8.115.3.15 void stdair::OptimisationNotificationStruct::toStream (std::ostream & *ioOut*) const

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::StructAbstract.

Definition at line 68 of file OptimisationNotificationStruct.cpp.

References describe().

8.115.3.16 void stdair::OptimisationNotificationStruct::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::StructAbstract.

Definition at line 73 of file OptimisationNotificationStruct.cpp.

8.115.3.17 const std::string stdair::OptimisationNotificationStruct::describe () const [virtual]

Display of the structure.

Implements stdair::StructAbstract.

Definition at line 77 of file OptimisationNotificationStruct.cpp.

Referenced by toStream().

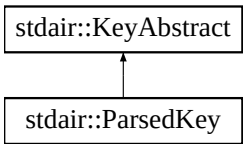
The documentation for this struct was generated from the following files:

- stdair/bom/OptimisationNotificationStruct.hpp
- stdair/bom/OptimisationNotificationStruct.cpp

8.116 stdair::ParsedKey Struct Reference

`#include <stdair/bom/ParsedKey.hpp>`

Inheritance diagram for stdair::ParsedKey::



Public Member Functions

- InventoryKey getInventoryKey () const
- FlightDateKey getFlightDateKey () const
- SegmentDateKey getSegmentKey () const
- LegDateKey getLegKey () const
- const Duration_T getBoardingTime () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const
- ParsedKey ()
- ~ParsedKey ()

Public Attributes

- std::string _fullKey
- std::string _airlineCode
- std::string _flightNumber
- std::string _departureDate
- std::string _boardingPoint
- std::string _offPoint
- std::string _boardingTime

8.116.1 Detailed Description

Structure which holds the results/keys after the parsing.

Definition at line 22 of file ParsedKey.hpp.

8.116.2 Constructor & Destructor Documentation

8.116.2.1 stdair::ParsedKey::ParsedKey ()

Definition at line 41 of file ParsedKey.cpp.

8.116.2.2 stdair::ParsedKey::~ParsedKey ()

Definition at line 47 of file ParsedKey.cpp.

8.116.3 Member Function Documentation

8.116.3.1 InventoryKey stdair::ParsedKey::getInventoryKey () const

Inventory key.

Definition at line 51 of file ParsedKey.cpp.

References _airlineCode, _fullKey, STDAIR_LOG_DEBUG, STDAIR_LOG_ERROR, and toString().

Referenced by stdair::BomKeyManager::extractInventoryKey().

8.116.3.2 FlightDateKey stdair::ParsedKey::getFlightDateKey () const

Flight-date key.

Definition at line 62 of file ParsedKey.cpp.

References _departureDate, _flightNumber, _fullKey, STDAIR_LOG_DEBUG, STDAIR_LOG_ERROR, stdair::TokeniserDashSeparator(), and toString().

Referenced by stdair::BomKeyManager::extractFlightDateKey(), and stdair::BomRetriever::retrieveSegmentDateFromLongKey().

8.116.3.3 SegmentDateKey stdair::ParsedKey::getSegmentKey () const

Segment-date key.

Definition at line 98 of file ParsedKey.cpp.

References _boardingPoint, _fullKey, _offPoint, STDAIR_LOG_DEBUG, STDAIR_LOG_ERROR, and toString().

Referenced by stdair::BomKeyManager::extractSegmentDateKey(), and stdair::BomRetriever::retrieveSegmentDateFromLongKey().

8.116.3.4 LegDateKey stdair::ParsedKey::getLegKey () const

Leg-date key.

Definition at line 84 of file ParsedKey.cpp.

References _boardingPoint, _fullKey, STDAIR_LOG_DEBUG, STDAIR_LOG_ERROR, and toString().

Referenced by stdair::BomKeyManager::extractLegDateKey().

8.116.3.5 const Duration_T stdair::ParsedKey::getBoardingTime () const

Boarding time.

Definition at line 112 of file ParsedKey.cpp.

References _boardingTime, _fullKey, STDAIR_LOG_DEBUG, STDAIR_LOG_ERROR, stdair::TokeniserTimeSeparator(), and toString().

8.116.3.6 void stdair::ParsedKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 130 of file ParsedKey.cpp.

References toString().

8.116.3.7 void stdair::ParsedKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 135 of file ParsedKey.cpp.

8.116.3.8 const std::string stdair::ParsedKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from stdair::KeyAbstract.

Definition at line 139 of file ParsedKey.cpp.

References _airlineCode, _boardingPoint, _boardingTime, _departureDate, _flightNumber, _offPoint, stdair::DEFAULT_KEY_FLD_DELIMITER, and stdair::DEFAULT_KEY_SUB_FLD_DELIMITER.

Referenced by stdair::TravelSolutionStruct::describe(), stdair::TravelSolutionStruct::describeSegmentPath(), stdair::TravelSolutionStruct::display(), getBoardingTime(), getFlightDateKey(), getInventoryKey(), getLegKey(), getSegmentKey(), and toStream().

8.116.4 Member Data Documentation

8.116.4.1 std::string stdair::ParsedKey::_fullKey

Definition at line 76 of file ParsedKey.hpp.

Referenced by stdair::BomKeyManager::extractKeys(), getBoardingTime(), getFlightDateKey(), getInventoryKey(), getLegKey(), and getSegmentKey().

8.116.4.2 std::string stdair::ParsedKey::_airlineCode

Definition at line 77 of file ParsedKey.hpp.

Referenced by stdair::BomKeyManager::extractKeys(), getInventoryKey(), stdair::BomRetriever::retrieveSegmentDateFromLongKey(), and toString().

8.116.4.3 std::string stdair::ParsedKey::_flightNumber

Definition at line 78 of file ParsedKey.hpp.

Referenced by stdair::BomKeyManager::extractKeys(), getFlightDateKey(), and toString().

8.116.4.4 std::string stdair::ParsedKey::_departureDate

Definition at line 79 of file ParsedKey.hpp.

Referenced by stdair::BomKeyManager::extractKeys(), getFlightDateKey(), and toString().

8.116.4.5 std::string stdair::ParsedKey::_boardingPoint

Definition at line 80 of file ParsedKey.hpp.

Referenced by stdair::BomKeyManager::extractKeys(), getLegKey(), getSegmentKey(), and toString().

8.116.4.6 std::string stdair::ParsedKey::_offPoint

Definition at line 81 of file ParsedKey.hpp.

Referenced by stdair::BomKeyManager::extractKeys(), getSegmentKey(), and toString().

8.116.4.7 std::string stdair::ParsedKey::_boardingTime

Definition at line 82 of file ParsedKey.hpp.

Referenced by stdair::BomKeyManager::extractKeys(), getBoardingTime(), and toString().

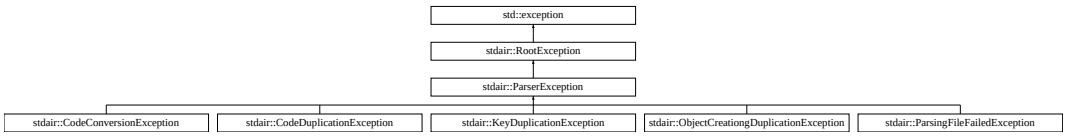
The documentation for this struct was generated from the following files:

- stdair/bom/ParsedKey.hpp
- stdair/bom/ParsedKey.cpp

8.117 stdair::ParserException Class Reference

`#include <stdair/stdair_exceptions.hpp>`

Inheritance diagram for stdair::ParserException::



Public Member Functions

- ParserException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

8.117.1 Detailed Description

Parser.

Definition at line 112 of file stdair_exceptions.hpp.

8.117.2 Constructor & Destructor Documentation

8.117.2.1 stdair::ParserException::ParserException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 115 of file stdair_exceptions.hpp.

8.117.3 Member Function Documentation

8.117.3.1 const char* stdair::RootException::what () const throw () [inline, inherited]

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

8.117.4 Member Data Documentation

8.117.4.1 std::string stdair::RootException::_what [protected, inherited]

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

Referenced by stdair::RootException::what().

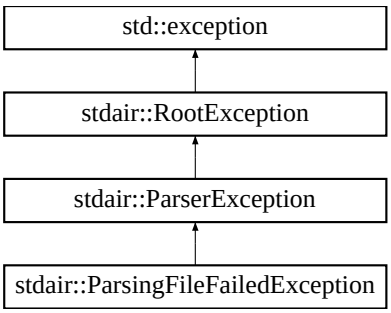
The documentation for this class was generated from the following file:

- stdair/stdair_exceptions.hpp

8.118 stdair::ParsingFileFailedException Class Reference

`#include <stdair/stdair_exceptions.hpp>`

Inheritance diagram for stdair::ParsingFileFailedException::



Public Member Functions

- ParsingFileFailedException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

8.118.1 Detailed Description

Input file parsing failure.

Definition at line 173 of file stdair_exceptions.hpp.

8.118.2 Constructor & Destructor Documentation

8.118.2.1 stdair::ParsingFileFailedException::ParsingFileFailedException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 176 of file stdair_exceptions.hpp.

8.118.3 Member Function Documentation

8.118.3.1 const char* stdair::RootException::what () const throw () [inline, inherited]

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

8.118.4 Member Data Documentation

8.118.4.1 std::string stdair::RootException::_what [protected, inherited]

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

Referenced by stdair::RootException::what().

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

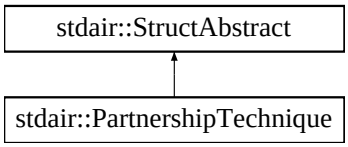
- stdair/stdair_exceptions.hpp

8.119 stdair::PartnershipTechnique Struct Reference

Enumeration of partnership techniques.

`#include <stdair/basic/PartnershipTechnique.hpp>`

Inheritance diagram for stdair::PartnershipTechnique::



Public Types

- NONE = 0
- RAE_DA
- RAE_YP
- IBP_DA
- IBP_YP
- IBP_YP_U
- RMC
- A_RMC
- LAST_VALUE
- enum EN_PartnershipTechnique {
  NONE = 0, RAE_DA, RAE_YP, IBP_DA,
  IBP_YP, IBP_YP_U, RMC, A_RMC,
  LAST_VALUE }

Public Member Functions

- EN_PartnershipTechnique getTechnique () const
- char getTechniqueAsChar () const
- std::string getTechniqueAsString () const
- const std::string describe () const
- bool operator== (const EN_PartnershipTechnique &) const
- PartnershipTechnique (const EN_PartnershipTechnique &)
- PartnershipTechnique (const char iTechnique)
- PartnershipTechnique (const std::string &iTechnique)
- PartnershipTechnique (const PartnershipTechnique &)
- void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

Static Public Member Functions

- static const std::string & getLabel (const EN_PartnershipTechnique &)
- static EN_PartnershipTechnique getTechnique (const char)
- static char getTechniqueLabel (const EN_PartnershipTechnique &)
- static std::string getTechniqueLabelAsString (const EN_PartnershipTechnique &)
- static std::string describeLabels ()

8.119.1 Detailed Description

Enumeration of partnership techniques.

Definition at line 17 of file PartnershipTechnique.hpp.

8.119.2 Member Enumeration Documentation

8.119.2.1 enum stdair::PartnershipTechnique::EN_PartnershipTechnique

Enumerator:

NONE

RAE_DA

RAE_YP

IBP_DA

IBP_YP

IBP_YP_U

RMC

A_RMC

LAST_VALUE

Definition at line 19 of file PartnershipTechnique.hpp.

8.119.3 Constructor & Destructor Documentation

8.119.3.1 stdair::PartnershipTechnique::PartnershipTechnique (const EN_PartnershipTechnique &)

Main constructor.

Definition at line 48 of file PartnershipTechnique.cpp.

8.119.3.2 stdair::PartnershipTechnique::PartnershipTechnique (const char *iTechnique*)

Alternative constructor.

Definition at line 82 of file PartnershipTechnique.cpp.

8.119.3.3 stdair::PartnershipTechnique::PartnershipTechnique (const std::string & *iTechnique*)

Alternative constructor.

Definition at line 88 of file PartnershipTechnique.cpp.

References getTechnique().

8.119.3.4 stdair::PartnershipTechnique::PartnershipTechnique (const PartnershipTechnique &)

Default copy constructor.

Definition at line 42 of file PartnershipTechnique.cpp.

8.119.4 Member Function Documentation

8.119.4.1 const std::string & stdair::PartnershipTechnique::getLabel (const EN_PartnershipTechnique &) [static]

Get the label as a string (e.g., "RevenueManagementCooperation").

Definition at line 98 of file PartnershipTechnique.cpp.

8.119.4.2 PartnershipTechnique::EN_PartnershipTechnique stdair::PartnershipTechnique::getTechnique (const *char*) [static]

Get the technique value from parsing a single char (e.g., 'r' or 'C').

Definition at line 54 of file PartnershipTechnique.cpp.

References A_RMC, describeLabels(), IBP_DA, IBP_YP, IBP_YP_U, LAST_VALUE, NONE, RAE_DA, RAE_YP, and RMC.

Referenced by stdair::AirlineFeature::getPartnershipTechnique().

8.119.4.3 char stdair::PartnershipTechnique::getTechniqueLabel (const EN_PartnershipTechnique &) [static]

Get the label as a single char (e.g., 'r' or 'C').

Definition at line 104 of file PartnershipTechnique.cpp.

8.119.4.4 std::string stdair::PartnershipTechnique::getTechniqueLabelAsString (const EN_PartnershipTechnique &) [static]

Get the label as a string of a single char (e.g., "r" or "C").

Definition at line 110 of file PartnershipTechnique.cpp.

8.119.4.5 std::string stdair::PartnershipTechnique::describeLabels () [static]

List the labels.

Definition at line 117 of file PartnershipTechnique.cpp.

References LAST_VALUE.

Referenced by getTechnique().

8.119.4.6 PartnershipTechnique::EN_PartnershipTechnique stdair::PartnershipTechnique::getTechnique () const

Get the enumerated value.

Definition at line 130 of file PartnershipTechnique.cpp.

Referenced by PartnershipTechnique().

8.119.4.7 char stdair::PartnershipTechnique::getTechniqueAsChar () const

Get the enumerated value as a char (e.g., 'r' or 'C').

Definition at line 135 of file PartnershipTechnique.cpp.

8.119.4.8 std::string stdair::PartnershipTechnique::getTechniqueAsString () const

Get the enumerated value as a short string (e.g., "r" or "C").

Definition at line 141 of file PartnershipTechnique.cpp.

8.119.4.9 const std::string stdair::PartnershipTechnique::describe () const [virtual]

Give a description of the structure (e.g., "RevenueManagementCooperation" or "InterlineBidPriceYieldProration").

Implements stdair::StructAbstract.

Definition at line 148 of file PartnershipTechnique.cpp.

8.119.4.10 bool stdair::PartnershipTechnique::operator== (const EN_PartnershipTechnique &) const

Comparison operator.

Definition at line 156 of file PartnershipTechnique.cpp.

8.119.4.11 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const [inline, inherited]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References stdair::StructAbstract::describe().

8.119.4.12 virtual void stdair::StructAbstract::fromStream (std::istream & *ioIn*) [inline, virtual, inherited]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented in stdair::ProgressStatusSet, stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::EventStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 38 of file StructAbstract.hpp.

Referenced by operator>>().

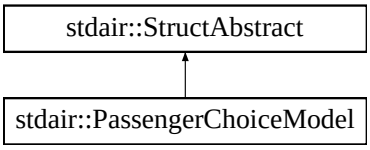
The documentation for this struct was generated from the following files:

- stdair/basic/PartnershipTechnique.hpp
- stdair/basic/PartnershipTechnique.cpp

8.120 stdair::PassengerChoiceModel Struct Reference

`#include <stdair/basic/PassengerChoiceModel.hpp>`

Inheritance diagram for stdair::PassengerChoiceModel::



Public Types

- HARD_RESTRICTION = 0
- PRICE_ORIENTED
- HYBRID

- LAST_VALUE
- enum EN_PassengerChoiceModel { HARD_RESTRICTION = 0, PRICE_ORIENTED, HYBRID, LAST_VALUE }

Public Member Functions

- EN_PassengerChoiceModel getModel () const
- std::string getModelAsString () const
- const std::string describe () const
- bool operator== (const EN_PassengerChoiceModel &) const
- PassengerChoiceModel (const EN_PassengerChoiceModel &)
- PassengerChoiceModel (const char iModel)
- PassengerChoiceModel (const PassengerChoiceModel &)
- void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

Static Public Member Functions

- static const std::string & getLabel (const EN_PassengerChoiceModel &)
- static char getModelLabel (const EN_PassengerChoiceModel &)
- static std::string getModelLabelAsString (const EN_PassengerChoiceModel &)
- static std::string describeLabels ()

8.120.1 Detailed Description

Enumeration of passenger choice models.

Definition at line 15 of file PassengerChoiceModel.hpp.

8.120.2 Member Enumeration Documentation

8.120.2.1 enum stdair::PassengerChoiceModel::EN_PassengerChoiceModel

Enumerator:

HARD_RESTRICTION

PRICE_ORIENTED

HYBRID

LAST_VALUE

Definition at line 17 of file PassengerChoiceModel.hpp.

8.120.3 Constructor & Destructor Documentation

8.120.3.1 stdair::PassengerChoiceModel::PassengerChoiceModel (const EN_PassengerChoiceModel &)

Constructor.

Definition at line 36 of file PassengerChoiceModel.cpp.

8.120.3.2 stdair::PassengerChoiceModel::PassengerChoiceModel (const char *iModel*)

Constructor.

Definition at line 41 of file PassengerChoiceModel.cpp.

References describeLabels(), HARD_RESTRICTION, HYBRID, LAST_VALUE, and PRICE_-ORIENTED.

8.120.3.3 stdair::PassengerChoiceModel::PassengerChoiceModel (const PassengerChoiceModel &)

Default copy constructor.

Definition at line 30 of file PassengerChoiceModel.cpp.

8.120.4 Member Function Documentation

8.120.4.1 const std::string & stdair::PassengerChoiceModel::getLabel (const EN_Passenger-ChoiceModel &) [static]

Get the label as a string (e.g., HardRestrictionModel", "PriceOrientedModel" or "HybridModel").

Definition at line 60 of file PassengerChoiceModel.cpp.

8.120.4.2 char stdair::PassengerChoiceModel::getModelLabel (const EN_PassengerChoiceModel &) [static]

Get the label as a single char (e.g., 'R', 'P' or 'H').

Definition at line 65 of file PassengerChoiceModel.cpp.

8.120.4.3 std::string stdair::PassengerChoiceModel::getModelLabelAsString (const EN_-PassengerChoiceModel &) [static]

Get the label as a string of a single char (e.g., "R", "P" or "H").

Definition at line 71 of file PassengerChoiceModel.cpp.

8.120.4.4 std::string stdair::PassengerChoiceModel::describeLabels () [static]

List the labels.

Definition at line 78 of file PassengerChoiceModel.cpp.

References LAST_VALUE.

Referenced by PassengerChoiceModel().

8.120.4.5 PassengerChoiceModel::EN_PassengerChoiceModel stdair::PassengerChoice-Model::getModel () const

Get the enumerated value.

Definition at line 90 of file PassengerChoiceModel.cpp.

8.120.4.6 std::string stdair::PassengerChoiceModel::getModelAsString () const

Get the enumerated value as a short string (e.g., "R", "P" or "H").

Definition at line 95 of file PassengerChoiceModel.cpp.

8.120.4.7 const std::string stdair::PassengerChoiceModel::describe () const [virtual]

Give a description of the structure (e.g., HardRestrictionModel", "PriceOrientedModel" or "Hybrid-Model").

Implements stdair::StructAbstract.

Definition at line 102 of file PassengerChoiceModel.cpp.

8.120.4.8 bool stdair::PassengerChoiceModel::operator== (const EN_PassengerChoiceModel &) const

Comparaison operator.

Definition at line 110 of file PassengerChoiceModel.cpp.

8.120.4.9 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const [inline, inherited]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References stdair::StructAbstract::describe().

8.120.4.10 virtual void stdair::StructAbstract::fromStream (std::istream & *ioIn*) [inline, virtual, inherited]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented in stdair::ProgressStatusSet, stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::EventStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 38 of file StructAbstract.hpp.

Referenced by operator>>().

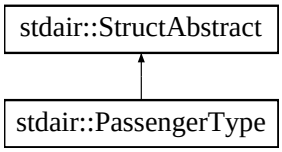
The documentation for this struct was generated from the following files:

- stdair/basic/PassengerChoiceModel.hpp
- stdair/basic/PassengerChoiceModel.cpp

8.121 stdair::PassengerType Struct Reference

`#include <stdair/basic/PassengerType.hpp>`

Inheritance diagram for stdair::PassengerType::



Public Types

- LEISURE = 0
- BUSINESS
- FIRST
- LAST_VALUE
- enum EN_PassengerType { LEISURE = 0, BUSINESS, FIRST, LAST_VALUE }

Public Member Functions

- EN_PassengerType getType () const
- std::string getTypeAsString () const
- const std::string describe () const
- bool operator== (const EN_PassengerType &) const
- PassengerType (const EN_PassengerType &)
- PassengerType (const char iType)
- void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

Static Public Member Functions

- static const std::string & getLabel (const EN_PassengerType &)
- static char getTypeLabel (const EN_PassengerType &)
- static std::string getTypeLabelAsString (const EN_PassengerType &)
- static std::string describeLabels ()

8.121.1 Detailed Description

Enumeration of Frequent Flyer types.

Definition at line 15 of file PassengerType.hpp.

8.121.2 Member Enumeration Documentation

8.121.2.1 enum stdair::PassengerType::EN_PassengerType

Enumerator:

LEISURE
BUSINESS
FIRST
LAST_VALUE

Definition at line 17 of file PassengerType.hpp.

8.121.3 Constructor & Destructor Documentation

8.121.3.1 stdair::PassengerType::PassengerType (const EN_PassengerType &)

Constructor.

Definition at line 21 of file PassengerType.cpp.

8.121.3.2 stdair::PassengerType::PassengerType (const char *iType*)

Constructor.

Definition at line 26 of file PassengerType.cpp.

References BUSINESS, describeLabels(), FIRST, LAST_VALUE, and LEISURE.

8.121.4 Member Function Documentation

8.121.4.1 const std::string & stdair::PassengerType::getLabel (const EN_PassengerType &) [static]

Get the label as a string (e.g., "Leisure" or "Business").

Definition at line 44 of file PassengerType.cpp.

8.121.4.2 char stdair::PassengerType::getTypeLabel (const EN_PassengerType &) [static]

Get the label as a single char (e.g., 'L' or 'B').

Definition at line 49 of file PassengerType.cpp.

8.121.4.3 std::string stdair::PassengerType::getTypeLabelAsString (const EN_PassengerType &) [static]

Get the label as a single char (e.g., 'L' or 'B').

Definition at line 55 of file PassengerType.cpp.

8.121.4.4 std::string stdair::PassengerType::describeLabels () [static]

List the labels.

Definition at line 62 of file PassengerType.cpp.

References LAST_VALUE.

Referenced by PassengerType().

8.121.4.5 PassengerType::EN_PassengerType stdair::PassengerType::getType () const

Get the enumerated value.

Definition at line 74 of file PassengerType.cpp.

8.121.4.6 std::string stdair::PassengerType::getTypeAsString () const

Get the enumerated value as a short string (e.g., 'L' or 'B').

Definition at line 79 of file PassengerType.cpp.

8.121.4.7 const std::string stdair::PassengerType::describe () const [virtual]

Give a description of the structure (e.g., "Leisure" or "Business").

Implements stdair::StructAbstract.

Definition at line 86 of file PassengerType.cpp.

8.121.4.8 bool stdair::PassengerType::operator== (const EN_PassengerType &) const

Comparison operator.

Definition at line 93 of file PassengerType.cpp.

8.121.4.9 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const [inline, inherited]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References stdair::StructAbstract::describe().

8.121.4.10 virtual void stdair::StructAbstract::fromStream (std::istream & *ioIn*) [inline, virtual, inherited]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented in stdair::ProgressStatusSet, stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::EventStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 38 of file StructAbstract.hpp.

Referenced by operator>>().

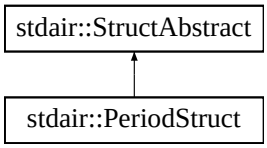
The documentation for this struct was generated from the following files:

- stdair/basic/PassengerType.hpp
- stdair/basic/PassengerType.cpp

8.122 stdair::PeriodStruct Struct Reference

`#include <stdair/bom/PeriodStruct.hpp>`

Inheritance diagram for stdair::PeriodStruct::



Public Member Functions

- const DatePeriod_T & getDateRange () const
- const DoWStruct & getDoW () const
- void setDateRange (const DatePeriod_T &iDateRange)
- void setDoW (const DoWStruct &iDoW)
- const std::string describe () const
- const std::string describeShort () const
- PeriodStruct addDateOffset (const DateOffset_T &) const
- PeriodStruct intersection (const PeriodStruct &) const
- const bool isValid () const
- PeriodStruct (const DatePeriod_T &, const DoWStruct &)
- PeriodStruct ()
- PeriodStruct (const PeriodStruct &)
- ~PeriodStruct ()
- void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

8.122.1 Detailed Description

Define a departure period

A period is defined by a date range and a day-of-week struct.

Definition at line 19 of file PeriodStruct.hpp.

8.122.2 Constructor & Destructor Documentation

8.122.2.1 stdair::PeriodStruct::PeriodStruct (const DatePeriod_T &, const DoWStruct &)

Constructor.

Definition at line 19 of file PeriodStruct.cpp.

8.122.2.2 stdair::PeriodStruct::PeriodStruct ()

Default constructors.

Definition at line 14 of file PeriodStruct.cpp.

Referenced by addDateOffset(), and intersection().

8.122.2.3 stdair::PeriodStruct::PeriodStruct (const PeriodStruct &)

Definition at line 25 of file PeriodStruct.cpp.

8.122.2.4 stdair::PeriodStruct::~PeriodStruct () [inline]

Default destructor.

Definition at line 64 of file PeriodStruct.hpp.

8.122.3 Member Function Documentation

8.122.3.1 const DatePeriod_T& stdair::PeriodStruct::getDateRange () const [inline]

Retrieve the attributes.

Definition at line 23 of file PeriodStruct.hpp.

Referenced by addDateOffset().

8.122.3.2 const DoWStruct& stdair::PeriodStruct::getDoW () const [inline]

Definition at line 26 of file PeriodStruct.hpp.

Referenced by addDateOffset().

8.122.3.3 void stdair::PeriodStruct::setDateRange (const DatePeriod_T & *iDateRange*) [inline]

Set the new value for the attributes.

Definition at line 33 of file PeriodStruct.hpp.

8.122.3.4 void stdair::PeriodStruct::setDoW (const DoWStruct & *iDoW*) [inline]

Definition at line 36 of file PeriodStruct.hpp.

8.122.3.5 const std::string stdair::PeriodStruct::describe () const [virtual]

Display explicitly (e.g., "Mon.Tue.Wed.Thu.Fri.").

Implements stdair::StructAbstract.

Definition at line 38 of file PeriodStruct.cpp.

References stdair::DoWStruct::describe().

8.122.3.6 const std::string stdair::PeriodStruct::describeShort () const

Display as a bit set (e.g., "1111100").

Definition at line 31 of file PeriodStruct.cpp.

References stdair::DoWStruct::describeShort().

Referenced by stdair::FlightPeriodKey::toString().

8.122.3.7 PeriodStruct stdair::PeriodStruct::addDateOffset (const DateOffset_T &) const

Build a period struct from this period struct by adding a date offset.

Definition at line 46 of file PeriodStruct.cpp.

References getDateRange(), getDoW(), PeriodStruct(), and stdair::DoWStruct::shift().

8.122.3.8 PeriodStruct stdair::PeriodStruct::intersection (const PeriodStruct &) const

Build a new period struct which is the intersection of two period structs.

Definition at line 63 of file PeriodStruct.cpp.

References _dateRange, _dow, stdair::DoWStruct::intersection(), and PeriodStruct().

8.122.3.9 const bool stdair::PeriodStruct::isValid () const

Return if the period is valid (i.e., valid date range and valid DoW).

Definition at line 72 of file PeriodStruct.cpp.

References stdair::DoWStruct::isValid().

8.122.3.10 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const [inline, inherited]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References stdair::StructAbstract::describe().

8.122.3.11 virtual void stdair::StructAbstract::fromStream (std::istream & *ioIn*) [inline, virtual, inherited]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented in stdair::ProgressStatusSet, stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::EventStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 38 of file StructAbstract.hpp.

Referenced by operator>>().

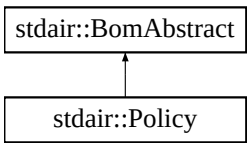
The documentation for this struct was generated from the following files:

- stdair/bom/PeriodStruct.hpp
- stdair/bom/PeriodStruct.cpp

8.123 stdair::Policy Class Reference

`#include <stdair/bom/Policy.hpp>`

Inheritance diagram for stdair::Policy::



Public Types

- typedef PolicyKey Key_T

Public Member Functions

- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const HolderMap_T & getHolderMap () const
- const BookingClassList_T & getBookingClassList () const
- const NbOfBookings_T & getDemand () const
- const StdDevValue_T & getStdDev () const
- const Yield_T & getYield () const
- const Revenue_T getTotalRevenue () const
- void setDemand (const NbOfBookings_T &iDemand)
- void setStdDev (const StdDevValue_T &iStdDev)
- void setYield (const Yield_T &iYield)
- void resetDemandForecast ()
- void addYieldDemand (const Yield_T &, const NbOfBookings_T &)

- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- template<class Archive> void serialize (Archive &ar, const unsigned int iFileVersion)

Protected Member Functions

- Policy (const Key_T &)
- virtual ~Policy ()

Friends

- class FacBom
- class FacBomManager
- class boost::serialization::access

8.123.1 Detailed Description

Structure holding the elements of a policy. A policy is a set of booking classes, each booking class belongs to a different Fare Family.

Definition at line 30 of file Policy.hpp.

8.123.2 Member Typedef Documentation

8.123.2.1 typedef PolicyKey stdair::Policy::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 40 of file Policy.hpp.

8.123.3 Constructor & Destructor Documentation

8.123.3.1 stdair::Policy::Policy (const Key_T &) [protected]

Main constructor.

Definition at line 31 of file Policy.cpp.

8.123.3.2 stdair::Policy::~Policy () [protected, virtual]

Destructor.

Definition at line 35 of file Policy.cpp.

8.123.4 Member Function Documentation

8.123.4.1 const Key_T& stdair::Policy::getKey () const [inline]

Get the policy key.

Definition at line 45 of file Policy.hpp.

8.123.4.2 BomAbstract* const stdair::Policy::getParent () const [inline]

Get the parent object.

Definition at line 50 of file Policy.hpp.

8.123.4.3 const HolderMap_T& stdair::Policy::getHolderMap () const [inline]

Get the map of children holders.

Definition at line 57 of file Policy.hpp.

8.123.4.4 const BookingClassList_T & stdair::Policy::getBookingClassList () const

Getter for the booking classes.

Definition at line 52 of file Policy.cpp.

8.123.4.5 const NbOfBookings_T& stdair::Policy::getDemand () const [inline]

Getter for the demand.

Definition at line 65 of file Policy.hpp.

8.123.4.6 const StdDevValue_T& stdair::Policy::getStdDev () const [inline]

Getter for the standard deviation demand.

Definition at line 70 of file Policy.hpp.

8.123.4.7 const Yield_T& stdair::Policy::getYield () const [inline]

Getter for the yield.

Definition at line 75 of file Policy.hpp.

8.123.4.8 const Revenue_T stdair::Policy::getTotalRevenue () const

Get the total revenue of the policy.

Definition at line 57 of file Policy.cpp.

8.123.4.9 void stdair::Policy::setDemand (const NbOfBookings_T & *iDemand*) [inline]

Setter for the unconstraining demand.

Definition at line 85 of file Policy.hpp.

8.123.4.10 void stdair::Policy::setStdDev (const StdDevValue_T & *iStdDev*) [inline]

Setter for standard deviation demand.

Definition at line 90 of file Policy.hpp.

8.123.4.11 void stdair::Policy::setYield (const Yield_T & *iYield*) [inline]

Setter for the yield.

Definition at line 95 of file Policy.hpp.

8.123.4.12 void stdair::Policy::resetDemandForecast () [inline]

Reset demand forecast.

Definition at line 100 of file Policy.hpp.

8.123.4.13 void stdair::Policy::addYieldDemand (const Yield_T &, const NbOfBookings_T &)

Add the new pair (yield, demand) to the map.

Definition at line 70 of file Policy.cpp.

8.123.4.14 void stdair::Policy::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 116 of file Policy.hpp.

References toString().

8.123.4.15 void stdair::Policy::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 125 of file Policy.hpp.

8.123.4.16 std::string stdair::Policy::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 39 of file Policy.cpp.

References describeKey().

Referenced by toStream().

8.123.4.17 const std::string stdair::Policy::describeKey () const [inline]

Get a string describing the key.

Definition at line 136 of file Policy.hpp.

References stdair::PolicyKey::toString().

Referenced by toString().

8.123.4.18 template<class Archive> void stdair::Policy::serialize (Archive & *ar*, const unsigned int *iFileVersion*)

Serialisation.

8.123.5 Friends And Related Function Documentation

8.123.5.1 friend class FacBom [friend]

Definition at line 31 of file Policy.hpp.

8.123.5.2 friend class FacBomManager [friend]

Definition at line 32 of file Policy.hpp.

8.123.5.3 friend class boost::serialization::access [friend]

Definition at line 33 of file Policy.hpp.

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

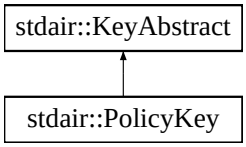
- stdair/bom/Policy.hpp
- stdair/bom/Policy.cpp

8.124 stdair::PolicyKey Struct Reference

Key of a given policy, made of a policy code.

`#include <stdair/bom/PolicyKey.hpp>`

Inheritance diagram for stdair::PolicyKey::



Public Member Functions

- PolicyKey (const PolicyCode_T &iPolicyCode)
- PolicyKey (const PolicyKey &)
- ~PolicyKey ()
- const PolicyCode_T & getPolicyCode () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const
- template<class Archive> void serialize (Archive &ar, const unsigned int iFileVersion)

Friends

- class boost::serialization::access

8.124.1 Detailed Description

Key of a given policy, made of a policy code.

Definition at line 26 of file PolicyKey.hpp.

8.124.2 Constructor & Destructor Documentation

8.124.2.1 stdair::PolicyKey::PolicyKey (const PolicyCode_T & *iPolicyCode*)

Constructor.

Definition at line 28 of file PolicyKey.cpp.

8.124.2.2 stdair::PolicyKey::PolicyKey (const PolicyKey &)

Copy constructor.

Definition at line 23 of file PolicyKey.cpp.

8.124.2.3 stdair::PolicyKey::~PolicyKey ()

Destructor.

Definition at line 33 of file PolicyKey.cpp.

8.124.3 Member Function Documentation

8.124.3.1 const PolicyCode_T& stdair::PolicyKey::getPolicyCode () const [inline]

Get the policy code.

Definition at line 56 of file PolicyKey.hpp.

8.124.3.2 void stdair::PolicyKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 37 of file PolicyKey.cpp.

References toString().

8.124.3.3 void stdair::PolicyKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 42 of file PolicyKey.cpp.

8.124.3.4 const std::string stdair::PolicyKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from stdair::KeyAbstract.

Definition at line 46 of file PolicyKey.cpp.

Referenced by stdair::Policy::describeKey(), and toStream().

8.124.3.5 template<class Archive> void stdair::PolicyKey::serialize (Archive & *ar*, const unsigned int *iFileVersion*)

Serialisation.

Definition at line 68 of file PolicyKey.cpp.

8.124.4 Friends And Related Function Documentation

8.124.4.1 friend class boost::serialization::access [friend]

Definition at line 27 of file PolicyKey.hpp.

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

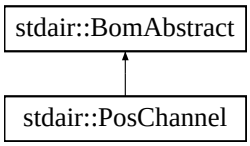
- stdair/bom/PolicyKey.hpp
- stdair/bom/PolicyKey.cpp

8.125 stdair::PosChannel Class Reference

Class representing the actual attributes for a fare point of sale.

`#include <stdair/bom/PosChannel.hpp>`

Inheritance diagram for stdair::PosChannel::



Public Types

- typedef PosChannelKey Key_T

Public Member Functions

- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)

- std::string toString () const
- const std::string describeKey () const
- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const stdair::HolderMap_T & getHolderMap () const
- const CityCode_T & getPos () const
- const ChannelLabel_T & getChannel () const

Protected Member Functions

- PosChannel (const Key_T &)
- virtual ~PosChannel ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent
- HolderMap_T _holderMap

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager

8.125.1 Detailed Description

Class representing the actual attributes for a fare point of sale.

Definition at line 19 of file PosChannel.hpp.

8.125.2 Member Typedef Documentation

8.125.2.1 typedef PosChannelKey stdair::PosChannel::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 29 of file PosChannel.hpp.

8.125.3 Constructor & Destructor Documentation

8.125.3.1 stdair::PosChannel::PosChannel (const Key_T &) [protected]

Main constructor.

Definition at line 28 of file PosChannel.cpp.

8.125.3.2 stdair::PosChannel::~PosChannel () [protected, virtual]

Destructor.

Definition at line 33 of file PosChannel.cpp.

8.125.4 Member Function Documentation

8.125.4.1 void stdair::PosChannel::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 38 of file PosChannel.hpp.

References toString().

8.125.4.2 void stdair::PosChannel::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 47 of file PosChannel.hpp.

8.125.4.3 std::string stdair::PosChannel::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 37 of file PosChannel.cpp.

References describeKey().

Referenced by toStream().

8.125.4.4 const std::string stdair::PosChannel::describeKey () const [inline]

Get a string describing the key.

Definition at line 58 of file PosChannel.hpp.

References _key, and stdair::PosChannelKey::toString().

Referenced by toString().

8.125.4.5 const Key_T& stdair::PosChannel::getKey () const [inline]

Get the primary key (pos, channel).

Definition at line 67 of file PosChannel.hpp.

References _key.

8.125.4.6 BomAbstract* const stdair::PosChannel::getParent () const [inline]

Get a reference on the parent object instance.

Definition at line 74 of file PosChannel.hpp.

References _parent.

8.125.4.7 const stdair::HolderMap_T& stdair::PosChannel::getHolderMap () const [inline]

Get a reference on the children holder.

Definition at line 81 of file PosChannel.hpp.

References _holderMap.

8.125.4.8 const CityCode_T& stdair::PosChannel::getPos () const [inline]

Get the point-of-sale.

Definition at line 88 of file PosChannel.hpp.

References _key, and stdair::PosChannelKey::getPos().

8.125.4.9 const ChannelLabel_T& stdair::PosChannel::getChannel () const [inline]

Get the channel.

Definition at line 95 of file PosChannel.hpp.

References _key, and stdair::PosChannelKey::getChannel().

8.125.5 Friends And Related Function Documentation

8.125.5.1 friend class FacBom [friend]

Definition at line 20 of file PosChannel.hpp.

8.125.5.2 friend class FacCloneBom [friend]

Definition at line 21 of file PosChannel.hpp.

8.125.5.3 friend class FacBomManager [friend]

Definition at line 22 of file PosChannel.hpp.

8.125.6 Member Data Documentation

8.125.6.1 Key_T stdair::PosChannel::_key [protected]

Primary key (flight number and departure date).

Definition at line 127 of file PosChannel.hpp.

Referenced by describeKey(), getChannel(), getKey(), and getPos().

8.125.6.2 BomAbstract* stdair::PosChannel::_parent [protected]

Pointer on the parent class.

Definition at line 132 of file PosChannel.hpp.

Referenced by getParent().

8.125.6.3 HolderMap_T stdair::PosChannel::_holderMap [protected]

Map holding the children.

Definition at line 137 of file PosChannel.hpp.

Referenced by getHolderMap().

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

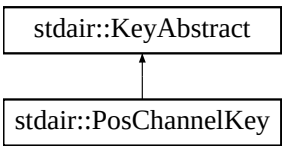
- stdair/bom/PosChannel.hpp
- stdair/bom/PosChannel.cpp

8.126 stdair::PosChannelKey Struct Reference

Key of point of sale and channel.

`#include <stdair/bom/PosChannelKey.hpp>`

Inheritance diagram for stdair::PosChannelKey::



Public Member Functions

- PosChannelKey (const stdair::CityCode_T &, const stdair::ChannelLabel_T &)
- PosChannelKey (const PosChannelKey &)
- ~PosChannelKey ()
- const stdair::CityCode_T & getPos () const
- const stdair::ChannelLabel_T & getChannel () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const

8.126.1 Detailed Description

Key of point of sale and channel.

Definition at line 15 of file PosChannelKey.hpp.

8.126.2 Constructor & Destructor Documentation

8.126.2.1 stdair::PosChannelKey::PosChannelKey (const stdair::CityCode_T &, const stdair::ChannelLabel_T &)

Main constructor.

Definition at line 22 of file PosChannelKey.cpp.

8.126.2.2 stdair::PosChannelKey::PosChannelKey (const PosChannelKey &)

Copy constructor.

Definition at line 28 of file PosChannelKey.cpp.

8.126.2.3 stdair::PosChannelKey::~PosChannelKey ()

Destructor.

Definition at line 33 of file PosChannelKey.cpp.

8.126.3 Member Function Documentation

8.126.3.1 const stdair::CityCode_T& stdair::PosChannelKey::getPos () const [inline]

Get the point of sale.

Definition at line 43 of file PosChannelKey.hpp.

Referenced by stdair::PosChannel::getPos().

8.126.3.2 const stdair::ChannelLabel_T& stdair::PosChannelKey::getChannel () const [inline]

Get the channel.

Definition at line 50 of file PosChannelKey.hpp.

Referenced by stdair::PosChannel::getChannel().

8.126.3.3 void stdair::PosChannelKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream&	the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 37 of file PosChannelKey.cpp.

References toString().

8.126.3.4 void stdair::PosChannelKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 42 of file PosChannelKey.cpp.

8.126.3.5 const std::string stdair::PosChannelKey::toString () const [virtual]

Get the serialised version of the Business Object Key. That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

Reimplemented from stdair::KeyAbstract.

Definition at line 46 of file PosChannelKey.cpp.

References stdair::DEFAULT_KEY_SUB_FLD_DELIMITER.

Referenced by stdair::PosChannel::describeKey(), and toStream().

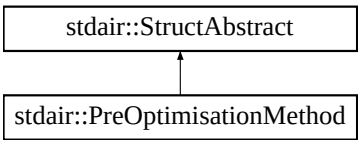
The documentation for this struct was generated from the following files:

- stdair/bom/PosChannelKey.hpp
- stdair/bom/PosChannelKey.cpp

8.127 stdair::PreOptimisationMethod Struct Reference

`#include <stdair/basic/PreOptimisationMethod.hpp>`

Inheritance diagram for stdair::PreOptimisationMethod::



Public Types

- NONE = 0
- FA
- MRT
- LAST_VALUE
- enum EN_PreOptimisationMethod { NONE = 0, FA, MRT, LAST_VALUE }

Public Member Functions

- EN_PreOptimisationMethod getMethod () const
- std::string getMethodAsString () const
- const std::string describe () const
- bool operator== (const EN_PreOptimisationMethod &) const
- PreOptimisationMethod (const EN_PreOptimisationMethod &)
- PreOptimisationMethod (const char iMethod)

- PreOptimisationMethod (const PreOptimisationMethod &)
- void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

Static Public Member Functions

- static const std::string & getLabel (const EN_PreOptimisationMethod &)
- static char getMethodLabel (const EN_PreOptimisationMethod &)
- static std::string getMethodLabelAsString (const EN_PreOptimisationMethod &)
- static std::string describeLabels ()

8.127.1 Detailed Description

Enumeration of PreOptimisation methods.

Definition at line 15 of file PreOptimisationMethod.hpp.

8.127.2 Member Enumeration Documentation

8.127.2.1 enum stdair::PreOptimisationMethod::EN_PreOptimisationMethod

Enumerator:

NONE

FA

MRT

LAST_VALUE

Definition at line 17 of file PreOptimisationMethod.hpp.

8.127.3 Constructor & Destructor Documentation

8.127.3.1 stdair::PreOptimisationMethod::PreOptimisationMethod (const EN_PreOptimisationMethod &)

Constructor.

Definition at line 36 of file PreOptimisationMethod.cpp.

8.127.3.2 stdair::PreOptimisationMethod::PreOptimisationMethod (const char *iMethod*)

Constructor.

Definition at line 41 of file PreOptimisationMethod.cpp.

References describeLabels(), FA, LAST_VALUE, MRT, and NONE.

8.127.3.3 stdair::PreOptimisationMethod::PreOptimisationMethod (const PreOptimisationMethod &)

Default copy constructor.

Definition at line 30 of file PreOptimisationMethod.cpp.

8.127.4 Member Function Documentation

8.127.4.1 const std::string & stdair::PreOptimisationMethod::getLabel (const EN_PreOptimisationMethod &) [static]

Get the label as a string (e.g., MRT or FA).

Definition at line 60 of file PreOptimisationMethod.cpp.

8.127.4.2 char stdair::PreOptimisationMethod::getMethodLabel (const EN_PreOptimisationMethod &) [static]

Get the label as a single char (e.g., 'M' or 'E').

Definition at line 65 of file PreOptimisationMethod.cpp.

8.127.4.3 std::string stdair::PreOptimisationMethod::getMethodLabelAsString (const EN_PreOptimisationMethod &) [static]

Get the label as a string of a single char (e.g., "M" or "E").

Definition at line 71 of file PreOptimisationMethod.cpp.

8.127.4.4 std::string stdair::PreOptimisationMethod::describeLabels () [static]

List the labels.

Definition at line 78 of file PreOptimisationMethod.cpp.

References LAST_VALUE.

Referenced by PreOptimisationMethod().

8.127.4.5 PreOptimisationMethod::EN_PreOptimisationMethod stdair::PreOptimisationMethod::getMethod () const

Get the enumerated value.

Definition at line 90 of file PreOptimisationMethod.cpp.

Referenced by stdair::AirlineFeature::getPreOptimisationMethod().

8.127.4.6 std::string stdair::PreOptimisationMethod::getMethodAsString () const

Get the enumerated value as a short string (e.g., "M" or "E").

Definition at line 95 of file PreOptimisationMethod.cpp.

8.127.4.7 const std::string stdair::PreOptimisationMethod::describe () const [virtual]

Give a description of the structure (e.g., MRT or FA).

Implements stdair::StructAbstract.

Definition at line 102 of file PreOptimisationMethod.cpp.

8.127.4.8 bool stdair::PreOptimisationMethod::operator== (const EN_PreOptimisationMethod &) const

Comparaison operator.

Definition at line 110 of file PreOptimisationMethod.cpp.

8.127.4.9 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const [inline, inherited]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References stdair::StructAbstract::describe().

8.127.4.10 virtual void stdair::StructAbstract::fromStream (std::istream & *ioIn*) [inline, virtual, inherited]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented in stdair::ProgressStatusSet, stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::EventStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 38 of file StructAbstract.hpp.

Referenced by operator>>().

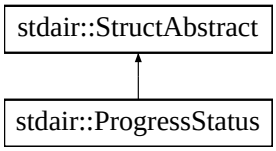
The documentation for this struct was generated from the following files:

- stdair/basic/PreOptimisationMethod.hpp
- stdair/basic/PreOptimisationMethod.cpp

8.128 stdair::ProgressStatus Struct Reference

`#include <stdair/basic/ProgressStatus.hpp>`

Inheritance diagram for stdair::ProgressStatus::



Public Member Functions

- const Count_T & count () const
- const Count_T & getCurrentNb () const
- const Count_T & getExpectedNb () const
- const Count_T & getActualNb () const
- const ProgressPercentage_T progress () const
- void setCurrentNb (const Count_T &iCurrentNb)
- void setExpectedNb (const Count_T &iExpectedNb)
- void setActualNb (const Count_T &iActualNb)
- void reset ()
- Count_T operator+= (Count_T iIncrement)
- Count_T operator++ ()
- const std::string describe () const
- const std::string toString () const
- ProgressStatus (const Count_T &iCurrentNb, const Count_T &iExpectedNb, const Count_T &iActualNb)
- ProgressStatus (const Count_T &iExpectedNb, const Count_T &iActualNb)
- ProgressStatus (const Count_T &iActualNb)
- ProgressStatus ()
- ProgressStatus (const ProgressStatus &)
- void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

8.128.1 Detailed Description

Structure holding the details of a progress status.

The progress status is given by the ratio between the "current" and the "expected" (or "actual") numbers. For instance, when the expected/actual number is 1000 and the current number is 200, then the progress status is 20% (= 200 / 1000).

Definition at line 27 of file ProgressStatus.hpp.

8.128.2 Constructor & Destructor Documentation

8.128.2.1 stdair::ProgressStatus::ProgressStatus (const Count_T & *iCurrentNb*, const Count_T & *iExpectedNb*, const Count_T & *iActualNb*)

Constructor.

Parameters:

const Count_T& The current number.

const Count_T& The expected number.

const Count_T& The actual number.

Definition at line 15 of file ProgressStatus.cpp.

8.128.2.2 stdair::ProgressStatus::ProgressStatus (const Count_T & *iExpectedNb*, const Count_T & *iActualNb*)

Constructor.

As no current number is given, it is set to 0.

Parameters:

const Count_T& The expected number.
const Count_T& The actual number.

Definition at line 23 of file ProgressStatus.cpp.

8.128.2.3 stdair::ProgressStatus::ProgressStatus (const Count_T & *iActualNb*)

Constructor.

As no expected number is given, it is assumed to be equal to the actual one. The current number is set to 0.

Parameters:

const Count_T& The actual number.

Definition at line 30 of file ProgressStatus.cpp.

8.128.2.4 stdair::ProgressStatus::ProgressStatus ()

Constructor.

All the numbers are set to 0.

Definition at line 36 of file ProgressStatus.cpp.

8.128.2.5 stdair::ProgressStatus::ProgressStatus (const ProgressStatus &)

Copy Constructor.

Definition at line 43 of file ProgressStatus.cpp.

8.128.3 Member Function Documentation

8.128.3.1 const Count_T& stdair::ProgressStatus::count () const [inline]

Get the current number.

Definition at line 31 of file ProgressStatus.hpp.

8.128.3.2 const Count_T& stdair::ProgressStatus::getCurrentNb () const [inline]

Get the current number.

Definition at line 36 of file ProgressStatus.hpp.

Referenced by stdair::ProgressStatusSet::describe().

8.128.3.3 const Count_T& stdair::ProgressStatus::getExpectedNb () const [inline]

Get the expected number.

Definition at line 41 of file ProgressStatus.hpp.

Referenced by stdair::ProgressStatusSet::describe().

8.128.3.4 const Count_T& stdair::ProgressStatus::getActualNb () const [inline]

Get the actual number.

Definition at line 46 of file ProgressStatus.hpp.

Referenced by stdair::ProgressStatusSet::describe().

8.128.3.5 const ProgressPercentage_T stdair::ProgressStatus::progress () const [inline]

Get the progress as a percentage.

Definition at line 51 of file ProgressStatus.hpp.

References stdair::MAXIMUM_PROGRESS_STATUS.

Referenced by toString().

8.128.3.6 void stdair::ProgressStatus::setCurrentNb (const Count_T & *iCurrentNb*) [inline]

Set the current number.

Definition at line 65 of file ProgressStatus.hpp.

8.128.3.7 void stdair::ProgressStatus::setExpectedNb (const Count_T & *iExpectedNb*) [inline]

Set the expected number.

Definition at line 70 of file ProgressStatus.hpp.

8.128.3.8 void stdair::ProgressStatus::setActualNb (const Count_T & *iActualNb*) [inline]

Set the actual number.

Definition at line 75 of file ProgressStatus.hpp.

8.128.3.9 void stdair::ProgressStatus::reset ()

Reset the current number (to 0).

Definition at line 50 of file ProgressStatus.cpp.

References stdair::DEFAULT_PROGRESS_STATUS.

8.128.3.10 Count_T stdair::ProgressStatus::operator+= (Count_T *iIncrement*) [inline]

Increment the current number.

Definition at line 83 of file ProgressStatus.hpp.

8.128.3.11 Count_T stdair::ProgressStatus::operator++ () [inline]

Increment the current number.

Definition at line 89 of file ProgressStatus.hpp.

8.128.3.12 const std::string stdair::ProgressStatus::describe () const [virtual]

Give a description of the structure (e.g., "1 {99, 100}").

Implements stdair::StructAbstract.

Definition at line 56 of file ProgressStatus.cpp.

8.128.3.13 const std::string stdair::ProgressStatus::toString () const

Give a description of the structure (e.g., "1% (1/ 100)").

Definition at line 63 of file ProgressStatus.cpp.

References progress().

8.128.3.14 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const [inline, inherited]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References stdair::StructAbstract::describe().

8.128.3.15 virtual void stdair::StructAbstract::fromStream (std::istream & *ioIn*) [inline, virtual, inherited]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented in stdair::ProgressStatusSet, stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::EventStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 38 of file StructAbstract.hpp.

Referenced by operator>>().

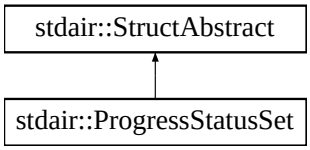
The documentation for this struct was generated from the following files:

- stdair/basic/ProgressStatus.hpp
- stdair/basic/ProgressStatus.cpp

8.129 stdair::ProgressStatusSet Struct Reference

`#include <stdair/basic/ProgressStatusSet.hpp>`

Inheritance diagram for stdair::ProgressStatusSet::



Public Member Functions

- const ProgressStatus & getTypeSpecificStatus () const
- const ProgressStatus & getSpecificGeneratorStatus () const
- const ProgressStatus & getOverallStatus () const
- void setTypeSpecificStatus (const ProgressStatus &iProgressStatus)
- void setSpecificGeneratorStatus (const ProgressStatus &iProgressStatus, const EventGenerator-Key_T &iKey)
- void setOverallStatus (const ProgressStatus &iProgressStatus)
- void fromStream (std::istream &ioIn)
- const std::string describe () const
- ProgressStatusSet (const EventType::EN_EventType &)
- ProgressStatusSet (const ProgressStatusSet &)
- ~ProgressStatusSet ()
- void toStream (std::ostream &ioOut) const

8.129.1 Detailed Description

Structrure holding a set of progress status.

Definition at line 22 of file ProgressStatusSet.hpp.

8.129.2 Constructor & Destructor Documentation

8.129.2.1 stdair::ProgressStatusSet::ProgressStatusSet (const EventType::EN_EventType &)

Constructor .

Definition at line 20 of file ProgressStatusSet.cpp.

8.129.2.2 stdair::ProgressStatusSet::ProgressStatusSet (const ProgressStatusSet &)

Copy constructor.

Definition at line 27 of file ProgressStatusSet.cpp.

8.129.2.3 stdair::ProgressStatusSet::~ProgressStatusSet ()

Destructor.

Definition at line 36 of file ProgressStatusSet.cpp.

8.129.3 Member Function Documentation

8.129.3.1 const ProgressStatus& stdair::ProgressStatusSet::getTypeSpecificStatus () const [inline]

Get the progress status specific to that event type.

Note that that progress status may not be up-to-date. That attribute is up-to-date only after a call to the popEvent() method of SEvMgr.

Definition at line 31 of file ProgressStatusSet.hpp.

8.129.3.2 const ProgressStatus& stdair::ProgressStatusSet::getSpecificGeneratorStatus () const [inline]

Get the progress status specific to the content key for that event.

Note that that progress status may not be up-to-date. That attribute is up-to-date only after a call to the popEvent() method of SEvMgr.

Definition at line 43 of file ProgressStatusSet.hpp.

8.129.3.3 const ProgressStatus& stdair::ProgressStatusSet::getOverallStatus () const [inline]

Get the overall progress status (absolute, for all the events).

Note that that progress status may not be up-to-date. That attribute is up-to-date only after a call to the popEvent() method of SEvMgr.

Definition at line 54 of file ProgressStatusSet.hpp.

8.129.3.4 void stdair::ProgressStatusSet::setTypeSpecificStatus (const ProgressStatus & *iProgressStatus*) [inline]

Set/update the progress status specific to that event type.

Definition at line 62 of file ProgressStatusSet.hpp.

8.129.3.5 void stdair::ProgressStatusSet::setSpecificGeneratorStatus (const ProgressStatus & *iProgressStatus*, const EventGeneratorKey_T & *iKey*) [inline]

Set/update the progress status specific to the content key for that event.

Definition at line 68 of file ProgressStatusSet.hpp.

8.129.3.6 void stdair::ProgressStatusSet::setOverallStatus (const ProgressStatus & *iProgressStatus*) [inline]

Set/update the overall progress status (absolute, for all the events).

Definition at line 76 of file ProgressStatusSet.hpp.

8.129.3.7 void stdair::ProgressStatusSet::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::StructAbstract.

Definition at line 40 of file ProgressStatusSet.cpp.

8.129.3.8 const std::string stdair::ProgressStatusSet::describe () const [virtual]

Display of the structure.

Implements stdair::StructAbstract.

Definition at line 44 of file ProgressStatusSet.cpp.

References stdair::ProgressStatus::getActualNb(), stdair::ProgressStatus::getCurrentNb(), and stdair::ProgressStatus::getExpectedNb().

8.129.3.9 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const [inline, inherited]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References stdair::StructAbstract::describe().

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

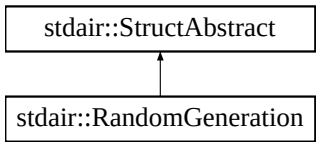
- stdair/basic/ProgressStatusSet.hpp
- stdair/basic/ProgressStatusSet.cpp

8.130 stdair::RandomGeneration Struct Reference

Class holding a random generator.

`#include <stdair/basic/RandomGeneration.hpp>`

Inheritance diagram for stdair::RandomGeneration::



Public Member Functions

- RealNumber_T generateUniform01 ()
- RealNumber_T operator() ()
- RealNumber_T generateUniform (const RealNumber_T &, const RealNumber_T &)
- RealNumber_T generateNormal (const RealNumber_T &, const RealNumber_T &)
- RealNumber_T generateExponential (const RealNumber_T &)
- BaseGenerator_T & getBaseGenerator ()
- const std::string describe () const
- RandomGeneration (const RandomSeed_T &)
- RandomGeneration ()
- ~RandomGeneration ()
- void init (const RandomSeed_T &)
- void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

Public Attributes

- BaseGenerator_T _generator

8.130.1 Detailed Description

Class holding a random generator.

Definition at line 17 of file RandomGeneration.hpp.

8.130.2 Constructor & Destructor Documentation

8.130.2.1 stdair::RandomGeneration::RandomGeneration (const RandomSeed_T &)

Main constructor.

Definition at line 27 of file RandomGeneration.cpp.

8.130.2.2 stdair::RandomGeneration::RandomGeneration ()

Default constructor.

Definition at line 23 of file RandomGeneration.cpp.

8.130.2.3 stdair::RandomGeneration::~RandomGeneration ()

Destructor.

Definition at line 37 of file RandomGeneration.cpp.

8.130.3 Member Function Documentation

8.130.3.1 RealNumber_T stdair::RandomGeneration::generateUniform01 ()

Generate a randomised number following a uniform distribution between 0 (included) and 1 (excluded).

Definition at line 53 of file RandomGeneration.cpp.

References _generator.

Referenced by generateNormal(), generateUniform(), and operator()().

8.130.3.2 RealNumber_T stdair::RandomGeneration::operator() () [inline]

Same as generateUniform01(). That operator is provided for convenient reasons.

Definition at line 30 of file RandomGeneration.hpp.

References generateUniform01().

8.130.3.3 RealNumber_T stdair::RandomGeneration::generateUniform (const RealNumber_T &, const RealNumber_T &)

Generate a randomized number following a uniform distribution between a minimum (included) and a maximum (excluded) value.

Definition at line 59 of file RandomGeneration.cpp.

References generateUniform01().

8.130.3.4 RealNumber_T stdair::RandomGeneration::generateNormal (const RealNumber_T &, const RealNumber_T &)

Generate a randomized number following a normal distribution specified by a mean and a standard deviation.

Definition at line 68 of file RandomGeneration.cpp.

References generateUniform01().

Referenced by stdair::BookingClass::generateDemandSamples().

8.130.3.5 RealNumber_T stdair::RandomGeneration::generateExponential (const RealNumber_T &)

Generate a randomized number following an exponential distribution specified by a mean and a lambda parameter.

Definition at line 86 of file RandomGeneration.cpp.

References _generator.

8.130.3.6 BaseGenerator_T& stdair::RandomGeneration::getBaseGenerator () [inline]

Retrieve the base generator for initialising other random generators.

Definition at line 56 of file RandomGeneration.hpp.

References _generator.

8.130.3.7 const std::string stdair::RandomGeneration::describe () const [virtual]

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

Implements stdair::StructAbstract.

Definition at line 46 of file RandomGeneration.cpp.

References _generator.

8.130.3.8 void stdair::RandomGeneration::init (const RandomSeed_T &)

Initialise the random generator.

A uniform random number distribution is defined, which produces "real" values between 0 and 1 (0 inclusive, 1 exclusive).

Definition at line 41 of file RandomGeneration.cpp.

References _generator.

8.130.3.9 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const [inline, inherited]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References stdair::StructAbstract::describe().

8.130.3.10 virtual void stdair::StructAbstract::fromStream (std::istream & *ioIn*) [inline, virtual, inherited]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented in stdair::ProgressStatusSet, stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::EventStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 38 of file StructAbstract.hpp.

Referenced by operator>>().

8.130.4 Member Data Documentation

8.130.4.1 BaseGenerator_T stdair::RandomGeneration::_generator

Random number generator engine.

The random number generator is currently based on boost::minstd_rand. Alternates are boost::mt19937, boost::ecuyer1988.

Definition at line 112 of file RandomGeneration.hpp.

Referenced by describe(), generateExponential(), generateUniform01(), getBaseGenerator(), and init().

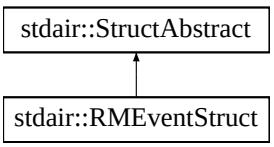
The documentation for this struct was generated from the following files:

- stdair/basic/RandomGeneration.hpp
- stdair/basic/RandomGeneration.cpp

8.131 stdair::RMEventStruct Struct Reference

`#include <stdair/bom/RMEventStruct.hpp>`

Inheritance diagram for stdair::RMEventStruct::



Public Member Functions

- const AirlineCode_T & getAirlineCode () const
- const KeyDescription_T & getFlightDateDescription () const
- const DateTime_T & getRMEventTime () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string describe () const
- RMEventStruct (const AirlineCode_T &, const KeyDescription_T &, const DateTime_T &)
- RMEventStruct (const RMEventStruct &)
- RMEventStruct ()
- ~RMEventStruct ()

8.131.1 Detailed Description

Structure holding the elements of a snapshot .

Definition at line 19 of file RMEventStruct.hpp.

8.131.2 Constructor & Destructor Documentation

8.131.2.1 stdair::RMEventStruct::RMEventStruct (const AirlineCode_T &, const KeyDescription_T &, const DateTime_T &)

Constructor.

Definition at line 27 of file RMEventStruct.cpp.

8.131.2.2 stdair::RMEventStruct::RMEventStruct (const RMEventStruct &)

Copy constructor.

Definition at line 19 of file RMEventStruct.cpp.

8.131.2.3 stdair::RMEventStruct::RMEventStruct ()

Default constructor.

It is private so that it can not be used.

Definition at line 13 of file RMEventStruct.cpp.

8.131.2.4 stdair::RMEventStruct::~RMEventStruct ()

Destructor.

Definition at line 36 of file RMEventStruct.cpp.

8.131.3 Member Function Documentation

8.131.3.1 const AirlineCode_T& stdair::RMEventStruct::getAirlineCode () const [inline]

Get the airline code.

Definition at line 23 of file RMEventStruct.hpp.

8.131.3.2 const KeyDescription_T& stdair::RMEventStruct::getFlightDateDescription () const [inline]

Get the string describing the flight-date key.

Definition at line 28 of file RMEventStruct.hpp.

8.131.3.3 const DateTime_T& stdair::RMEventStruct::getRMEventTime () const [inline]

Get the snapshot action time.

Definition at line 33 of file RMEventStruct.hpp.

8.131.3.4 void stdair::RMEventStruct::toStream (std::ostream & *ioOut*) const

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::StructAbstract.

Definition at line 40 of file RMEventStruct.cpp.

References describe().

8.131.3.5 void stdair::RMEventStruct::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::StructAbstract.

Definition at line 45 of file RMEventStruct.cpp.

8.131.3.6 const std::string stdair::RMEventStruct::describe () const [virtual]

Display of the structure.

Implements stdair::StructAbstract.

Definition at line 49 of file RMEventStruct.cpp.

Referenced by toStream().

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

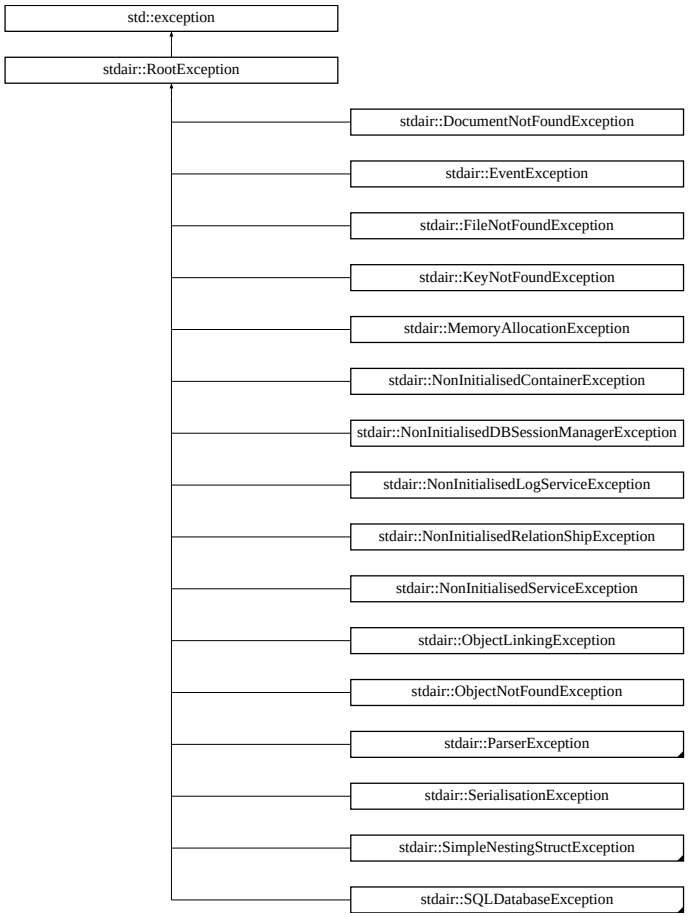
- stdair/bom/RMEventStruct.hpp
- stdair/bom/RMEventStruct.cpp

8.132 stdair::RootException Class Reference

Root of the stdair exceptions.

`#include <stdair/stdair_exceptions.hpp>`

Inheritance diagram for stdair::RootException::



Public Member Functions

- RootException (const std::string &iWhat)
- RootException ()
- virtual ~RootException () throw ()
- const char * what () const throw ()

Protected Attributes

- std::string _what

8.132.1 Detailed Description

Root of the stdair exceptions.

All the stdair exceptions inherit from that root, allowing to catch them and to spot them easily when arising in code wrapping the stdair library.

Definition at line 19 of file stdair_exceptions.hpp.

8.132.2 Constructor & Destructor Documentation

8.132.2.1 stdair::RootException::RootException (const std::string & *iWhat*) [inline]

Main Constructor.

Definition at line 24 of file stdair_exceptions.hpp.

8.132.2.2 stdair::RootException::RootException () [inline]

Default constructor.

Definition at line 28 of file stdair_exceptions.hpp.

8.132.2.3 virtual stdair::RootException::~RootException () throw () [inline, virtual]

Destructor.

Definition at line 33 of file stdair_exceptions.hpp.

8.132.3 Member Function Documentation

8.132.3.1 const char* stdair::RootException::what () const throw () [inline]

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References _what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

8.132.4 Member Data Documentation

8.132.4.1 std::string stdair::RootException::_what [protected]

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

Referenced by what().

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

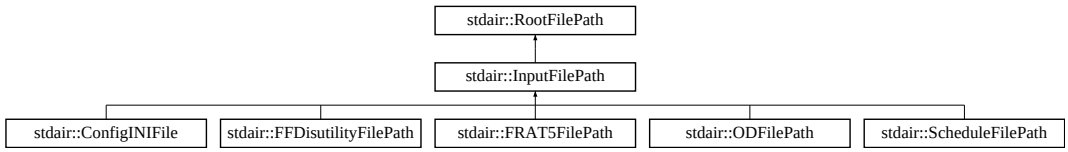
- stdair/stdair_exceptions.hpp

8.133 stdair::RootFilePath Class Reference

Root of the input and output files.

`#include <stdair/stdair_file.hpp>`

Inheritance diagram for stdair::RootFilePath::



Public Member Functions

- RootFilePath (const Filename_T &iFilename)
- RootFilePath ()
- virtual ~RootFilePath ()
- const char * name () const

Protected Attributes

- const Filename_T _filename

8.133.1 Detailed Description

Root of the input and output files.

All the files inherit from that root.

Definition at line 22 of file stdair_file.hpp.

8.133.2 Constructor & Destructor Documentation

8.133.2.1 stdair::RootFilePath::RootFilePath (const Filename_T & *iFilename*) [inline]

Main Constructor.

Definition at line 27 of file stdair_file.hpp.

8.133.2.2 stdair::RootFilePath::RootFilePath () [inline]

Default constructor.

Definition at line 32 of file stdair_file.hpp.

8.133.2.3 virtual stdair::RootFilePath::~RootFilePath () [inline, virtual]

Destructor.

Definition at line 37 of file stdair_file.hpp.

8.133.3 Member Function Documentation

8.133.3.1 const char* stdair::RootFilePath::name () const [inline]

Give the details of the exception.

Definition at line 42 of file stdair_file.hpp.

References _filename.

Referenced by stdair::BomINIImport::importINIConfig().

8.133.4 Member Data Documentation

8.133.4.1 const Filename_T stdair::RootFilePath::_filename [protected]

Name of the file.

Definition at line 50 of file stdair_file.hpp.

Referenced by name().

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

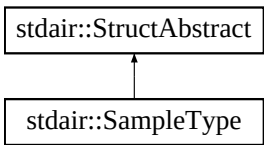
- stdair/stdair_file.hpp

8.134 stdair::SampleType Struct Reference

Enumeration of BOM sample types.

`#include <stdair/basic/SampleType.hpp>`

Inheritance diagram for stdair::SampleType::



Public Types

- ALL = 0
- A4P
- RMS
- INV
- SCH
- RAC
- FQT
- CRS
- DEM
- EVT
- CCM
- LAST_VALUE
- enum EN_SampleType {
  ALL = 0, A4P, RMS, INV,
  SCH, RAC, FQT, CRS,
  DEM, EVT, CCM, LAST_VALUE }

Public Member Functions

- EN_SampleType getType () const
- std::string getTypeAsString () const
- const std::string describe () const
- bool operator== (const EN_SampleType &) const
- SampleType (const EN_SampleType &)
- SampleType (const char iType)
- SampleType (const SampleType &)
- void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

Static Public Member Functions

- static const std::string & getLabel (const EN_SampleType &)
- static char getTypeLabel (const EN_SampleType &)
- static std::string getTypeLabelAsString (const EN_SampleType &)
- static std::string describeLabels ()

8.134.1 Detailed Description

Enumeration of BOM sample types.

In order to test some components, it is often easier to fill the BOM tree with hard-coded structures than set up CSV input files and parsing them. That enumeration structure tells for which component(s) the sample BOM tree should be built. By default, a BOM sample tree is built for all the components, i.e., it contains StdAir objects for all the other components (AirInv, AirSched, etc).

Definition at line 25 of file SampleType.hpp.

8.134.2 Member Enumeration Documentation

8.134.2.1 enum stdair::SampleType::EN_SampleType

Enumerator:

ALL
A4P
RMS
INV
SCH
RAC
FQT
CRS
DEM
EVT
CCM
LAST_VALUE

Definition at line 27 of file SampleType.hpp.

8.134.3 Constructor & Destructor Documentation

8.134.3.1 stdair::SampleType::SampleType (const EN_SampleType &)

Constructor.

Definition at line 36 of file SampleType.cpp.

8.134.3.2 stdair::SampleType::SampleType (const char *iType*)

Constructor.

Definition at line 41 of file SampleType.cpp.

References A4P, ALL, CCM, CRS, DEM, describeLabels(), EVT, FQT, INV, LAST_VALUE, RAC, RMS, and SCH.

8.134.3.3 stdair::SampleType::SampleType (const SampleType &)

Default copy constructor.

Definition at line 31 of file SampleType.cpp.

8.134.4 Member Function Documentation

8.134.4.1 const std::string & stdair::SampleType::getLabel (const EN_SampleType &) [static]

Get the label as a string (e.g., "Inventory" or "Schedule").

Definition at line 67 of file SampleType.cpp.

8.134.4.2 char stdair::SampleType::getTypeLabel (const EN_SampleType &) [static]

Get the label as a single char (e.g., 'I' or 'S').

Definition at line 72 of file SampleType.cpp.

8.134.4.3 std::string stdair::SampleType::getTypeLabelAsString (const EN_SampleType &) [static]

Get the label as a string of a single char (e.g., "I" or "S").

Definition at line 77 of file SampleType.cpp.

8.134.4.4 std::string stdair::SampleType::describeLabels () [static]

List the labels.

Definition at line 84 of file SampleType.cpp.

References LAST_VALUE.

Referenced by SampleType().

8.134.4.5 SampleType::EN_SampleType stdair::SampleType::getType () const

Get the enumerated value.

Definition at line 96 of file SampleType.cpp.

8.134.4.6 std::string stdair::SampleType::getTypeAsString () const

Get the enumerated value as a short string (e.g., "I" or "S").

Definition at line 101 of file SampleType.cpp.

8.134.4.7 const std::string stdair::SampleType::describe () const [virtual]

Give a description of the structure (e.g., "Inventory" or "Schedule").

Implements stdair::StructAbstract.

Definition at line 108 of file SampleType.cpp.

8.134.4.8 bool stdair::SampleType::operator== (const EN_SampleType &) const

Comparison operator.

Definition at line 115 of file SampleType.cpp.

8.134.4.9 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const [inline, inherited]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References stdair::StructAbstract::describe().

8.134.4.10 virtual void stdair::StructAbstract::fromStream (std::istream & *ioIn*) [inline, virtual, inherited]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented in stdair::ProgressStatusSet, stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::EventStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 38 of file StructAbstract.hpp.

Referenced by operator>>().

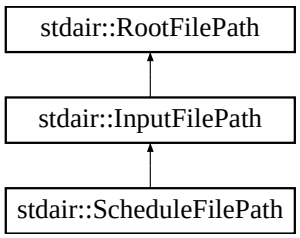
The documentation for this struct was generated from the following files:

- stdair/basic/SampleType.hpp
- stdair/basic/SampleType.cpp

8.135 stdair::ScheduleFilePath Class Reference

`#include <stdair/stdair_file.hpp>`

Inheritance diagram for stdair::ScheduleFilePath::



Public Member Functions

- ScheduleFilePath (const Filename_T &iFilename)
- const char * name () const

Protected Attributes

- const Filename_T _filename

8.135.1 Detailed Description

Schedule input file.

Definition at line 64 of file stdair_file.hpp.

8.135.2 Constructor & Destructor Documentation

8.135.2.1 stdair::ScheduleFilePath::ScheduleFilePath (const Filename_T & *iFilename*) `[inline, explicit]`

Constructor.

Definition at line 69 of file stdair_file.hpp.

8.135.3 Member Function Documentation

8.135.3.1 const char* stdair::RootFilePath::name () const `[inline, inherited]`

Give the details of the exception.

Definition at line 42 of file stdair_file.hpp.

References stdair::RootFilePath::_filename.

Referenced by stdair::BomINIImport::importINIConfig().

8.135.4 Member Data Documentation

8.135.4.1 const Filename_T stdair::RootFilePath::_filename [protected, inherited]

Name of the file.

Definition at line 50 of file stdair_file.hpp.

Referenced by stdair::RootFilePath::name().

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

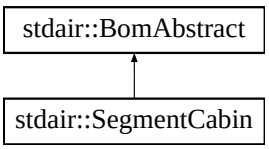
- stdair/stdair_file.hpp

8.136 stdair::SegmentCabin Class Reference

Class representing the actual attributes for an airline segment-cabin.

`#include <stdair/bom/SegmentCabin.hpp>`

Inheritance diagram for stdair::SegmentCabin::



Public Types

- typedef SegmentCabinKey Key_T

Public Member Functions

- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const HolderMap_T & getHolderMap () const
- const CabinCode_T & getCabinCode () const
- const MapKey_T getFullerKey () const
- const SegmentSnapshotTable & getSegmentSnapshotTable () const
- const CabinCapacity_T & getCapacity () const
- const BlockSpace_T & getBlockSpace () const
- const BlockSpace_T & getMIN () const
- const UPR_T & getUPR () const
- const NbOfBookings_T & getBookingCounter () const
- const CommittedSpace_T & getCommittedSpace () const
- const Availability_T & getAvailabilityPool () const
- const BidPrice_T & getCurrentBidPrice () const
- const BidPriceVector_T & getBidPriceVector () const

- const bool getFareFamilyStatus () const
- const PolicyList_T & getConvexHull () const
- void setSegmentSnapshotTable (SegmentSnapshotTable &ioTable)
- void setCapacity (const CabinCapacity_T &iCapacity)
- void setBlockSpace (const BlockSpace_T &iBlockSpace)
- void setMIN (const BlockSpace_T &iMIN)
- void setUPR (const UPR_T &iUPR)
- void setBookingCounter (const NbOfBookings_T &iBookingCounter)
- void setCommittedSpace (const CommittedSpace_T &iCommittedSpace)
- void setAvailabilityPool (const Availability_T &iAvailabilityPool)
- void setBidPriceVector (const BidPriceVector_T &iBPV)
- void activateFareFamily ()
- void updateFromReservation (const NbOfBookings_T &)
- void resetConvexHull ()
- void addPolicy (Policy &)
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- const std::string describeConvexHull () const
- template<class Archive> void serialize (Archive &ar, const unsigned int iFileVersion)

Protected Member Functions

- SegmentCabin (const Key_T &)
- virtual ~SegmentCabin ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent
- HolderMap_T _holderMap
- SegmentSnapshotTable * _segmentSnapshotTable
- CabinCapacity_T _capacity
- BlockSpace_T _blockSpace
- BlockSpace_T _min
- UPR_T _upr
- NbOfBookings_T _bookingCounter
- CommittedSpace_T _committedSpace
- Availability_T _availabilityPool
- BidPriceVector_T _bidPriceVector
- BidPrice_T _currentBidPrice
- bool _fareFamilyActivation
- PolicyList_T _convexHull

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager
- class boost::serialization::access

8.136.1 Detailed Description

Class representing the actual attributes for an airline segment-cabin.

Definition at line 33 of file SegmentCabin.hpp.

8.136.2 Member Typedef Documentation

8.136.2.1 typedef SegmentCabinKey stdair::SegmentCabin::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 44 of file SegmentCabin.hpp.

8.136.3 Constructor & Destructor Documentation

8.136.3.1 stdair::SegmentCabin::SegmentCabin (const Key_T &) [protected]

Constructor.

Definition at line 39 of file SegmentCabin.cpp.

8.136.3.2 stdair::SegmentCabin::~SegmentCabin () [protected, virtual]

Destructor.

Definition at line 52 of file SegmentCabin.cpp.

8.136.4 Member Function Documentation

8.136.4.1 const Key_T& stdair::SegmentCabin::getKey () const [inline]

Get the segment-cabin key (cabin code).

Definition at line 52 of file SegmentCabin.hpp.

References _key.

8.136.4.2 BomAbstract* const stdair::SegmentCabin::getParent () const [inline]

Get the parent object.

Definition at line 59 of file SegmentCabin.hpp.

References _parent.

8.136.4.3 const HolderMap_T& stdair::SegmentCabin::getHolderMap () const [inline]

Get the map of children holders.

Definition at line 66 of file SegmentCabin.hpp.

References _holderMap.

8.136.4.4 const CabinCode_T& stdair::SegmentCabin::getCabinCode () const [inline]

Get the cabin code (primary key).

Definition at line 73 of file SegmentCabin.hpp.

References _key, and stdair::SegmentCabinKey::getCabinCode().

Referenced by getFullerKey().

8.136.4.5 const MapKey_T stdair::SegmentCabin::getFullerKey () const

Get the (segment-date, segment-cabin) key (board point, off point and cabin code).

Note:

That method assumes that the parent object derives from the SegmentDate class, as it needs to have access to the describeKey() method.

Definition at line 56 of file SegmentCabin.cpp.

References stdair::DEFAULT_KEY_FLD_DELIMITER, stdair::SegmentDate::describeKey(), and getCabinCode().

8.136.4.6 const SegmentSnapshotTable& stdair::SegmentCabin::getSegmentSnapshotTable () const [inline]

Get the guilltine block.

Definition at line 88 of file SegmentCabin.hpp.

References _segmentSnapshotTable.

8.136.4.7 const CabinCapacity_T& stdair::SegmentCabin::getCapacity () const [inline]

Get the cabin capacity.

Definition at line 94 of file SegmentCabin.hpp.

References _capacity.

8.136.4.8 const BlockSpace_T& stdair::SegmentCabin::getBlockSpace () const [inline]

Get the blocked number of bookings.

Definition at line 99 of file SegmentCabin.hpp.

References _blockSpace.

8.136.4.9 const BlockSpace_T& stdair::SegmentCabin::getMIN () const [inline]

Get the blocked number of bookings.

Definition at line 104 of file SegmentCabin.hpp.

References _min.

8.136.4.10 const UPR_T& stdair::SegmentCabin::getUPR () const [inline]

Unsold Protection (UPR).

Definition at line 109 of file SegmentCabin.hpp.

References _upr.

8.136.4.11 const NbOfBookings_T& stdair::SegmentCabin::getBookingCounter () const [inline]

Get the booking counter.

Definition at line 114 of file SegmentCabin.hpp.

References _bookingCounter.

8.136.4.12 const CommittedSpace_T& stdair::SegmentCabin::getCommittedSpace () const [inline]

Get the committed Space value.

Definition at line 119 of file SegmentCabin.hpp.

References _committedSpace.

8.136.4.13 const Availability_T& stdair::SegmentCabin::getAvailabilityPool () const [inline]

Get the availability pool value.

Definition at line 124 of file SegmentCabin.hpp.

References _availabilityPool.

8.136.4.14 const BidPrice_T& stdair::SegmentCabin::getCurrentBidPrice () const [inline]

Retrieve the current Bid-Price.

Definition at line 129 of file SegmentCabin.hpp.

References _currentBidPrice.

8.136.4.15 const BidPriceVector_T& stdair::SegmentCabin::getBidPriceVector () const [inline]

Retrieve the Bid-Price Vector.

Definition at line 134 of file SegmentCabin.hpp.

References _bidPriceVector.

8.136.4.16 const bool stdair::SegmentCabin::getFareFamilyStatus () const [inline]

Retrieve the status of fare family.

Definition at line 139 of file SegmentCabin.hpp.

References _fareFamilyActivation.

8.136.4.17 const PolicyList_T& stdair::SegmentCabin::getConvexHull () const [inline]

Retrieve the convex hull.

Definition at line 144 of file SegmentCabin.hpp.

References _convexHull.

8.136.4.18 void stdair::SegmentCabin::setSegmentSnapshotTable (SegmentSnapshotTable & *ioTable*) [inline]

Set the snapshot table.

Definition at line 151 of file SegmentCabin.hpp.

References _segmentSnapshotTable.

8.136.4.19 void stdair::SegmentCabin::setCapacity (const CabinCapacity_T & *iCapacity*) [inline]

Set the cabin capacity.

Definition at line 156 of file SegmentCabin.hpp.

References _capacity.

8.136.4.20 void stdair::SegmentCabin::setBlockSpace (const BlockSpace_T & *iBlockSpace*) [inline]

Set the blocked number of seats.

Definition at line 161 of file SegmentCabin.hpp.

References _blockSpace.

8.136.4.21 void stdair::SegmentCabin::setMIN (const BlockSpace_T & *iMIN*) [inline]

Set the blocked number of seats.

Definition at line 166 of file SegmentCabin.hpp.

References _min.

8.136.4.22 void stdair::SegmentCabin::setUPR (const UPR_T & *iUPR*) [inline]

Set the Unsold Protection (UPR).

Definition at line 171 of file SegmentCabin.hpp.

References _upr.

8.136.4.23 void stdair::SegmentCabin::setBookingCounter (const NbOfBookings_T & *iBookingCounter*) [inline]

Set the total number of bookings.

Definition at line 176 of file SegmentCabin.hpp.

References _bookingCounter.

8.136.4.24 void stdair::SegmentCabin::setCommittedSpace (const CommittedSpace_T & *iCommittedSpace*) [inline]

Set the value of committed space.

Definition at line 181 of file SegmentCabin.hpp.

References _committedSpace.

8.136.4.25 void stdair::SegmentCabin::setAvailabilityPool (const Availability_T & *iAvailabilityPool*) [inline]

Set the value of availability pool.

Definition at line 186 of file SegmentCabin.hpp.

References _availabilityPool.

8.136.4.26 void stdair::SegmentCabin::setBidPriceVector (const BidPriceVector_T & *iBPV*) [inline]

Set the Bid-Price Vector.

Definition at line 191 of file SegmentCabin.hpp.

References _bidPriceVector.

8.136.4.27 void stdair::SegmentCabin::activateFareFamily () [inline]

Activate fare family.

Definition at line 196 of file SegmentCabin.hpp.

References _fareFamilyActivation.

8.136.4.28 void stdair::SegmentCabin::updateFromReservation (const NbOfBookings_T &)

Register a sale.

Definition at line 85 of file SegmentCabin.cpp.

References _committedSpace.

8.136.4.29 void stdair::SegmentCabin::resetConvexHull () [inline]

Reset the convex hull.

Definition at line 206 of file SegmentCabin.hpp.

References _convexHull.

8.136.4.30 void stdair::SegmentCabin::addPolicy (Policy &)

Add a policy to the convex hull. Note: we do not use the FacBomManager here because the convex hull is not a list of children but a temporary list of policies.

Definition at line 90 of file SegmentCabin.cpp.

References _convexHull.

8.136.4.31 void stdair::SegmentCabin::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 221 of file SegmentCabin.hpp.

References toString().

8.136.4.32 void stdair::SegmentCabin::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 230 of file SegmentCabin.hpp.

8.136.4.33 std::string stdair::SegmentCabin::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 65 of file SegmentCabin.cpp.

References describeKey().

Referenced by toStream().

8.136.4.34 const std::string stdair::SegmentCabin::describeKey () const [inline]

Get a string describing the key.

Definition at line 241 of file SegmentCabin.hpp.

References _key, and stdair::SegmentCabinKey::toString().

Referenced by toString().

8.136.4.35 const std::string stdair::SegmentCabin::describeConvexHull () const

Get a string describing the convex hull.

Definition at line 72 of file SegmentCabin.cpp.

References _convexHull.

8.136.4.36 template<class Archive> void stdair::SegmentCabin::serialize (Archive & *ar*, const unsigned int *iFileVersion*)

Serialisation.

Definition at line 229 of file CmdBomSerialiser.cpp.

References _key.

8.136.5 Friends And Related Function Documentation

8.136.5.1 friend class FacBom [friend]

Definition at line 34 of file SegmentCabin.hpp.

8.136.5.2 friend class FacCloneBom [friend]

Definition at line 35 of file SegmentCabin.hpp.

8.136.5.3 friend class FacBomManager [friend]

Definition at line 36 of file SegmentCabin.hpp.

8.136.5.4 friend class boost::serialization::access [friend]

Definition at line 37 of file SegmentCabin.hpp.

8.136.6 Member Data Documentation

8.136.6.1 Key_T stdair::SegmentCabin::_key [protected]

Primary key (cabin code).

Definition at line 300 of file SegmentCabin.hpp.

Referenced by describeKey(), getCabinCode(), getKey(), and serialize().

8.136.6.2 BomAbstract* stdair::SegmentCabin::_parent [protected]

Pointer on the parent class (SegmentDate).

Definition at line 305 of file SegmentCabin.hpp.

Referenced by getParent().

8.136.6.3 HolderMap_T stdair::SegmentCabin::_holderMap [protected]

Map holding the children (FareFamily or BookingClass objects).

Definition at line 310 of file SegmentCabin.hpp.

Referenced by getHolderMap().

8.136.6.4 SegmentSnapshotTable* stdair::SegmentCabin::_segmentSnapshotTable [protected]

The data table used for Revenue Management activities.

Definition at line 315 of file SegmentCabin.hpp.

Referenced by getSegmentSnapshotTable(), and setSegmentSnapshotTable().

8.136.6.5 CabinCapacity_T stdair::SegmentCabin::_capacity [protected]

Capacity of the cabin.

Definition at line 318 of file SegmentCabin.hpp.

Referenced by getCapacity(), and setCapacity().

8.136.6.6 BlockSpace_T stdair::SegmentCabin::_blockSpace [protected]

Blocked capacity.

Definition at line 321 of file SegmentCabin.hpp.

Referenced by getBlockSpace(), and setBlockSpace().

8.136.6.7 BlockSpace_T stdair::SegmentCabin::_min [protected]

Blocked number of seats.

Definition at line 324 of file SegmentCabin.hpp.

Referenced by getMIN(), and setMIN().

8.136.6.8 UPR_T stdair::SegmentCabin::_upr [protected]

Unsold Protection (UPR).

Definition at line 327 of file SegmentCabin.hpp.

Referenced by getUPR(), and setUPR().

8.136.6.9 NbOfBookings_T stdair::SegmentCabin::_bookingCounter [protected]

Aggregated number of bookings.

Definition at line 330 of file SegmentCabin.hpp.

Referenced by getBookingCounter(), and setBookingCounter().

8.136.6.10 CommittedSpace_T stdair::SegmentCabin::_committedSpace [protected]

Aggreated committed space.

Definition at line 333 of file SegmentCabin.hpp.

Referenced by getCommittedSpace(), setCommittedSpace(), and updateFromReservation().

8.136.6.11 Availability_T stdair::SegmentCabin::_availabilityPool [protected]

Aggregated availability pool.

Definition at line 336 of file SegmentCabin.hpp.

Referenced by getAvailabilityPool(), and setAvailabilityPool().

8.136.6.12 BidPriceVector_T stdair::SegmentCabin::_bidPriceVector [protected]

Bid-Price Vector (BPV).

Definition at line 339 of file SegmentCabin.hpp.

Referenced by getBidPriceVector(), and setBidPriceVector().

8.136.6.13 BidPrice_T stdair::SegmentCabin::_currentBidPrice [protected]

Current Bid-Price (BP).

Definition at line 342 of file SegmentCabin.hpp.

Referenced by getCurrentBidPrice().

8.136.6.14 bool stdair::SegmentCabin::_fareFamilyActivation [protected]

Indicate if fare family is in use.

Definition at line 345 of file SegmentCabin.hpp.

Referenced by activateFareFamily(), and getFareFamilyStatus().

8.136.6.15 PolicyList_T stdair::SegmentCabin::_convexHull [protected]

The convex hull of MRT.

Definition at line 348 of file SegmentCabin.hpp.

Referenced by addPolicy(), describeConvexHull(), getConvexHull(), and resetConvexHull().

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

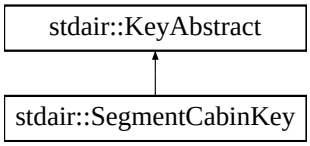
- stdair/bom/SegmentCabin.hpp
- stdair/bom/SegmentCabin.cpp
- stdair/command/CmdBomSerialiser.cpp

8.137 stdair::SegmentCabinKey Struct Reference

Key of a given segment-cabin, made of a cabin code (only).

`#include <stdair/bom/SegmentCabinKey.hpp>`

Inheritance diagram for stdair::SegmentCabinKey::



Public Member Functions

- SegmentCabinKey (const CabinCode_T &iCabinCode)
- SegmentCabinKey (const SegmentCabinKey &)
- ~SegmentCabinKey ()
- const CabinCode_T & getCabinCode () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const
- template<class Archive> void serialize (Archive &ar, const unsigned int iFileVersion)

Friends

- class boost::serialization::access

8.137.1 Detailed Description

Key of a given segment-cabin, made of a cabin code (only).

Definition at line 26 of file SegmentCabinKey.hpp.

8.137.2 Constructor & Destructor Documentation

8.137.2.1 stdair::SegmentCabinKey::SegmentCabinKey (const CabinCode_T & *iCabinCode*)

Constructor.

Definition at line 23 of file SegmentCabinKey.cpp.

8.137.2.2 stdair::SegmentCabinKey::SegmentCabinKey (const SegmentCabinKey &)

Copy constructor.

Definition at line 28 of file SegmentCabinKey.cpp.

8.137.2.3 stdair::SegmentCabinKey::~SegmentCabinKey ()

Destructor.

Definition at line 33 of file SegmentCabinKey.cpp.

8.137.3 Member Function Documentation

8.137.3.1 const CabinCode_T& stdair::SegmentCabinKey::getCabinCode () const [inline]

Get the cabin code.

Definition at line 56 of file SegmentCabinKey.hpp.

Referenced by stdair::SegmentCabin::getCabinCode().

8.137.3.2 void stdair::SegmentCabinKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 37 of file SegmentCabinKey.cpp.

References toString().

8.137.3.3 void stdair::SegmentCabinKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 42 of file SegmentCabinKey.cpp.

8.137.3.4 const std::string stdair::SegmentCabinKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from stdair::KeyAbstract.

Definition at line 46 of file SegmentCabinKey.cpp.

Referenced by stdair::SegmentCabin::describeKey(), and toStream().

8.137.3.5 template<class Archive> void stdair::SegmentCabinKey::serialize (Archive & *ar*, const unsigned int *iFileVersion*)

Serialisation.

Definition at line 68 of file SegmentCabinKey.cpp.

8.137.4 Friends And Related Function Documentation

8.137.4.1 friend class boost::serialization::access [friend]

Definition at line 27 of file SegmentCabinKey.hpp.

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

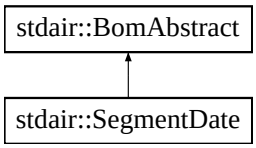
- stdair/bom/SegmentCabinKey.hpp
- stdair/bom/SegmentCabinKey.cpp

8.138 stdair::SegmentDate Class Reference

Class representing the actual attributes for an airline segment-date.

`#include <stdair/bom/SegmentDate.hpp>`

Inheritance diagram for stdair::SegmentDate::



Public Types

- typedef SegmentDateKey Key_T

Public Member Functions

- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const AirportCode_T & getBoardingPoint () const
- const AirportCode_T & getOffPoint () const
- const HolderMap_T & getHolderMap () const
- const Date_T & getBoardingDate () const
- const Duration_T & getBoardingTime () const
- const Date_T & getOffDate () const
- const Duration_T & getOffTime () const
- const Duration_T & getElapsedTime () const
- const Distance_T & getDistance () const
- const DateOffset_T getDateOffset () const
- const Duration_T getTimeOffset () const
- SegmentDate * getOperatingSegmentDate () const
- const SegmentDateList_T & getMarketingSegmentDateList () const
- const RoutingLegKeyList_T & getLegKeyList () const
- void setBoardingDate (const Date_T &iBoardingDate)
- void setBoardingTime (const Duration_T &iBoardingTime)
- void setOffDate (const Date_T &iOffDate)
- void setOffTime (const Duration_T &iOffTime)
- void setElapsedTime (const Duration_T &iElapsedTime)
- void setDistance (const Distance_T &iDistance)
- void addLegKey (const std::string &iLegKey)
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- template<class Archive> void serialize (Archive &ar, const unsigned int iFileVersion)

Protected Member Functions

- SegmentDate (const Key_T &)
- virtual ~SegmentDate ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent
- HolderMap_T _holderMap
- SegmentDate * _operatingSegmentDate
- SegmentDateList_T _marketingSegmentDateList
- Date_T _boardingDate
- Duration_T _boardingTime
- Date_T _offDate

- Duration_T _offTime
- Duration_T _elapsedTime
- Distance_T _distance
- RoutingLegKeyList_T _routingLegKeyList

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager
- class boost::serialization::access

8.138.1 Detailed Description

Class representing the actual attributes for an airline segment-date.

Definition at line 36 of file SegmentDate.hpp.

8.138.2 Member Typedef Documentation

8.138.2.1 typedef SegmentDateKey stdair::SegmentDate::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 47 of file SegmentDate.hpp.

8.138.3 Constructor & Destructor Documentation

8.138.3.1 stdair::SegmentDate::SegmentDate (const Key_T &) [protected]

Constructor.

Definition at line 38 of file SegmentDate.cpp.

8.138.3.2 stdair::SegmentDate::~SegmentDate () [protected, virtual]

Destructor.

Definition at line 44 of file SegmentDate.cpp.

8.138.4 Member Function Documentation

8.138.4.1 const Key_T& stdair::SegmentDate::getKey () const [inline]

Get the segment-date key.

Definition at line 55 of file SegmentDate.hpp.

References _key.

8.138.4.2 BomAbstract* const stdair::SegmentDate::getParent () const [inline]

Get the parent object.

Definition at line 62 of file SegmentDate.hpp.

References _parent.

8.138.4.3 const AirportCode_T& stdair::SegmentDate::getBoardingPoint () const [inline]

Get the boarding point (part of the primary key).

Definition at line 69 of file SegmentDate.hpp.

References _key, and stdair::SegmentDateKey::getBoardingPoint().

8.138.4.4 const AirportCode_T& stdair::SegmentDate::getOffPoint () const [inline]

Get the off point (part of the primary key).

Definition at line 76 of file SegmentDate.hpp.

References _key, and stdair::SegmentDateKey::getOffPoint().

8.138.4.5 const HolderMap_T& stdair::SegmentDate::getHolderMap () const [inline]

Get the map of children holders.

Definition at line 83 of file SegmentDate.hpp.

References _holderMap.

8.138.4.6 const Date_T& stdair::SegmentDate::getBoardingDate () const [inline]

Get the boarding date.

Definition at line 90 of file SegmentDate.hpp.

References _boardingDate.

8.138.4.7 const Duration_T& stdair::SegmentDate::getBoardingTime () const [inline]

Get the boarding time.

Definition at line 97 of file SegmentDate.hpp.

References _boardingTime.

8.138.4.8 const Date_T& stdair::SegmentDate::getOffDate () const [inline]

Get the off date.

Definition at line 104 of file SegmentDate.hpp.

References _offDate.

8.138.4.9 const Duration_T& stdair::SegmentDate::getOffTime () const [inline]

Get the off time.

Definition at line 111 of file SegmentDate.hpp.

References _offTime.

8.138.4.10 const Duration_T& stdair::SegmentDate::getElapsedTime () const [inline]

Get the elapsed time.

Definition at line 118 of file SegmentDate.hpp.

References _elapsedTime.

8.138.4.11 const Distance_T& stdair::SegmentDate::getDistance () const [inline]

Get the distance.

Definition at line 125 of file SegmentDate.hpp.

References _distance.

8.138.4.12 const DateOffset_T stdair::SegmentDate::getDateOffset () const [inline]

Get the date offset (off date - boarding date).

Definition at line 132 of file SegmentDate.hpp.

References _boardingDate, and _offDate.

Referenced by getTimeOffset().

8.138.4.13 const Duration_T stdair::SegmentDate::getTimeOffset () const

Get the time offset between boarding and off points.

It is defined as being:

TimeOffset = (OffTime - BoardingTime) + (OffDate - BoardingDate) $*$ 24

- ElapsedTime.

Definition at line 55 of file SegmentDate.cpp.

References _boardingTime, _elapsedTime, _offTime, and getDateOffset().

8.138.4.14 SegmentDate* stdair::SegmentDate::getOperatingSegmentDate () const [inline]

Get the "operating" segment date.

Definition at line 149 of file SegmentDate.hpp.

References _operatingSegmentDate.

8.138.4.15 const SegmentDateList_T& stdair::SegmentDate::getMarketingSegmentDateList () const [inline]

Get the list of marketing segment dates.

Definition at line 156 of file SegmentDate.hpp.

References _marketingSegmentDateList.

8.138.4.16 const RoutingLegKeyList_T& stdair::SegmentDate::getLegKeyList () const [inline]

Get the list of routing leg keys.

Definition at line 163 of file SegmentDate.hpp.

References _routingLegKeyList.

8.138.4.17 void stdair::SegmentDate::setBoardingDate (const Date_T & *iBoardingDate*) [inline]

Set the boarding date.

Definition at line 172 of file SegmentDate.hpp.

References _boardingDate.

8.138.4.18 void stdair::SegmentDate::setBoardingTime (const Duration_T & *iBoardingTime*) [inline]

Set the boarding time.

Definition at line 179 of file SegmentDate.hpp.

References _boardingTime.

8.138.4.19 void stdair::SegmentDate::setOffDate (const Date_T & *iOffDate*) [inline]

Set the off date.

Definition at line 186 of file SegmentDate.hpp.

References _offDate.

8.138.4.20 void stdair::SegmentDate::setOffTime (const Duration_T & *iOffTime*) [inline]

Set the off time.

Definition at line 193 of file SegmentDate.hpp.

References _offTime.

8.138.4.21 void stdair::SegmentDate::setElapsedTime (const Duration_T & *iElapsedTime*) [inline]

Set the elapsed time.

Definition at line 200 of file SegmentDate.hpp.

References _elapsedTime.

8.138.4.22 void stdair::SegmentDate::setDistance (const Distance_T & *iDistance*) [inline]

Set the distance.

Definition at line 207 of file SegmentDate.hpp.

References _distance.

8.138.4.23 void stdair::SegmentDate::addLegKey (const std::string & *iLegKey*) [inline]

Add a routing leg key to the list.

Definition at line 214 of file SegmentDate.hpp.

References _routingLegKeyList.

8.138.4.24 void stdair::SegmentDate::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 233 of file SegmentDate.hpp.

References toString().

8.138.4.25 void stdair::SegmentDate::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 242 of file SegmentDate.hpp.

8.138.4.26 std::string stdair::SegmentDate::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 48 of file SegmentDate.cpp.

References describeKey().

Referenced by stdair::BomRetriever::retrieveDummySegmentCabin(), and toStream().

8.138.4.27 const std::string stdair::SegmentDate::describeKey () const [inline]

Get a string describing the key.

Definition at line 253 of file SegmentDate.hpp.

References _key, and stdair::SegmentDateKey::toString().

Referenced by stdair::SegmentCabin::getFullerKey(), stdair::BomRetriever::retrieveFullKeyFromSegmentDate(), and toString().

8.138.4.28 template<class Archive> void stdair::SegmentDate::serialize (Archive & *ar*, const unsigned int *iFileVersion*)

Serialisation.

Definition at line 208 of file CmdBomSerialiser.cpp.

References _key.

8.138.5 Friends And Related Function Documentation

8.138.5.1 friend class FacBom [friend]

Definition at line 37 of file SegmentDate.hpp.

8.138.5.2 friend class FacCloneBom [friend]

Definition at line 38 of file SegmentDate.hpp.

8.138.5.3 friend class FacBomManager [friend]

Definition at line 39 of file SegmentDate.hpp.

8.138.5.4 friend class boost::serialization::access [friend]

Definition at line 40 of file SegmentDate.hpp.

8.138.6 Member Data Documentation

8.138.6.1 Key_T stdair::SegmentDate::_key [protected]

Primary key (origin and destination).

Definition at line 307 of file SegmentDate.hpp.

Referenced by describeKey(), getBoardingPoint(), getKey(), getOffPoint(), and serialize().

8.138.6.2 BomAbstract* stdair::SegmentDate::_parent [protected]

Pointer on the parent class (FlightDate).

Definition at line 312 of file SegmentDate.hpp.

Referenced by getParent().

8.138.6.3 HolderMap_T stdair::SegmentDate::_holderMap [protected]

Map holding the children (SegmentCabin objects).

Definition at line 317 of file SegmentDate.hpp.

Referenced by getHolderMap().

8.138.6.4 SegmentDate* stdair::SegmentDate::_operatingSegmentDate [protected]

Pointer on the operating SegmentDate. Nota: 1. "operating" refers to the codeshare contract seller. 2. the pointer will be NULL if the segment date is itself the "operating" one.

Definition at line 325 of file SegmentDate.hpp.

Referenced by getOperatingSegmentDate().

8.138.6.5 SegmentDateList_T stdair::SegmentDate::_marketingSegmentDateList [protected]

List holding the marketing segment dates. Nota: 1. "marketing" refers to the codeshare contract seller. 2. the list will be empty if the segment date is itself the "marketing" one.

Definition at line 333 of file SegmentDate.hpp.

Referenced by getMarketingSegmentDateList().

8.138.6.6 Date_T stdair::SegmentDate::_boardingDate [protected]

Boarding date.

Definition at line 338 of file SegmentDate.hpp.

Referenced by getBoardingDate(), getDateOffset(), and setBoardingDate().

8.138.6.7 Duration_T stdair::SegmentDate::_boardingTime [protected]

Boarding time.

Definition at line 343 of file SegmentDate.hpp.

Referenced by getBoardingTime(), getTimeOffset(), and setBoardingTime().

8.138.6.8 Date_T stdair::SegmentDate::_offDate [protected]

Landing date.

Definition at line 348 of file SegmentDate.hpp.

Referenced by getDateOffset(), getOffDate(), and setOffDate().

8.138.6.9 Duration_T stdair::SegmentDate::_offTime [protected]

Landing time.

Definition at line 353 of file SegmentDate.hpp.

Referenced by getOffTime(), getTimeOffset(), and setOffTime().

8.138.6.10 Duration_T stdair::SegmentDate::_elapsedTime [protected]

Trip elapsed time.

Definition at line 358 of file SegmentDate.hpp.

Referenced by getElapsedTime(), getTimeOffset(), and setElapsedTime().

8.138.6.11 Distance_T stdair::SegmentDate::_distance [protected]

Trip distance.

Definition at line 363 of file SegmentDate.hpp.

Referenced by getDistance(), and setDistance().

8.138.6.12 RoutingLegKeyList_T stdair::SegmentDate::_routingLegKeyList [protected]

List of routing leg keys.

Definition at line 368 of file SegmentDate.hpp.

Referenced by addLegKey(), and getLegKeyList().

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

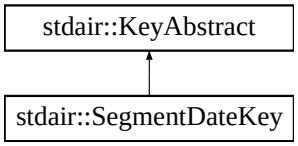
- stdair/bom/SegmentDate.hpp
- stdair/bom/SegmentDate.cpp
- stdair/command/CmdBomSerialiser.cpp

8.139 stdair::SegmentDateKey Struct Reference

Key of a given segment-date, made of an origin and a destination airports.

`#include <stdair/bom/SegmentDateKey.hpp>`

Inheritance diagram for stdair::SegmentDateKey::



Public Member Functions

- SegmentDateKey (const AirportCode_T &, const AirportCode_T &)
- SegmentDateKey (const SegmentDateKey &)
- ~SegmentDateKey ()
- const AirportCode_T & getBoardingPoint () const
- const AirportCode_T & getOffPoint () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const
- template<class Archive> void serialize (Archive &ar, const unsigned int iFileVersion)

Friends

- class boost::serialization::access

8.139.1 Detailed Description

Key of a given segment-date, made of an origin and a destination airports.

Definition at line 24 of file SegmentDateKey.hpp.

8.139.2 Constructor & Destructor Documentation

8.139.2.1 stdair::SegmentDateKey::SegmentDateKey (const AirportCode_T &, const AirportCode_T &)

Main constructor.

Definition at line 25 of file SegmentDateKey.cpp.

8.139.2.2 stdair::SegmentDateKey::SegmentDateKey (const SegmentDateKey &)

Copy constructor.

Definition at line 31 of file SegmentDateKey.cpp.

8.139.2.3 stdair::SegmentDateKey::~SegmentDateKey ()

Destructor.

Definition at line 36 of file SegmentDateKey.cpp.

8.139.3 Member Function Documentation

8.139.3.1 const AirportCode_T& stdair::SegmentDateKey::getBoardingPoint () const [inline]

Get the boardinging point.

Definition at line 51 of file SegmentDateKey.hpp.

Referenced by stdair::SegmentDate::getBoardingPoint(), and stdair::OnDDateKey::getOrigin().

8.139.3.2 const AirportCode_T& stdair::SegmentDateKey::getOffPoint () const [inline]

Get the arrival point.

Definition at line 56 of file SegmentDateKey.hpp.

Referenced by stdair::OnDDateKey::getDestination(), and stdair::SegmentDate::getOffPoint().

8.139.3.3 void stdair::SegmentDateKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 40 of file SegmentDateKey.cpp.

References toString().

8.139.3.4 void stdair::SegmentDateKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 45 of file SegmentDateKey.cpp.

8.139.3.5 const std::string stdair::SegmentDateKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from stdair::KeyAbstract.

Definition at line 49 of file SegmentDateKey.cpp.

References stdair::DEFAULT_KEY_SUB_FLD_DELIMITER.

Referenced by stdair::SegmentDate::describeKey(), stdair::FlightDate::getSegmentDate(), stdair::BomRetriever::retrieveSegmentDateFromLongKey(), and toStream().

8.139.3.6 template<class Archive> void stdair::SegmentDateKey::serialize (Archive & *ar*, const unsigned int *iFileVersion*)

Serialisation.

Definition at line 72 of file SegmentDateKey.cpp.

8.139.4 Friends And Related Function Documentation

8.139.4.1 friend class boost::serialization::access [friend]

Definition at line 25 of file SegmentDateKey.hpp.

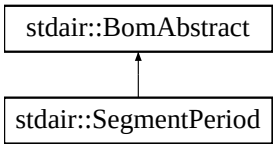
The documentation for this struct was generated from the following files:

- stdair/bom/SegmentDateKey.hpp
- stdair/bom/SegmentDateKey.cpp

8.140 stdair::SegmentPeriod Class Reference

`#include <stdair/bom/SegmentPeriod.hpp>`

Inheritance diagram for stdair::SegmentPeriod::



Public Types

- typedef SegmentPeriodKey Key_T

Public Member Functions

- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const AirportCode_T & getBoardingPoint () const
- const AirportCode_T & getOffPoint () const
- const Duration_T & getBoardingTime () const
- const Duration_T & getOffTime () const
- const DateOffset_T & getBoardingDateOffset () const
- const DateOffset_T & getOffDateOffset () const
- const Duration_T & getElapsedTime () const
- const CabinBookingClassMap_T & getCabinBookingClassMap () const
- const HolderMap_T & getHolderMap () const
- void setBoardingTime (const Duration_T &iBoardingTime)
- void setOffTime (const Duration_T &iOffTime)
- void setBoardingDateOffset (const DateOffset_T &iDateOffset)
- void setOffDateOffset (const DateOffset_T &iDateOffset)
- void setElapsedTime (const Duration_T &iElapsedTime)
- void addCabinBookingClassList (const CabinCode_T &, const ClassList_String_T &)
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const

Protected Member Functions

- SegmentPeriod (const Key_T &)
- virtual ~SegmentPeriod ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent
- Duration_T _boardingTime
- Duration_T _offTime
- DateOffset_T _boardingDateOffset
- DateOffset_T _offDateOffset
- Duration_T _elapsedTime
- CabinBookingClassMap_T _cabinBookingClassMap
- HolderMap_T _holderMap

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager

8.140.1 Detailed Description

Class representing the actual attributes for an airline segment-period.

Definition at line 15 of file SegmentPeriod.hpp.

8.140.2 Member Typedef Documentation

8.140.2.1 typedef SegmentPeriodKey stdair::SegmentPeriod::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 23 of file SegmentPeriod.hpp.

8.140.3 Constructor & Destructor Documentation

8.140.3.1 stdair::SegmentPeriod::SegmentPeriod (const Key_T &) [protected]

Main constructor.

Definition at line 13 of file SegmentPeriod.cpp.

8.140.3.2 stdair::SegmentPeriod::~SegmentPeriod () [protected, virtual]

Destructor.

Definition at line 29 of file SegmentPeriod.cpp.

8.140.4 Member Function Documentation

8.140.4.1 const Key_T& stdair::SegmentPeriod::getKey () const [inline]

Get the segment-period key.

Definition at line 28 of file SegmentPeriod.hpp.

References _key.

8.140.4.2 BomAbstract* const stdair::SegmentPeriod::getParent () const [inline]

Get the parent object.

Definition at line 31 of file SegmentPeriod.hpp.

References _parent.

8.140.4.3 const AirportCode_T& stdair::SegmentPeriod::getBoardingPoint () const [inline]

Get the boarding point (part of the primary key).

Definition at line 34 of file SegmentPeriod.hpp.

References _key, and stdair::SegmentPeriodKey::getBoardingPoint().

8.140.4.4 const AirportCode_T& stdair::SegmentPeriod::getOffPoint () const [inline]

Get the off point (part of the primary key).

Definition at line 39 of file SegmentPeriod.hpp.

References _key, and stdair::SegmentPeriodKey::getOffPoint().

8.140.4.5 const Duration_T& stdair::SegmentPeriod::getBoardingTime () const [inline]

Get the boarding time.

Definition at line 42 of file SegmentPeriod.hpp.

References _boardingTime.

8.140.4.6 const Duration_T& stdair::SegmentPeriod::getOffTime () const [inline]

Get the off time.

Definition at line 45 of file SegmentPeriod.hpp.

References _offTime.

8.140.4.7 const DateOffset_T& stdair::SegmentPeriod::getBoardingDateOffset () const [inline]

Get the boarding date offset.

Definition at line 48 of file SegmentPeriod.hpp.

References _boardingDateOffset.

8.140.4.8 const DateOffset_T& stdair::SegmentPeriod::getOffDateOffset () const [inline]

Get the off date offset.

Definition at line 53 of file SegmentPeriod.hpp.

References _offDateOffset.

8.140.4.9 const Duration_T& stdair::SegmentPeriod::getElapsedTime () const [inline]

Get the elapsed time.

Definition at line 56 of file SegmentPeriod.hpp.

References _elapsedTime.

8.140.4.10 const CabinBookingClassMap_T& stdair::SegmentPeriod::getCabinBookingClassMap () const [inline]

Get the cabin booking class map.

Definition at line 59 of file SegmentPeriod.hpp.

References _cabinBookingClassMap.

8.140.4.11 const HolderMap_T& stdair::SegmentPeriod::getHolderMap () const [inline]

Get the map of children holders.

Definition at line 64 of file SegmentPeriod.hpp.

References _holderMap.

8.140.4.12 void stdair::SegmentPeriod::setBoardingTime (const Duration_T & *iBoardingTime*) [inline]

Set the boarding time.

Definition at line 69 of file SegmentPeriod.hpp.

References _boardingTime.

8.140.4.13 void stdair::SegmentPeriod::setOffTime (const Duration_T & *iOffTime*) [inline]

Set the off time.

Definition at line 74 of file SegmentPeriod.hpp.

References _offTime.

8.140.4.14 void stdair::SegmentPeriod::setBoardingDateOffset (const DateOffset_T & *iDateOffset*) [inline]

Set the boarding date offset.

Definition at line 77 of file SegmentPeriod.hpp.

References _boardingDateOffset.

8.140.4.15 void stdair::SegmentPeriod::setOffDateOffset (const DateOffset_T & *iDateOffset*) [inline]

Set the off date offset.

Definition at line 82 of file SegmentPeriod.hpp.

References _offDateOffset.

8.140.4.16 void stdair::SegmentPeriod::setElapsedTime (const Duration_T & *iElapsedTime*) [inline]

Set the elapsed time.

Definition at line 87 of file SegmentPeriod.hpp.

References _elapsedTime.

8.140.4.17 void stdair::SegmentPeriod::addCabinBookingClassList (const CabinCode_T &, const ClassList_String_T &)

Add a pair cabin code and list of class codes within the cabin to the cabin booking class map.

Definition at line 41 of file SegmentPeriod.cpp.

References _cabinBookingClassMap.

8.140.4.18 void stdair::SegmentPeriod::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 100 of file SegmentPeriod.hpp.

References toString().

8.140.4.19 void stdair::SegmentPeriod::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 104 of file SegmentPeriod.hpp.

8.140.4.20 std::string stdair::SegmentPeriod::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 33 of file SegmentPeriod.cpp.

References describeKey().

Referenced by toStream().

8.140.4.21 const std::string stdair::SegmentPeriod::describeKey () const [inline]

Get a string describing the key.

Definition at line 110 of file SegmentPeriod.hpp.

References _key, and stdair::SegmentPeriodKey::toString().

Referenced by toString().

8.140.5 Friends And Related Function Documentation

8.140.5.1 friend class FacBom [friend]

Definition at line 16 of file SegmentPeriod.hpp.

8.140.5.2 friend class FacCloneBom [friend]

Definition at line 17 of file SegmentPeriod.hpp.

8.140.5.3 friend class FacBomManager [friend]

Definition at line 18 of file SegmentPeriod.hpp.

8.140.6 Member Data Documentation

8.140.6.1 Key_T stdair::SegmentPeriod::_key [protected]

Definition at line 135 of file SegmentPeriod.hpp.

Referenced by describeKey(), getBoardingPoint(), getKey(), and getOffPoint().

8.140.6.2 BomAbstract* stdair::SegmentPeriod::_parent [protected]

Definition at line 136 of file SegmentPeriod.hpp.

Referenced by getParent().

8.140.6.3 Duration_T stdair::SegmentPeriod::_boardingTime [protected]

Definition at line 137 of file SegmentPeriod.hpp.

Referenced by getBoardingTime(), and setBoardingTime().

8.140.6.4 Duration_T stdair::SegmentPeriod::_offTime [protected]

Definition at line 138 of file SegmentPeriod.hpp.

Referenced by getOffTime(), and setOffTime().

8.140.6.5 DateOffset_T stdair::SegmentPeriod::_boardingDateOffset [protected]

Definition at line 139 of file SegmentPeriod.hpp.

Referenced by getBoardingDateOffset(), and setBoardingDateOffset().

8.140.6.6 DateOffset_T stdair::SegmentPeriod::_offDateOffset [protected]

Definition at line 140 of file SegmentPeriod.hpp.

Referenced by getOffDateOffset(), and setOffDateOffset().

8.140.6.7 Duration_T stdair::SegmentPeriod::_elapsedTime [protected]

Definition at line 141 of file SegmentPeriod.hpp.

Referenced by getElapsedTime(), and setElapsedTime().

8.140.6.8 CabinBookingClassMap_T stdair::SegmentPeriod::_cabinBookingClassMap [protected]

Definition at line 142 of file SegmentPeriod.hpp.

Referenced by addCabinBookingClassList(), and getCabinBookingClassMap().

8.140.6.9 HolderMap_T stdair::SegmentPeriod::_holderMap [protected]

Definition at line 143 of file SegmentPeriod.hpp.

Referenced by getHolderMap().

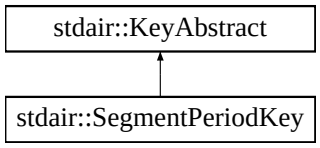
The documentation for this class was generated from the following files:

- stdair/bom/SegmentPeriod.hpp
- stdair/bom/SegmentPeriod.cpp

8.141 stdair::SegmentPeriodKey Struct Reference

`#include <stdair/bom/SegmentPeriodKey.hpp>`

Inheritance diagram for stdair::SegmentPeriodKey::



Public Member Functions

- SegmentPeriodKey (const AirportCode_T &, const AirportCode_T &)
- SegmentPeriodKey (const SegmentPeriodKey &)
- ~SegmentPeriodKey ()
- const AirportCode_T & getBoardingPoint () const
- const AirportCode_T & getOffPoint () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const

8.141.1 Detailed Description

Key of segment-period.

Definition at line 14 of file SegmentPeriodKey.hpp.

8.141.2 Constructor & Destructor Documentation

8.141.2.1 stdair::SegmentPeriodKey::SegmentPeriodKey (const AirportCode_T &, const AirportCode_T &)

Constructors.

Definition at line 12 of file SegmentPeriodKey.cpp.

8.141.2.2 stdair::SegmentPeriodKey::SegmentPeriodKey (const SegmentPeriodKey &)

Definition at line 18 of file SegmentPeriodKey.cpp.

8.141.2.3 stdair::SegmentPeriodKey::~SegmentPeriodKey ()

Destructor.

Definition at line 23 of file SegmentPeriodKey.cpp.

8.141.3 Member Function Documentation

8.141.3.1 const AirportCode_T& stdair::SegmentPeriodKey::getBoardingPoint () const [inline]

Get the boardinging point.

Definition at line 29 of file SegmentPeriodKey.hpp.

Referenced by stdair::SegmentPeriod::getBoardingPoint().

8.141.3.2 const AirportCode_T& stdair::SegmentPeriodKey::getOffPoint () const [inline]

Get the arrival point.

Definition at line 34 of file SegmentPeriodKey.hpp.

Referenced by stdair::SegmentPeriod::getOffPoint().

8.141.3.3 void stdair::SegmentPeriodKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 27 of file SegmentPeriodKey.cpp.

References toString().

8.141.3.4 void stdair::SegmentPeriodKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 32 of file SegmentPeriodKey.cpp.

8.141.3.5 const std::string stdair::SegmentPeriodKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-period.

Reimplemented from stdair::KeyAbstract.

Definition at line 36 of file SegmentPeriodKey.cpp.

Referenced by stdair::SegmentPeriod::describeKey(), and toStream().

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

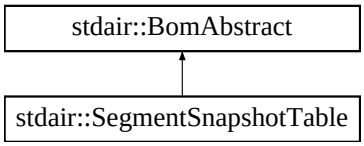
- stdair/bom/SegmentPeriodKey.hpp
- stdair/bom/SegmentPeriodKey.cpp

8.142 stdair::SegmentSnapshotTable Class Reference

Class representing the actual attributes for an airline segment data tables.

`#include <stdair/bom/SegmentSnapshotTable.hpp>`

Inheritance diagram for stdair::SegmentSnapshotTable::



Public Types

- typedef SegmentSnapshotTableKey Key_T

Public Member Functions

- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const TableID_T & getTableID () const
- const HolderMap_T & getHolderMap () const
- const SegmentCabinIndexMap_T & getSegmentCabinIndexMap () const
- const ClassIndexMap_T & getClassIndexMap () const
- const ClassIndex_T & getClassIndex (const MapKey_T &) const
- const SegmentDataID_T & getSegmentDataID (const SegmentCabin &) const
- ConstSegmentCabinDTDSnapshotView_T getConstSegmentCabinDTDBookingSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T) const
- ConstSegmentCabinDTDRangeSnapshotView_T getConstSegmentCabinDTDRangeBooking-SnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T, const DTD_T) const
- SegmentCabinDTDSnapshotView_T getSegmentCabinDTDBookingSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T)
- SegmentCabinDTDRangeSnapshotView_T getSegmentCabinDTDRangeBookingSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T, const DTD_T)
- ConstSegmentCabinDTDSnapshotView_T getConstSegmentCabinDTDCancellationSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T) const
- ConstSegmentCabinDTDRangeSnapshotView_T getConstSegmentCabinDTDRangeCancellation-SnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T, const DTD_T) const
- SegmentCabinDTDSnapshotView_T getSegmentCabinDTDCancellationSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T)
- SegmentCabinDTDRangeSnapshotView_T getSegmentCabinDTDRangeCancellationSnapshot-View (const SegmentDataID_T, const SegmentDataID_T, const DTD_T, const DTD_T)
- ConstSegmentCabinDTDSnapshotView_T getConstSegmentCabinDTDProductOrientedNet-BookingSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T) const

- ConstSegmentCabinDTDRangeSnapshotView_T getConstSegmentCabinDTDRangeProductOrientedNetBookingSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T, const DTD_T) const
- SegmentCabinDTDSnapshotView_T getSegmentCabinDTDProductOrientedNetBookingSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T)
- SegmentCabinDTDRangeSnapshotView_T getSegmentCabinDTDRangeProductOrientedNetBookingSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T, const DTD_T)
- ConstSegmentCabinDTDSnapshotView_T getConstSegmentCabinDTDPriceOrientedNetBookingSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T) const
- ConstSegmentCabinDTDRangeSnapshotView_T getConstSegmentCabinDTDRangePriceOrientedNetBookingSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T, const DTD_T) const
- SegmentCabinDTDSnapshotView_T getSegmentCabinDTDPriceOrientedNetBookingSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T)
- SegmentCabinDTDRangeSnapshotView_T getSegmentCabinDTDRangePriceOrientedNetBookingSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T, const DTD_T)
- ConstSegmentCabinDTDSnapshotView_T getConstSegmentCabinDTDProductOrientedGrossBookingSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T) const
- ConstSegmentCabinDTDRangeSnapshotView_T getConstSegmentCabinDTDRangeProductOrientedGrossBookingSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T, const DTD_T) const
- SegmentCabinDTDSnapshotView_T getSegmentCabinDTDProductOrientedGrossBookingSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T)
- SegmentCabinDTDRangeSnapshotView_T getSegmentCabinDTDRangeProductOrientedGrossBookingSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T, const DTD_T)
- ConstSegmentCabinDTDSnapshotView_T getConstSegmentCabinDTDPriceOrientedGrossBookingSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T) const
- ConstSegmentCabinDTDRangeSnapshotView_T getConstSegmentCabinDTDRangePriceOrientedGrossBookingSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T, const DTD_T) const
- SegmentCabinDTDSnapshotView_T getSegmentCabinDTDPriceOrientedGrossBookingSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T)
- SegmentCabinDTDRangeSnapshotView_T getSegmentCabinDTDRangePriceOrientedGrossBookingSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T, const DTD_T)
- ConstSegmentCabinDTDSnapshotView_T getConstSegmentCabinDTDAvailabilitySnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T) const
- ConstSegmentCabinDTDRangeSnapshotView_T getConstSegmentCabinDTDRangeAvailabilitySnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T, const DTD_T) const
- SegmentCabinDTDSnapshotView_T getSegmentCabinDTDAvailabilitySnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T)
- SegmentCabinDTDRangeSnapshotView_T getSegmentCabinDTDRangeAvailabilitySnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T, const DTD_T)
- void initSnapshotBlocks (const SegmentCabinIndexMap_T &, const ClassIndexMap_T &)
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- template<class Archive> void serialize (Archive &ar, const unsigned int iFileVersion)

Protected Member Functions

- SegmentSnapshotTable (const Key_T &)
- virtual ~SegmentSnapshotTable ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent
- HolderMap_T _holderMap
- SegmentCabinIndexMap_T _segmentCabinIndexMap
- ClassIndexMap_T _classIndexMap
- SnapshotBlock_T _bookingSnapshotBlock
- SnapshotBlock_T _cancellationSnapshotBlock
- SnapshotBlock_T _productOrientedNetBookingSnapshotBlock
- SnapshotBlock_T _priceOrientedNetBookingSnapshotBlock
- SnapshotBlock_T _productOrientedGrossBookingSnapshotBlock
- SnapshotBlock_T _priceOrientedGrossBookingSnapshotBlock
- SnapshotBlock_T _availabilitySnapshotBlock

Friends

- class FacBom
- class FacBomManager
- class boost::serialization::access

8.142.1 Detailed Description

Class representing the actual attributes for an airline segment data tables.

Definition at line 31 of file SegmentSnapshotTable.hpp.

8.142.2 Member Typedef Documentation

8.142.2.1 typedef SegmentSnapshotTableKey stdair::SegmentSnapshotTable::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 41 of file SegmentSnapshotTable.hpp.

8.142.3 Constructor & Destructor Documentation

8.142.3.1 stdair::SegmentSnapshotTable::SegmentSnapshotTable (const Key_T &) [protected]

Main constructor.

Definition at line 34 of file SegmentSnapshotTable.cpp.

8.142.3.2 stdair::SegmentSnapshotTable::~SegmentSnapshotTable () [protected, virtual]

Destructor.

Definition at line 38 of file SegmentSnapshotTable.cpp.

8.142.4 Member Function Documentation

8.142.4.1 const Key_T& stdair::SegmentSnapshotTable::getKey () const [inline]

Get the segment data table key.

Definition at line 47 of file SegmentSnapshotTable.hpp.

References _key.

8.142.4.2 BomAbstract* const stdair::SegmentSnapshotTable::getParent () const [inline]

Get the parent object.

Definition at line 52 of file SegmentSnapshotTable.hpp.

References _parent.

8.142.4.3 const TableID_T& stdair::SegmentSnapshotTable::getTableID () const [inline]

Get the table ID (part of the primary key).

Definition at line 57 of file SegmentSnapshotTable.hpp.

References _key, and stdair::SegmentSnapshotTableKey::getTableID().

8.142.4.4 const HolderMap_T& stdair::SegmentSnapshotTable::getHolderMap () const [inline]

Get the map of children holders.

Definition at line 64 of file SegmentSnapshotTable.hpp.

References _holderMap.

8.142.4.5 const SegmentCabinIndexMap_T& stdair::SegmentSnapshotTable::getSegmentCabinIndexMap () const [inline]

Get the segment-cabin index map.

Definition at line 69 of file SegmentSnapshotTable.hpp.

References _segmentCabinIndexMap.

8.142.4.6 const ClassIndexMap_T& stdair::SegmentSnapshotTable::getClassIndexMap () const [inline]

Get the class index map.

Definition at line 74 of file SegmentSnapshotTable.hpp.

References _classIndexMap.

8.142.4.7 const ClassIndex_T & stdair::SegmentSnapshotTable::getClassIndex (const MapKey_T &) const

Get the index corresponding to the given class.

Definition at line 88 of file SegmentSnapshotTable.cpp.

References _classIndexMap.

8.142.4.8 const SegmentDataID_T & stdair::SegmentSnapshotTable::getSegmentDataID (const SegmentCabin &) const

Get the segment data ID corresponding to the givent segment-cabin.

Definition at line 97 of file SegmentSnapshotTable.cpp.

References _segmentCabinIndexMap.

8.142.4.9 ConstSegmentCabinDTDSnapshotView_T stdair::SegmentSnapshotTable::getConstSegmentCabinDTDBookingSnapshotView (const *SegmentDataID_T*, const *SegmentDataID_T*, const *DTD_T*) const

Get the const view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 106 of file SegmentSnapshotTable.cpp.

References _bookingSnapshotBlock, and _classIndexMap.

8.142.4.10 ConstSegmentCabinDTDRangeSnapshotView_T stdair::SegmentSnapshotTable::getConstSegmentCabinDTDRangeBookingSnapshotView (const *SegmentDataID_T*, const *SegmentDataID_T*, const *DTD_T*, const *DTD_T*) const

Get the const view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 119 of file SegmentSnapshotTable.cpp.

8.142.4.11 SegmentCabinDTDSnapshotView_T stdair::SegmentSnapshotTable::getSegmentCabinDTDBookingSnapshotView (const *SegmentDataID_T*, const *SegmentDataID_T*, const *DTD_T*)

Get the view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 130 of file SegmentSnapshotTable.cpp.

References _bookingSnapshotBlock, and _classIndexMap.

8.142.4.12 SegmentCabinDTDRangeSnapshotView_T stdair::SegmentSnapshotTable::getSegmentCabinDTDRangeBookingSnapshotView (const *SegmentDataID_T*, const *SegmentDataID_T*, const *DTD_T*, const *DTD_T*)

Get the view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 142 of file SegmentSnapshotTable.cpp.

References _bookingSnapshotBlock, and _classIndexMap.

8.142.4.13 ConstSegmentCabinDTDSnapshotView_T stdair::SegmentSnapshotTable::getConstSegmentCabinDTDCancellationSnapshotView (const *SegmentDataID_T*, const *SegmentDataID_T*, const *DTD_T*) const

Get the const view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 155 of file SegmentSnapshotTable.cpp.

References _cancellationSnapshotBlock, and _classIndexMap.

8.142.4.14 ConstSegmentCabinDTDRangeSnapshotView_T stdair::SegmentSnapshotTable::getConstSegmentCabinDTDRangeCancellationSnapshotView (const *SegmentDataID_T*, const *SegmentDataID_T*, const *DTD_T*, const *DTD_T*) const

Get the const view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 168 of file SegmentSnapshotTable.cpp.

8.142.4.15 SegmentCabinDTDSnapshotView_T stdair::SegmentSnapshotTable::getSegmentCabinDTDCancellationSnapshotView (const *SegmentDataID_T*, const *SegmentDataID_T*, const *DTD_T*)

Get the view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 179 of file SegmentSnapshotTable.cpp.

References _cancellationSnapshotBlock, and _classIndexMap.

8.142.4.16 SegmentCabinDTDRangeSnapshotView_T stdair::SegmentSnapshotTable::getSegmentCabinDTDRangeCancellationSnapshotView (const *SegmentDataID_T*, const *SegmentDataID_T*, const *DTD_T*, const *DTD_T*)

Get the view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 191 of file SegmentSnapshotTable.cpp.

References _cancellationSnapshotBlock, and _classIndexMap.

8.142.4.17 ConstSegmentCabinDTDSnapshotView_T stdair::SegmentSnapshotTable::getConstSegmentCabinDTDProductOrientedNetBookingSnapshotView (const *SegmentDataID_T*, const *SegmentDataID_T*, const *DTD_T*) const

Get the const view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 204 of file SegmentSnapshotTable.cpp.

References _classIndexMap, and _productOrientedNetBookingSnapshotBlock.

8.142.4.18 ConstSegmentCabinDTDRangeSnapshotView_T stdair::SegmentSnapshotTable::getConstSegmentCabinDTDRangeProductOrientedNetBookingSnapshotView (const *SegmentDataID_T*, const *SegmentDataID_T*, const *DTD_T*, const *DTD_T*) const

Get the const view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 217 of file SegmentSnapshotTable.cpp.

8.142.4.19 SegmentCabinDTDSnapshotView_T stdair::SegmentSnapshotTable::getSegmentCabinDTDProductOrientedNetBookingSnapshotView (const *SegmentDataID_T*, const *SegmentDataID_T*, const *DTD_T*)

Get the view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 228 of file SegmentSnapshotTable.cpp.

References _classIndexMap, and _productOrientedNetBookingSnapshotBlock.

8.142.4.20 SegmentCabinDTDRangeSnapshotView_T stdair::SegmentSnapshotTable::getSegmentCabinDTDRangeProductOrientedNetBookingSnapshotView (const *SegmentDataID_T*, const *SegmentDataID_T*, const *DTD_T*, const *DTD_T*)

Get the view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 240 of file SegmentSnapshotTable.cpp.

References _classIndexMap, and _productOrientedNetBookingSnapshotBlock.

8.142.4.21 ConstSegmentCabinDTDSnapshotView_T stdair::SegmentSnapshotTable::getConstSegmentCabinDTDPriceOrientedNetBookingSnapshotView (const *SegmentDataID_T*, const *SegmentDataID_T*, const *DTD_T*) const

Get the const view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 254 of file SegmentSnapshotTable.cpp.

References _classIndexMap, and _priceOrientedNetBookingSnapshotBlock.

8.142.4.22 ConstSegmentCabinDTDRangeSnapshotView_T stdair::SegmentSnapshotTable::getConstSegmentCabinDTDRangePriceOrientedNetBookingSnapshotView (const *SegmentDataID_T*, const *SegmentDataID_T*, const *DTD_T*, const *DTD_T*) const

Get the const view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 267 of file SegmentSnapshotTable.cpp.

8.142.4.23 SegmentCabinDTDSnapshotView_T stdair::SegmentSnapshotTable::getSegmentCabinDTDPriceOrientedNetBookingSnapshotView (const *SegmentDataID_T*, const *SegmentDataID_T*, const *DTD_T*)

Get the view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 278 of file SegmentSnapshotTable.cpp.

References _classIndexMap, and _priceOrientedNetBookingSnapshotBlock.

8.142.4.24 SegmentCabinDTDRangeSnapshotView_T stdair::SegmentSnapshotTable::getSegmentCabinDTDRangePriceOrientedNetBookingSnapshotView (const *SegmentDataID_T*, const *SegmentDataID_T*, const *DTD_T*, const *DTD_T*)

Get the view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 290 of file SegmentSnapshotTable.cpp.

References _classIndexMap, and _priceOrientedNetBookingSnapshotBlock.

8.142.4.25 ConstSegmentCabinDTDSnapshotView_T stdair::SegmentSnapshotTable::getConstSegmentCabinDTDProductOrientedGrossBookingSnapshotView (const *SegmentDataID_T*, const *SegmentDataID_T*, const *DTD_T*) const

Get the const view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 303 of file SegmentSnapshotTable.cpp.

References _classIndexMap, and _productOrientedGrossBookingSnapshotBlock.

8.142.4.26 ConstSegmentCabinDTDRangeSnapshotView_T stdair::SegmentSnapshotTable::getConstSegmentCabinDTDRangeProductOrientedGrossBookingSnapshotView (const *SegmentDataID_T*, const *SegmentDataID_T*, const *DTD_T*, const *DTD_T*) const

Get the const view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 316 of file SegmentSnapshotTable.cpp.

8.142.4.27 SegmentCabinDTDSnapshotView_T stdair::SegmentSnapshotTable::getSegmentCabinDTDProductOrientedGrossBookingSnapshotView (const *SegmentDataID_T*, const *SegmentDataID_T*, const *DTD_T*)

Get the view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 327 of file SegmentSnapshotTable.cpp.

References _classIndexMap, and _productOrientedGrossBookingSnapshotBlock.

8.142.4.28 SegmentCabinDTDRangeSnapshotView_T stdair::SegmentSnapshotTable::getSegmentCabinDTDRangeProductOrientedGrossBookingSnapshotView (const *SegmentDataID_T*, const *SegmentDataID_T*, const *DTD_T*, const *DTD_T*)

Get the view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 339 of file SegmentSnapshotTable.cpp.

References _classIndexMap, and _productOrientedGrossBookingSnapshotBlock.

8.142.4.29 ConstSegmentCabinDTDSnapshotView_T stdair::SegmentSnapshotTable::getConstSegmentCabinDTDPriceOrientedGrossBookingSnapshotView (const *SegmentDataID_T*, const *SegmentDataID_T*, const *DTD_T*) const

Get the const view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 353 of file SegmentSnapshotTable.cpp.

References _classIndexMap, and _priceOrientedGrossBookingSnapshotBlock.

8.142.4.30 ConstSegmentCabinDTDRangeSnapshotView_T stdair::SegmentSnapshotTable::getConstSegmentCabinDTDRangePriceOrientedGrossBookingSnapshotView (const *SegmentDataID_T*, const *SegmentDataID_T*, const *DTD_T*, const *DTD_T*) const

Get the const view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 366 of file SegmentSnapshotTable.cpp.

8.142.4.31 SegmentCabinDTDSnapshotView_T stdair::SegmentSnapshotTable::getSegmentCabinDTDPriceOrientedGrossBookingSnapshotView (const *SegmentDataID_T*, const *SegmentDataID_T*, const *DTD_T*)

Get the view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 378 of file SegmentSnapshotTable.cpp.

8.142.4.32 SegmentCabinDTDRangeSnapshotView_T stdair::SegmentSnapshotTable::getSegmentCabinDTDRangePriceOrientedGrossBookingSnapshotView (const *SegmentDataID_T*, const *SegmentDataID_T*, const *DTD_T*, const *DTD_T*)

Get the view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 390 of file SegmentSnapshotTable.cpp.

8.142.4.33 ConstSegmentCabinDTDSnapshotView_T stdair::SegmentSnapshotTable::getConstSegmentCabinDTDAvailabilitySnapshotView (const *SegmentDataID_T*, const *SegmentDataID_T*, const *DTD_T*) const

Get the const view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 402 of file SegmentSnapshotTable.cpp.

8.142.4.34 ConstSegmentCabinDTDRangeSnapshotView_T stdair::SegmentSnapshotTable::getConstSegmentCabinDTDRangeAvailabilitySnapshotView (const *SegmentDataID_T*, const *SegmentDataID_T*, const *DTD_T*, const *DTD_T*) const

Get the const view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 414 of file SegmentSnapshotTable.cpp.

8.142.4.35 SegmentCabinDTDSnapshotView_T stdair::SegmentSnapshotTable::getSegmentCabinDTDAvailabilitySnapshotView (const *SegmentDataID_T*, const *SegmentDataID_T*, const *DTD_T*)

Get the view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 425 of file SegmentSnapshotTable.cpp.

References _availabilitySnapshotBlock, and _classIndexMap.

8.142.4.36 SegmentCabinDTDRangeSnapshotView_T stdair::SegmentSnapshotTable::getSegmentCabinDTDRangeAvailabilitySnapshotView (const *SegmentDataID_T*, const *SegmentDataID_T*, const *DTD_T*, const *DTD_T*)

Get the view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 437 of file SegmentSnapshotTable.cpp.

References _availabilitySnapshotBlock, and _classIndexMap.

8.142.4.37 void stdair::SegmentSnapshotTable::initSnapshotBlocks (const SegmentCabinIndexMap_T &, const ClassIndexMap_T &)

Set the segment-cabin and value type index maps and initialise the snapshot blocks.

Definition at line 50 of file SegmentSnapshotTable.cpp.

References _availabilitySnapshotBlock, _bookingSnapshotBlock, _cancellationSnapshotBlock, _classIndexMap, _priceOrientedGrossBookingSnapshotBlock, _priceOrientedNetBookingSnapshotBlock, _productOrientedGrossBookingSnapshotBlock, _productOrientedNetBookingSnapshotBlock, _segmentCabinIndexMap, and stdair::DEFAULT_MAX_DTD.

8.142.4.38 void stdair::SegmentSnapshotTable::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 283 of file SegmentSnapshotTable.hpp.

References toString().

8.142.4.39 void stdair::SegmentSnapshotTable::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 292 of file SegmentSnapshotTable.hpp.

8.142.4.40 std::string stdair::SegmentSnapshotTable::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 42 of file SegmentSnapshotTable.cpp.

References describeKey().

Referenced by toStream().

8.142.4.41 const std::string stdair::SegmentSnapshotTable::describeKey () const [inline]

Get a string describing the key.

Definition at line 303 of file SegmentSnapshotTable.hpp.

References _key, and stdair::SegmentSnapshotTableKey::toString().

Referenced by toString().

8.142.4.42 template<class Archive> void stdair::SegmentSnapshotTable::serialize (Archive & *ar*, const unsigned int *iFileVersion*)

Serialisation.

Definition at line 464 of file SegmentSnapshotTable.cpp.

References _key.

8.142.5 Friends And Related Function Documentation

8.142.5.1 friend class FacBom [friend]

Definition at line 32 of file SegmentSnapshotTable.hpp.

8.142.5.2 friend class FacBomManager [friend]

Definition at line 33 of file SegmentSnapshotTable.hpp.

8.142.5.3 friend class boost::serialization::access [friend]

Definition at line 34 of file SegmentSnapshotTable.hpp.

8.142.6 Member Data Documentation

8.142.6.1 Key_T stdair::SegmentSnapshotTable::_key [protected]

Primary key (table ID and departure block).

Definition at line 352 of file SegmentSnapshotTable.hpp.

Referenced by describeKey(), getKey(), getTableID(), and serialize().

8.142.6.2 BomAbstract* stdair::SegmentSnapshotTable::_parent [protected]

Pointer on the parent class (Inventory).

Definition at line 355 of file SegmentSnapshotTable.hpp.

Referenced by getParent().

8.142.6.3 HolderMap_T stdair::SegmentSnapshotTable::_holderMap [protected]

Map holding the children.

Definition at line 358 of file SegmentSnapshotTable.hpp.

Referenced by getHolderMap().

8.142.6.4 SegmentCabinIndexMap_T stdair::SegmentSnapshotTable::_segmentCabinIndexMap [protected]

Map holding the segment-cabin position within the snapshot blocks.

Definition at line 361 of file SegmentSnapshotTable.hpp.

Referenced by getSegmentCabinIndexMap(), getSegmentDataID(), and initSnapshotBlocks().

8.142.6.5 ClassIndexMap_T stdair::SegmentSnapshotTable::_classIndexMap [protected]

Map holding the value type (class, etc) within a a segment-cabin inside the snapshot blocks.

Definition at line 365 of file SegmentSnapshotTable.hpp.

Referenced by getClassIndex(), getClassIndexMap(), getConstSegmentCabinDTDBookingSnapshotView(), getConstSegmentCabinDTDCancellationSnapshotView(), getConstSegmentCabinDTDPriceOrientedGrossBookingSnapshotView(), getConstSegmentCabinDTDPriceOrientedNetBookingSnapshotView(), getConstSegmentCabinDTDProductOrientedGrossBookingSnapshotView(), getConstSegmentCabinDTDProductOrientedNetBookingSnapshotView(), getSegmentCabinDTDAvailabilitySnapshotView(), getSegmentCabinDTDBookingSnapshotView(), getSegmentCabinDTDCancellationSnapshotView(), getSegmentCabinDTDPriceOrientedNetBookingSnapshotView(), getSegmentCabinDTDProductOrientedGrossBookingSnapshotView(), getSegmentCabinDTDProductOrientedNetBookingSnapshotView(), getSegmentCabinDTDRangeAvailabilitySnapshotView(), getSegmentCabinDTDRange-

BookingSnapshotView(), getSegmentCabinDTDRangeCancellationSnapshotView(), getSegmentCabinDTDRangePriceOrientedNetBookingSnapshotView(), getSegmentCabinDTDRangeProductOrientedGrossBookingSnapshotView(), getSegmentCabinDTDRangeProductOrientedNetBookingSnapshotView(), and initSnapshotBlocks().

8.142.6.6 SnapshotBlock_T stdair::SegmentSnapshotTable::_bookingSnapshotBlock [protected]

Booking snapshot block.

Definition at line 368 of file SegmentSnapshotTable.hpp.

Referenced by getConstSegmentCabinDTDBookingSnapshotView(), getSegmentCabinDTDBookingSnapshotView(), getSegmentCabinDTDRangeBookingSnapshotView(), and initSnapshotBlocks().

8.142.6.7 SnapshotBlock_T stdair::SegmentSnapshotTable::_cancellationSnapshotBlock [protected]

Cancellation snapshot block.

Definition at line 371 of file SegmentSnapshotTable.hpp.

Referenced by getConstSegmentCabinDTDCancellationSnapshotView(), getSegmentCabinDTDCancellationSnapshotView(), getSegmentCabinDTDRangeCancellationSnapshotView(), and initSnapshotBlocks().

8.142.6.8 SnapshotBlock_T stdair::SegmentSnapshotTable::_productOrientedNetBookingSnapshotBlock [protected]

Product oriented net booking block.

Definition at line 374 of file SegmentSnapshotTable.hpp.

Referenced by getConstSegmentCabinDTDProductOrientedNetBookingSnapshotView(), getSegmentCabinDTDProductOrientedNetBookingSnapshotView(), getSegmentCabinDTDRangeProductOrientedNetBookingSnapshotView(), and initSnapshotBlocks().

8.142.6.9 SnapshotBlock_T stdair::SegmentSnapshotTable::_priceOrientedNetBookingSnapshotBlock [protected]

Price oriented net booking block.

Definition at line 377 of file SegmentSnapshotTable.hpp.

Referenced by getConstSegmentCabinDTDPriceOrientedNetBookingSnapshotView(), getSegmentCabinDTDPriceOrientedNetBookingSnapshotView(), getSegmentCabinDTDRangePriceOrientedNetBookingSnapshotView(), and initSnapshotBlocks().

8.142.6.10 SnapshotBlock_T stdair::SegmentSnapshotTable::_productOrientedGrossBookingSnapshotBlock [protected]

Product oriented gross booking block.

Definition at line 380 of file SegmentSnapshotTable.hpp.

Referenced by getConstSegmentCabinDTDProductOrientedGrossBookingSnapshotView(), getSegmentCabinDTDProductOrientedGrossBookingSnapshotView(), getSegmentCabinDTDRangeProductOrientedGrossBookingSnapshotView(), and initSnapshotBlocks().

8.142.6.11 SnapshotBlock_T stdair::SegmentSnapshotTable::_priceOrientedGrossBookingSnapshotBlock [protected]

Price oriented gross booking block.

Definition at line 383 of file SegmentSnapshotTable.hpp.

Referenced by getConstSegmentCabinDTDPriceOrientedGrossBookingSnapshotView(), and initSnapshotBlocks().

8.142.6.12 SnapshotBlock_T stdair::SegmentSnapshotTable::_availabilitySnapshotBlock [protected]

Availability block.

Definition at line 386 of file SegmentSnapshotTable.hpp.

Referenced by getSegmentCabinDTDAvailabilitySnapshotView(), getSegmentCabinDTDRangeAvailabilitySnapshotView(), and initSnapshotBlocks().

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

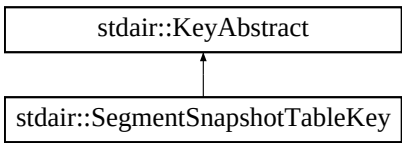
- stdair/bom/SegmentSnapshotTable.hpp
- stdair/bom/SegmentSnapshotTable.cpp

8.143 stdair::SegmentSnapshotTableKey Struct Reference

Key of a given guillotine block, made of a guillotine number.

`#include <stdair/bom/SegmentSnapshotTableKey.hpp>`

Inheritance diagram for stdair::SegmentSnapshotTableKey::



Public Member Functions

- SegmentSnapshotTableKey (const TableID_T &)
- SegmentSnapshotTableKey (const SegmentSnapshotTableKey &)
- ~SegmentSnapshotTableKey ()
- const TableID_T & getTableID () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const
- template<class Archive> void serialize (Archive &ar, const unsigned int iFileVersion)

Friends

- class boost::serialization::access

8.143.1 Detailed Description

Key of a given guillotine block, made of a guillotine number.

Definition at line 26 of file SegmentSnapshotTableKey.hpp.

8.143.2 Constructor & Destructor Documentation

8.143.2.1 stdair::SegmentSnapshotTableKey::SegmentSnapshotTableKey (const TableID_T &)

Constructor.

Definition at line 26 of file SegmentSnapshotTableKey.cpp.

8.143.2.2 stdair::SegmentSnapshotTableKey::SegmentSnapshotTableKey (const SegmentSnapshotTableKey &)

Copy constructor.

Definition at line 31 of file SegmentSnapshotTableKey.cpp.

8.143.2.3 stdair::SegmentSnapshotTableKey::~SegmentSnapshotTableKey ()

Destructor.

Definition at line 36 of file SegmentSnapshotTableKey.cpp.

8.143.3 Member Function Documentation

8.143.3.1 const TableID_T& stdair::SegmentSnapshotTableKey::getTableID () const [inline]

Get the table ID.

Definition at line 56 of file SegmentSnapshotTableKey.hpp.

Referenced by stdair::SegmentSnapshotTable::getTableID().

8.143.3.2 void stdair::SegmentSnapshotTableKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 40 of file SegmentSnapshotTableKey.cpp.

References toString().

8.143.3.3 void stdair::SegmentSnapshotTableKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 45 of file SegmentSnapshotTableKey.cpp.

8.143.3.4 const std::string stdair::SegmentSnapshotTableKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-block.

Reimplemented from stdair::KeyAbstract.

Definition at line 49 of file SegmentSnapshotTableKey.cpp.

Referenced by stdair::SegmentSnapshotTable::describeKey(), and toStream().

8.143.3.5 template<class Archive> void stdair::SegmentSnapshotTableKey::serialize (Archive & *ar*, const unsigned int *iFileVersion*)

Serialisation.

Definition at line 71 of file SegmentSnapshotTableKey.cpp.

8.143.4 Friends And Related Function Documentation

8.143.4.1 friend class boost::serialization::access [friend]

Definition at line 27 of file SegmentSnapshotTableKey.hpp.

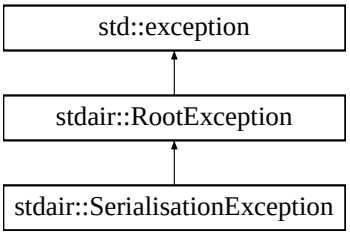
The documentation for this struct was generated from the following files:

- stdair/bom/SegmentSnapshotTableKey.hpp
- stdair/bom/SegmentSnapshotTableKey.cpp

8.144 stdair::SerialisationException Class Reference

`#include <stdair/stdair_exceptions.hpp>`

Inheritance diagram for stdair::SerialisationException::



Public Member Functions

- SerialisationException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

8.144.1 Detailed Description

Serialisation.

Definition at line 119 of file stdair_exceptions.hpp.

8.144.2 Constructor & Destructor Documentation

8.144.2.1 stdair::SerialisationException::SerialisationException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 122 of file stdair_exceptions.hpp.

8.144.3 Member Function Documentation

8.144.3.1 const char* stdair::RootException::what () const throw () [inline, inherited]

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

8.144.4 Member Data Documentation

8.144.4.1 std::string stdair::RootException::_what [protected, inherited]

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

Referenced by stdair::RootException::what().

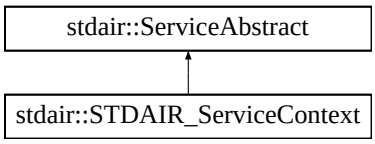
The documentation for this class was generated from the following file:

- stdair/stdair_exceptions.hpp

8.145 stdair::ServiceAbstract Class Reference

`#include <stdair/service/ServiceAbstract.hpp>`

Inheritance diagram for stdair::ServiceAbstract::



Public Member Functions

- virtual ~ServiceAbstract ()
- virtual void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

Protected Member Functions

- ServiceAbstract ()

8.145.1 Detailed Description

Base class for the Service layer.

Definition at line 15 of file ServiceAbstract.hpp.

8.145.2 Constructor & Destructor Documentation

8.145.2.1 virtual stdair::ServiceAbstract::~ServiceAbstract () [inline, virtual]

Destructor.

Definition at line 21 of file ServiceAbstract.hpp.

8.145.2.2 stdair::ServiceAbstract::ServiceAbstract () [inline, protected]

Protected Default Constructor to ensure this class is abtract.

Definition at line 46 of file ServiceAbstract.hpp.

8.145.3 Member Function Documentation

8.145.3.1 virtual void stdair::ServiceAbstract::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Definition at line 28 of file ServiceAbstract.hpp.

8.145.3.2 virtual void stdair::ServiceAbstract::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Definition at line 35 of file ServiceAbstract.hpp.

Referenced by operator>>().

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

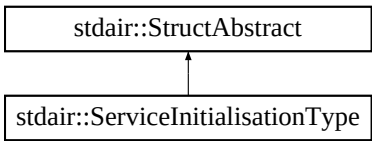
- stdair/service/ServiceAbstract.hpp

8.146 stdair::ServiceInitialisationType Struct Reference

Enumeration of service initialisation types.

`#include <stdair/basic/ServiceInitialisationType.hpp>`

Inheritance diagram for stdair::ServiceInitialisationType::



Public Types

- NOT_YET_INITIALISED = 0
- FILE_PARSING
- BUILTIN_SAMPLE
- LAST_VALUE
- enum EN_ServiceInitialisationType { NOT_YET_INITIALISED = 0, FILE_PARSING, BUILTIN_SAMPLE, LAST_VALUE }

Public Member Functions

- EN_ServiceInitialisationType getType () const
- char getTypeAsChar () const
- std::string getTypeAsString () const
- const std::string describe () const
- bool operator== (const EN_ServiceInitialisationType &) const
- ServiceInitialisationType (const EN_ServiceInitialisationType &)
- ServiceInitialisationType (const char iType)
- ServiceInitialisationType (const std::string &iType)
- ServiceInitialisationType (const ServiceInitialisationType &)
- void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

Static Public Member Functions

- static const std::string & getLabel (const EN_ServiceInitialisationType &)
- static EN_ServiceInitialisationType getType (const char)
- static char getTypeLabel (const EN_ServiceInitialisationType &)
- static std::string getTypeLabelAsString (const EN_ServiceInitialisationType &)
- static std::string describeLabels ()

8.146.1 Detailed Description

Enumeration of service initialisation types.

Definition at line 17 of file ServiceInitialisationType.hpp.

8.146.2 Member Enumeration Documentation

8.146.2.1 enum stdair::ServiceInitialisationType::EN_ServiceInitialisationType

Enumerator:

NOT_YET_INITIALISED
FILE_PARSING
BUILTIN_SAMPLE
LAST_VALUE

Definition at line 19 of file ServiceInitialisationType.hpp.

8.146.3 Constructor & Destructor Documentation

8.146.3.1 stdair::ServiceInitialisationType::ServiceInitialisationType (const EN_ServiceInitialisationType &)

Main constructor.

Definition at line 36 of file ServiceInitialisationType.cpp.

8.146.3.2 stdair::ServiceInitialisationType::ServiceInitialisationType (const char *iType*)

Alternative constructor.

Definition at line 65 of file ServiceInitialisationType.cpp.

8.146.3.3 stdair::ServiceInitialisationType::ServiceInitialisationType (const std::string & *iType*)

Alternative constructor.

Definition at line 71 of file ServiceInitialisationType.cpp.

References getType().

8.146.3.4 stdair::ServiceInitialisationType::ServiceInitialisationType (const ServiceInitialisationType &)

Default copy constructor.

Definition at line 30 of file ServiceInitialisationType.cpp.

8.146.4 Member Function Documentation

8.146.4.1 const std::string & stdair::ServiceInitialisationType::getLabel (const EN_ServiceInitialisationType &) [static]

Get the label as a string (e.g., "Not yet initialised", "File parsing" or "Built-in sample BOM").

Definition at line 81 of file ServiceInitialisationType.cpp.

8.146.4.2 ServiceInitialisationType::EN_ServiceInitialisationType stdair::ServiceInitialisationType::getType (const *char*) [static]

Get the type value from parsing a single char (e.g., 'N', 'F', 'B').

Definition at line 42 of file ServiceInitialisationType.cpp.

References BUILTIN_SAMPLE, describeLabels(), FILE_PARSING, LAST_VALUE, and NOT_YET_INITIALISED.

8.146.4.3 char stdair::ServiceInitialisationType::getTypeLabel (const EN_ServiceInitialisationType &) [static]

Get the label as a single char (e.g., 'N', 'F', 'B').

Definition at line 87 of file ServiceInitialisationType.cpp.

8.146.4.4 std::string stdair::ServiceInitialisationType::getTypeLabelAsString (const EN_ServiceInitialisationType &) [static]

Get the label as a string of a single char (e.g., "N", "F", "B").

Definition at line 93 of file ServiceInitialisationType.cpp.

8.146.4.5 std::string stdair::ServiceInitialisationType::describeLabels () [static]

List the labels.

Definition at line 100 of file ServiceInitialisationType.cpp.

References LAST_VALUE.

Referenced by getType().

8.146.4.6 ServiceInitialisationType::EN_ServiceInitialisationType stdair::ServiceInitialisationType::getType () const

Get the enumerated value.

Definition at line 113 of file ServiceInitialisationType.cpp.

Referenced by ServiceInitialisationType().

8.146.4.7 char stdair::ServiceInitialisationType::getTypeAsChar () const

Get the enumerated value as a short string (e.g., 'N', 'F', 'B').

Definition at line 118 of file ServiceInitialisationType.cpp.

8.146.4.8 std::string stdair::ServiceInitialisationType::getTypeAsString () const

Get the enumerated value as a short string (e.g., "N", "F", "B").

Definition at line 124 of file ServiceInitialisationType.cpp.

8.146.4.9 const std::string stdair::ServiceInitialisationType::describe () const [virtual]

Give a description of the structure (e.g., "Not yet initialised", "File parsing" or "Built-in sample BOM").

Implements stdair::StructAbstract.

Definition at line 131 of file ServiceInitialisationType.cpp.

8.146.4.10 bool stdair::ServiceInitialisationType::operator== (const EN_ServiceInitialisationType &) const

Comparison operator.

Definition at line 139 of file ServiceInitialisationType.cpp.

8.146.4.11 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const [inline, inherited]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References stdair::StructAbstract::describe().

8.146.4.12 virtual void stdair::StructAbstract::fromStream (std::istream & *ioIn*) [inline, virtual, inherited]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented in stdair::ProgressStatusSet, stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::EventStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 38 of file StructAbstract.hpp.

Referenced by operator>>().

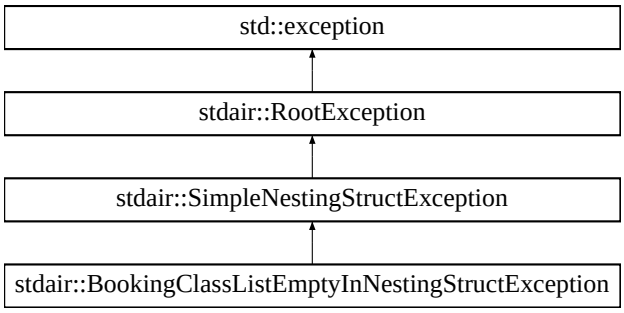
The documentation for this struct was generated from the following files:

- stdair/basic/ServiceInitialisationType.hpp
- stdair/basic/ServiceInitialisationType.cpp

8.147 stdair::SimpleNestingStructException Class Reference

`#include <stdair/stdair_exceptions.hpp>`

Inheritance diagram for stdair::SimpleNestingStructException::



Public Member Functions

- SimpleNestingStructException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

8.147.1 Detailed Description

Simple Nesting Structure.

Definition at line 211 of file stdair_exceptions.hpp.

8.147.2 Constructor & Destructor Documentation

8.147.2.1 stdair::SimpleNestingStructException::SimpleNestingStructException (const std::string & *iWhat*) `[inline]`

Constructor.

Definition at line 214 of file stdair_exceptions.hpp.

8.147.3 Member Function Documentation

8.147.3.1 const char* stdair::RootException::what () const throw () `[inline, inherited]`

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

8.147.4 Member Data Documentation

8.147.4.1 std::string stdair::RootException::_what [protected, inherited]

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

Referenced by stdair::RootException::what().

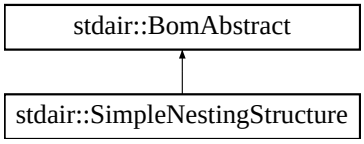
The documentation for this class was generated from the following file:

- stdair/stdair_exceptions.hpp

8.148 stdair::SimpleNestingStructure Class Reference

`#include <stdair/bom/SimpleNestingStructure.hpp>`

Inheritance diagram for stdair::SimpleNestingStructure::



Public Types

- typedef NestingStructureKey Key_T

Public Member Functions

- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const HolderMap_T & getHolderMap () const
- const NestingNodeList_T & getNestingNodeList () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- template<class Archive> void serialize (Archive &ar, const unsigned int iFileVersion)
- SimpleNestingStructure (const Key_T &)
- virtual ~SimpleNestingStructure ()

Friends

- class FacBom
- class FacBomManager
- class boost::serialization::access

8.148.1 Detailed Description

Structure holding a nesting node map according to the yield.

Definition at line 26 of file SimpleNestingStructure.hpp.

8.148.2 Member Typedef Documentation

8.148.2.1 typedef NestingStructureKey stdair::SimpleNestingStructure::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 36 of file SimpleNestingStructure.hpp.

8.148.3 Constructor & Destructor Documentation

8.148.3.1 stdair::SimpleNestingStructure::SimpleNestingStructure (const Key_T &)

Main constructor.

Definition at line 36 of file SimpleNestingStructure.cpp.

8.148.3.2 stdair::SimpleNestingStructure::~SimpleNestingStructure () [virtual]

Destructor.

Definition at line 41 of file SimpleNestingStructure.cpp.

8.148.4 Member Function Documentation

8.148.4.1 const Key_T& stdair::SimpleNestingStructure::getKey () const [inline]

Get the nesting key.

Definition at line 41 of file SimpleNestingStructure.hpp.

8.148.4.2 BomAbstract* const stdair::SimpleNestingStructure::getParent () const [inline]

Get the parent object.

Definition at line 46 of file SimpleNestingStructure.hpp.

8.148.4.3 const HolderMap_T& stdair::SimpleNestingStructure::getHolderMap () const [inline]

Get the map of children holders.

Definition at line 53 of file SimpleNestingStructure.hpp.

8.148.4.4 const NestingNodeList_T & stdair::SimpleNestingStructure::getNestingNodeList () const

Get the nesting node list

Definition at line 115 of file SimpleNestingStructure.cpp.

8.148.4.5 void stdair::SimpleNestingStructure::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 69 of file SimpleNestingStructure.hpp.

References toString().

8.148.4.6 void stdair::SimpleNestingStructure::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 78 of file SimpleNestingStructure.hpp.

8.148.4.7 std::string stdair::SimpleNestingStructure::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 107 of file SimpleNestingStructure.cpp.

References describeKey().

Referenced by toStream().

8.148.4.8 const std::string stdair::SimpleNestingStructure::describeKey () const [inline]

Get a string describing the key.

Definition at line 89 of file SimpleNestingStructure.hpp.

References stdair::NestingStructureKey::toString().

Referenced by toString().

8.148.4.9 template<class Archive> void stdair::SimpleNestingStructure::serialize (Archive & *ar*, const unsigned int *iFileVersion*)

Serialisation.

8.148.5 Friends And Related Function Documentation

8.148.5.1 friend class FacBom [friend]

Definition at line 27 of file SimpleNestingStructure.hpp.

8.148.5.2 friend class FacBomManager [friend]

Definition at line 28 of file SimpleNestingStructure.hpp.

8.148.5.3 friend class boost::serialization::access [friend]

Definition at line 29 of file SimpleNestingStructure.hpp.

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

- stdair/bom/SimpleNestingStructure.hpp
- stdair/bom/SimpleNestingStructure.cpp

8.149 swift::SKeymap Class Reference

The readline keymap wrapper.

`#include <stdair/ui/cmdline/SReadline.hpp>`

Public Member Functions

- SKeymap (bool PrintableBound=false)
  Creates a new keymap.
- SKeymap (Keymap Pattern)
  Creates a new keymap which is a copy of Pattern.
- ~SKeymap ()
  Frees the allocated keymap.
- void Bind (int Key, KeyCallback Callback)
  Binds the given key to a function.
- void Unbind (int Key)
  Unbinds the given key.
- SKeymap (const SKeymap &rhs)
  Copy constructor.
- SKeymap & operator= (const SKeymap &rhs)
  operator=

Friends

- class SReadline

8.149.1 Detailed Description

The readline keymap wrapper.

Attention: It is not thread safe! Supports: key binding, key unbinding

Definition at line 307 of file SReadline.hpp.

8.149.2 Constructor & Destructor Documentation

8.149.2.1 swift::SKeymap::SKeymap (bool *PrintableBound* = `false`) `[inline, explicit]`

Creates a new keymap.

Parameters:

PrintableBound if true - the printable characters are bound if false - the keymap is empty

Definition at line 319 of file SReadline.hpp.

References Keymaps.

8.149.2.2 swift::SKeymap::SKeymap (Keymap *Pattern*) `[inline, explicit]`

Creates a new keymap which is a copy of Pattern.

Parameters:

Pattern A keymap to be copied.

Definition at line 342 of file SReadline.hpp.

References Keymaps.

8.149.2.3 swift::SKeymap::~SKeymap () `[inline]`

Frees the allocated keymap.

Definition at line 354 of file SReadline.hpp.

References Keymaps.

8.149.2.4 swift::SKeymap::SKeymap (const SKeymap & *rhs*) `[inline]`

Copy constructor.

Parameters:

rhs Right hand side object of SKeymap

Definition at line 395 of file SReadline.hpp.

References keymap.

8.149.3 Member Function Documentation

8.149.3.1 void swift::SKeymap::Bind (int *Key*, KeyCallback *Callback*) [inline]

Binds the given key to a function.

Parameters:

Key A key to be bound

Callback A function to be called when the Key is pressed

Definition at line 366 of file SReadline.hpp.

References KeyDispatcher(), and Keymaps.

8.149.3.2 void swift::SKeymap::Unbind (int *Key*) [inline]

Unbinds the given key.

Parameters:

Key A key to be unbound

Definition at line 381 of file SReadline.hpp.

References Keymaps.

8.149.3.3 SKeymap& swift::SKeymap::operator= (const SKeymap & *rhs*) [inline]

operator=

Parameters:

rhs Right hand side object of SKeymap

Definition at line 407 of file SReadline.hpp.

References keymap.

8.149.4 Friends And Related Function Documentation

8.149.4.1 friend class SReadline [friend]

Definition at line 415 of file SReadline.hpp.

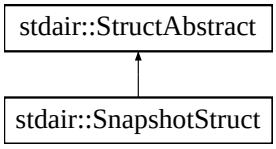
The documentation for this class was generated from the following file:

- stdair/ui/cmdline/SReadline.hpp

8.150 stdair::SnapshotStruct Struct Reference

`#include <stdair/bom/SnapshotStruct.hpp>`

Inheritance diagram for stdair::SnapshotStruct::



Public Member Functions

- const AirlineCode_T & getAirlineCode () const
- const DateTime_T & getSnapshotTime () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string describe () const
- SnapshotStruct (const AirlineCode_T &, const DateTime_T &)
- SnapshotStruct (const SnapshotStruct &)
- ~SnapshotStruct ()

8.150.1 Detailed Description

Structure holding the elements of a snapshot .

Definition at line 19 of file SnapshotStruct.hpp.

8.150.2 Constructor & Destructor Documentation

8.150.2.1 stdair::SnapshotStruct::SnapshotStruct (const AirlineCode_T &, const DateTime_T &)

Constructor.

Definition at line 26 of file SnapshotStruct.cpp.

8.150.2.2 stdair::SnapshotStruct::SnapshotStruct (const SnapshotStruct &)

Copy constructor.

Definition at line 19 of file SnapshotStruct.cpp.

8.150.2.3 stdair::SnapshotStruct::~SnapshotStruct ()

Destructor.

Definition at line 32 of file SnapshotStruct.cpp.

8.150.3 Member Function Documentation

8.150.3.1 const AirlineCode_T& stdair::SnapshotStruct::getAirlineCode () const [inline]

Get the airline code.

Definition at line 23 of file SnapshotStruct.hpp.

8.150.3.2 const DateTime_T& stdair::SnapshotStruct::getSnapshotTime () const [inline]

Get the snapshot action time.

Definition at line 28 of file SnapshotStruct.hpp.

8.150.3.3 void stdair::SnapshotStruct::toStream (std::ostream & *ioOut*) const

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::StructAbstract.

Definition at line 36 of file SnapshotStruct.cpp.

References describe().

8.150.3.4 void stdair::SnapshotStruct::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::StructAbstract.

Definition at line 41 of file SnapshotStruct.cpp.

8.150.3.5 const std::string stdair::SnapshotStruct::describe () const [virtual]

Display of the structure.

Implements stdair::StructAbstract.

Definition at line 45 of file SnapshotStruct.cpp.

Referenced by toStream().

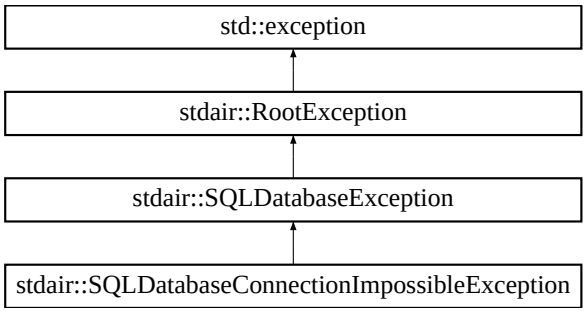
The documentation for this struct was generated from the following files:

- stdair/bom/SnapshotStruct.hpp
- stdair/bom/SnapshotStruct.cpp

8.151 stdair::SQLDatabaseConnectionImpossibleException Class Reference

`#include <stdair/stdair_exceptions.hpp>`

Inheritance diagram for stdair::SQLDatabaseConnectionImpossibleException::



Public Member Functions

- SQLDatabaseConnectionImpossibleException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

8.151.1 Detailed Description

Database connection.

Definition at line 196 of file stdair_exceptions.hpp.

8.151.2 Constructor & Destructor Documentation

8.151.2.1 stdair::SQLDatabaseConnectionImpossibleException::SQLDatabaseConnectionImpossibleException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 199 of file stdair_exceptions.hpp.

8.151.3 Member Function Documentation

8.151.3.1 const char* stdair::RootException::what () const throw () [inline, inherited]

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

8.151.4 Member Data Documentation

8.151.4.1 std::string stdair::RootException::_what [protected, inherited]

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

Referenced by stdair::RootException::what().

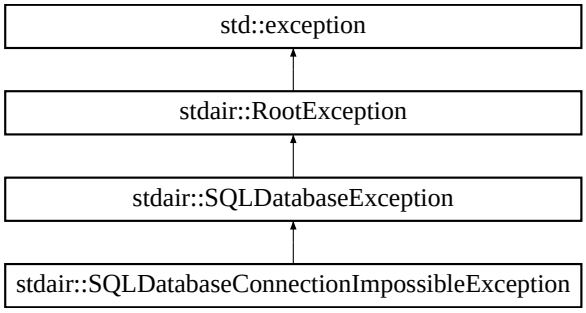
The documentation for this class was generated from the following file:

- stdair/stdair_exceptions.hpp

8.152 stdair::SQLDatabaseException Class Reference

`#include <stdair/stdair_exceptions.hpp>`

Inheritance diagram for stdair::SQLDatabaseException::



Public Member Functions

- SQLDatabaseException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

8.152.1 Detailed Description

Database.

Definition at line 181 of file stdair_exceptions.hpp.

8.152.2 Constructor & Destructor Documentation

8.152.2.1 stdair::SQLDatabaseException::SQLDatabaseException (const std::string & *iWhat*) `[inline]`

Constructor.

Definition at line 184 of file stdair_exceptions.hpp.

8.152.3 Member Function Documentation

8.152.3.1 const char* stdair::RootException::what () const throw () `[inline, inherited]`

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

8.152.4 Member Data Documentation

8.152.4.1 std::string stdair::RootException::_what [protected, inherited]

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

Referenced by stdair::RootException::what().

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

- stdair/stdair_exceptions.hpp

8.153 swift::SReadline Class Reference

The readline library wrapper.

`#include <stdair/ui/cmdline/SReadline.hpp>`

Public Member Functions

- SReadline (const size_t Limit=DefaultHistoryLimit)
  Constructs the object, sets the completion function.
- SReadline (const std::string &historyFileName, const size_t Limit=DefaultHistoryLimit)
  Constructs the object, sets the completion function, loads history.
- ~SReadline ()
  Saves the session history (if the file name was provided) and destroys the object.
- std::string GetLine (const std::string &Prompt)
  Gets a single line from a user.
- template<typename Container> std::string GetLine (const std::string &Prompt, Container &ReadTokens)
  Gets a single line from a user.
- template<typename Container> std::string GetLine (const std::string &Prompt, Container &ReadTokens, bool &BreakOut)
  Gets a single line from a user.
- std::string GetLine (const std::string &Prompt, bool &BreakOut)
  Gets a single line from a user.
- template<typename ContainerType> void GetHistory (ContainerType &Container)
  Fills the given container with the current history list.

- bool SaveHistory (std::ostream &OS)
  Saves the history to the given file stream.
- bool SaveHistory (const std::string &FileName)
  Saves the history to the given file.
- void ClearHistory ()
  Clears the history. Does not affect the file where the previous session history is saved.
- bool LoadHistory (std::istream &IS)
  Loads a history from a file stream.
- bool LoadHistory (const std::string &FileName)
  Loads a history from the given file.
- template<typename ContainerType> void RegisterCompletions (const ContainerType &Container)
  Allows to register custom completers.
- void SetKeymap (SKeymap &NewKeymap)
  Sets the given keymap.

8.153.1 Detailed Description

The readline library wrapper.

Attention: It is not thread safe! Supports: editing, history, custom completers

Definition at line 424 of file SReadline.hpp.

8.153.2 Constructor & Destructor Documentation

8.153.2.1 swift::SReadline::SReadline (const size_t *Limit* = `DefaultHistoryLimit`) `[inline]`

Constructs the object, sets the completion function.

Parameters:

Limit History size

Definition at line 431 of file SReadline.hpp.

References StartupHook(), and UserCompletion().

8.153.2.2 swift::SReadline::SReadline (const std::string & *historyFileName*, const size_t *Limit* = `DefaultHistoryLimit`) `[inline]`

Constructs the object, sets the completion function, loads history.

Parameters:

historyFileName File name to load history from

Limit History size

Definition at line 446 of file SReadline.hpp.

References LoadHistory(), StartupHook(), and UserCompletion().

8.153.2.3 swift::SReadline::~SReadline () [inline]

Saves the session history (if the file name was provided) and destroys the object.

Definition at line 462 of file SReadline.hpp.

References SaveHistory().

8.153.3 Member Function Documentation

8.153.3.1 std::string swift::SReadline::GetLine (const std::string & *Prompt*) [inline]

Gets a single line from a user.

Parameters:

Prompt A printed prompt

Returns:

A string which was actually inputed

Definition at line 473 of file SReadline.hpp.

Referenced by GetLine().

8.153.3.2 template<typename Container> std::string swift::SReadline::GetLine (const std::string & *Prompt*, Container & *ReadTokens*) [inline]

Gets a single line from a user.

Parameters:

Prompt A printed prompt

ReadTokens A user inputed string splitted into tokens. The container is cleared first

Returns:

A string which was actually inputed

Definition at line 487 of file SReadline.hpp.

References GetLine().

8.153.3.3 template<typename Container> std::string swift::SReadline::GetLine (const std::string & *Prompt*, Container & *ReadTokens*, bool & *BreakOut*) [inline]

Gets a single line from a user.

Parameters:

Prompt A printed prompt

BreakOut it is set to true if the EOF found

ReadTokens A user inputed string splitted into tokens. The container is cleared first

Returns:

A string which was actually inputed

Definition at line 502 of file SReadline.hpp.

References GetLine(), and SplitTokens().

8.153.3.4 std::string swift::SReadline::GetLine (const std::string & *Prompt*, bool & *BreakOut*) [inline]

Gets a single line from a user.

Parameters:

Prompt A printed prompt

BreakOut it is set to true if the EOF found

Returns:

A string which was actually inputed

Definition at line 517 of file SReadline.hpp.

8.153.3.5 template<typename ContainerType> void swift::SReadline::GetHistory (ContainerType & *Container*) [inline]

Fills the given container with the current history list.

Does not clear the given container

Definition at line 552 of file SReadline.hpp.

8.153.3.6 bool swift::SReadline::SaveHistory (std::ostream & *OS*) [inline]

Saves the history to the given file stream.

Parameters:

OS output file stream

Returns:

true if success

Definition at line 564 of file SReadline.hpp.

Referenced by SaveHistory(), and ~SReadline().

8.153.3.7 bool swift::SReadline::SaveHistory (const std::string & *FileName*) [inline]

Saves the history to the given file.

Parameters:

FileName File name to save the history to

Returns:

true if success

Definition at line 581 of file SReadline.hpp.

References SaveHistory().

8.153.3.8 void swift::SReadline::ClearHistory () [inline]

Clears the history. Does not affect the file where the previous session history is saved.

Definition at line 594 of file SReadline.hpp.

Referenced by LoadHistory().

8.153.3.9 bool swift::SReadline::LoadHistory (std::istream & *IS*) [inline]

Loads a history from a file stream.

Parameters:

IS Input file stream

Returns:

true if success

Definition at line 604 of file SReadline.hpp.

References ClearHistory().

Referenced by LoadHistory(), and SReadline().

8.153.3.10 bool swift::SReadline::LoadHistory (const std::string & *FileName*) [inline]

Loads a history from the given file.

Parameters:

FileName File name to be load from

Returns:

true if success

Definition at line 629 of file SReadline.hpp.

References LoadHistory().

8.153.3.11 template<typename ContainerType> void swift::SReadline::RegisterCompletions (const ContainerType & *Container*) [inline]

Allows to register custom completers.

Supports a special keyword: file. It means to use the standard file name completer.

For example the given container elements could be as follows:

- command1 opt1
- command1 opt2 file
- command2
- command2 opt1

Each container element must describe a single possible command line. The container element must have a conversion to std::string operator.

Parameters:

Container A container which has all the user possible commands.

Definition at line 658 of file SReadline.hpp.

References Completions, and SplitTokens().

8.153.3.12 void swift::SReadline::SetKeymap (SKeymap & *NewKeymap*) [inline]

Sets the given keymap.

Parameters:

NewKeymap The keymap that should be used from now.

Definition at line 675 of file SReadline.hpp.

References Earlykeymap(), swift::SKeymap::keymap, and KeymapWasSetup().

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

- stdair/ui/cmdline/SReadline.hpp

8.154 stdair::STDAIR_Service Class Reference

Interface for the STDAIR Services.

`#include <stdair/STDAIR_Service.hpp>`

Public Member Functions

- STDAIR_Service ()

  Default constructor.

- STDAIR_Service (const BasLogParams &)

Constructor.

- STDAIR_Service (const BasLogParams &, const BasDBParams &)

  Constructor.

- ~STDAIR_Service ()

  Destructor.

- void buildSampleBom ()
- void buildDummyInventory (const CabinCapacity_T &iCabinCapacity)
- void buildDummyLegSegmentAccesses (BomRoot &)
- void buildSampleTravelSolutionForPricing (TravelSolutionList_T &)
- void buildSampleTravelSolutions (TravelSolutionList_T &)
- BookingRequestStruct buildSampleBookingRequest (const bool isForCRS=false)
- void clonePersistentBom ()

  Clone the persistent Bom.

- std::string jsonExportFlightDateList (const AirlineCode_T &iAirlineCode="all", const FlightNumber_T &iFlightNumber=0) const
- std::string jsonExportFlightDateObjects (const AirlineCode_T &, const FlightNumber_T &, const Date_T &iDepartureDate) const
- std::string jsonExportEventObject (const EventStruct &) const
- std::string jsonExportConfiguration () const
- bool jsonImportConfiguration (const JSONString &) const
- std::string list (const AirlineCode_T &iAirlineCode="all", const FlightNumber_T &iFlightNumber=0) const
- std::string listAirportPairDateRange () const
- bool check (const AirlineCode_T &, const FlightNumber_T &, const Date_T &iDepartureDate) const
- bool check (const AirportCode_T &, const AirportCode_T &, const Date_T &iDepartureDate) const
- std::string configDisplay () const
- std::string csvDisplay () const
- std::string csvDisplay (const BomRoot &) const
- std::string csvDisplay (const AirlineCode_T &, const FlightNumber_T &, const Date_T &iDepartureDate) const
- std::string csvDisplay (const TravelSolutionList_T &) const
- std::string csvDisplay (const AirportCode_T &, const AirportCode_T &, const Date_T &iDepartureDate) const
- BomRoot & getBomRoot () const

  Get a reference on the BomRoot object.

- BomRoot & getPersistentBomRoot () const

  Get a reference on the BomRoot object.

- BasLogParams getLogParams () const
- const BasDBParams & getDBParams () const
- const ServiceInitialisationType & getServiceInitialisationType () const
- void importINIConfig (const ConfigINIFile &)

  Import the configuration INI input file (format cfg).

- void importConfigValue (const std::string &iValue, const std::string &iPath)

- template<typename ValueType> bool exportConfigValue (ValueType &ioValue, const std::string &iPath)
- void updateAirlineFeatures ()

  Update the airline features objects thanks to the configuration holder.

8.154.1 Detailed Description

Interface for the STDAIR Services.

Definition at line 44 of file STDAIR_Service.hpp.

8.154.2 Constructor & Destructor Documentation

8.154.2.1 stdair::STDAIR_Service::STDAIR_Service ()

Default constructor.

Definition at line 45 of file STDAIR_Service.cpp.

8.154.2.2 stdair::STDAIR_Service::STDAIR_Service (const BasLogParams &)

Constructor.

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

Moreover, a reference on an output stream is given, so that log outputs can be directed onto that stream.

Parameters:

← ***const*** BasLogParams& Parameters for the output log stream.

Definition at line 61 of file STDAIR_Service.cpp.

8.154.2.3 stdair::STDAIR_Service::STDAIR_Service (const BasLogParams &, const BasDBParams &)

Constructor.

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

A reference on an output stream is given, so that log outputs can be directed onto that stream.

Moreover, database connection parameters are given, so that database events can use the corresponding access.

Parameters:

← ***const*** BasLogParams& Parameters for the output log stream.
← ***const*** BasDBParams& Parameters for the database session.

Definition at line 75 of file STDAIR_Service.cpp.

8.154.2.4 stdair::STDAIR_Service::~STDAIR_Service ()

Destructor.

Definition at line 93 of file STDAIR_Service.cpp.

8.154.3 Member Function Documentation

8.154.3.1 void stdair::STDAIR_Service::buildSampleBom ()

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

As for now, a single sample BOM tree is built, with objects for all the simulator-related components, i.e.:

- schedule (e.g., AirSched),
- inventory (e.g., AirInv),
- revenue management (e.g., RMOL),
- pricing (e.g., SimFQT),
- revenue accounting (e.g., AirRAC),
- demand generation (e.g., TraDemGen),
- customer choice (e.g., TravelCCM),
- event manager (e.g., SEvMgr)

Most of the inventories just contain one flight. One of those flights has two legs (and therefore three segments).

Definition at line 172 of file STDAIR_Service.cpp.

References stdair::STDAIR_ServiceContext::getPersistentBomRoot().

8.154.3.2 void stdair::STDAIR_Service::buildDummyInventory (const CabinCapacity_T & *iCabinCapacity*)

Build a dummy inventory, containing a dummy flight-date with a single leg-cabin and some virtual booking classes. That structure is the bare minimum required to perform an optimisation on a leg-cabin.

As for now, that method is called only by RMOL. Indeed, the revenue management component (RMOL) needs very basic set up in order to perform optimisation at leg-level. Hence, there are:

- a dedicated inventory ('XX'),
- the corresponding flight-date (#9999, departing 01/01/1900),
- a leg-date (departing and arriving from/to 'XXX' airport),
-
- a leg-cabin ('X').
-

Most of the data is dummy because RMOL uses only the cabin capacity from that part of the BOM tree.

Parameters:

const CabinCapacity_T& Cabin capacity for revenue management optimisation.

Definition at line 187 of file STDAIR_Service.cpp.

References stdair::STDAIR_ServiceContext::getPersistentBomRoot().

8.154.3.3 void stdair::STDAIR_Service::buildDummyLegSegmentAccesses (BomRoot &)

Build the direct accesses between the dummy segment cabins and the dummy leg cabins within the dummy flight dates (the dummy fare family flight date and the classic dummy flight date).

As for now (May 2012), that method is called only by RMOL. It is a substitute for the code doing it automatically located in AirInv. See the AIRINV::InventoryManager::createDirectAccesses command.

Parameters:

BomRoot& Top of the BOM tree, to which the sample should be attached.

Definition at line 204 of file STDAIR_Service.cpp.

8.154.3.4 void stdair::STDAIR_Service::buildSampleTravelSolutionForPricing (TravelSolutionList_T &)

Build a sample list of travel solutions.

As of now (March 2011), that list is made of the following travel solutions:

- BA9
- LHR-SYD
- 2011-06-10

Parameters:

TravelSolutionList_T& Sample list of travel solution structures. It should be given empty. It is altered with the returned sample.

Definition at line 215 of file STDAIR_Service.cpp.

8.154.3.5 void stdair::STDAIR_Service::buildSampleTravelSolutions (TravelSolutionList_T &)

Build a sample list of travel solutions.

As of now (March 2011), that list is made of the following travel solutions:

- BA9
- LHR-SYD
- 2011-06-10
- Q
- WTP: 900
- Change fee: 20; Non refundable; Saturday night stay

Parameters:

TravelSolutionList_T& Sample list of travel solution structures. It should be given empty. It is altered with the returned sample.

Definition at line 222 of file STDAIR_Service.cpp.

8.154.3.6 BookingRequestStruct stdair::STDAIR_Service::buildSampleBookingRequest (const bool *isForCRS* = `false`)

Build a sample booking request structure.

As of now (March 2011), the sample booking request is made of the following parameters:

- Return trip (inbound): LHR-SYD (POS: LHR, Channel: DN),
- Departing 10-JUN-2011 around 8:00, staying 7 days
- Requested on 15-MAY-2011 at 10:00
- Economy cabin, 3 persons, FF member
- WTP: 1000.0 EUR
- Dis-utility: 100.0 EUR/hour

As of now (March 2011), the CRS-related booking request is made of the following parameters:

- Return trip (inbound): SIN-BKK (POS: SIN, Channel: IN),
- Departing 30-JAN-2010 around 10:00, staying 7 days
- Requested on 22-JAN-2010 at 10:00
- Economy cabin, 3 persons, FF member
- WTP: 1000.0 EUR
- Dis-utility: 100.0 EUR/hour

Parameters:

const bool isForCRS Whether the sample booking request is for CRS.

Returns:

BookingRequestStruct& Sample booking request structure.

Definition at line 229 of file STDAIR_Service.cpp.

8.154.3.7 void stdair::STDAIR_Service::clonePersistentBom ()

Clone the persistent Bom.

Definition at line 635 of file STDAIR_Service.cpp.

References stdair::FacSupervisor::cleanCloneBomLayer(), stdair::STDAIR_ServiceContext::getCloneBomRoot(), stdair::STDAIR_ServiceContext::getPersistentBomRoot(), stdair::STDAIR_ServiceContext::initCloneBomRoot(), and stdair::FacSupervisor::instance().

8.154.3.8 std::string stdair::STDAIR_Service::jsonExportFlightDateList (const AirlineCode_T & *iAirlineCode* = `"all"`, const FlightNumber_T & *iFlightNumber* = `0`) const

Recursively dump, in the returned string and in JSON format, the flight-date list corresponding to the parameters given as input.

Parameters:

const AirlineCode& Airline for which the flight-dates should be displayed. If set to "all" (default), all the inventories will be displayed.

const FlightNumber_T& Flight number for which all the departure dates should be displayed. If set to 0 (the default), all the flight numbers will be displayed.

Definition at line 242 of file STDAIR_Service.cpp.

References stdair::STDAIR_ServiceContext::getCloneBomRoot(), and stdair::BomJSONExport::jsonExportFlightDateList().

8.154.3.9 std::string stdair::STDAIR_Service::jsonExportFlightDateObjects (const AirlineCode_T &, const FlightNumber_T &, const Date_T & *iDepartureDate*) const

Recursively dump, in the returned string and in JSON format, the detailed flight-date (leg, segments, cabins, classes, ...) corresponding to the parameters given as input.

Parameters:

const AirlineCode_T& Airline code of the flight to dump.

const FlightNumber_T& Flight number of the flight to dump.

const Date_T& Departure date of the flight to dump.

Returns:

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

Definition at line 262 of file STDAIR_Service.cpp.

References stdair::STDAIR_ServiceContext::getCloneBomRoot(), stdair::BomJSONExport::jsonExportFlightDateObjects(), and stdair::BomRetriever::retrieveFlightDateFromKeySet().

8.154.3.10 std::string stdair::STDAIR_Service::jsonExportEventObject (const EventStruct &) const

Recursively dump, in the returned string and in JSON format, the event object.

Returns:

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

Definition at line 312 of file STDAIR_Service.cpp.

References stdair::EventType::BKG_REQ, stdair::EventType::BRK_PT, stdair::EventType::CX, stdair::EventStruct::getEventType(), stdair::BomJSONExport::jsonExportBookingRequestObject(), stdair::BomJSONExport::jsonExportBreakPointObject(), stdair::EventType::OPT_NOT_4_FD, stdair::EventType::OPT_NOT_4_NET, stdair::EventType::RM, stdair::EventType::SKD_CHG, and stdair::EventType::SNAPSHOT.

8.154.3.11 std::string stdair::STDAIR_Service::jsonExportConfiguration () const

Dump, in the returned string and in JSON format, the configuration.

Returns:

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

Definition at line 359 of file STDAIR_Service.cpp.

References stdair::STDAIR_ServiceContext::getConfigHolder(), and stdair::ConfigHolderStruct::jsonExport().

8.154.3.12 bool stdair::STDAIR_Service::jsonImportConfiguration (const JSONString &) const

Extract the configuration ptree from the given JSON-formatted string and add it to the configuration holder

Parameters:

const JSONString& JSON-formatted string.

Returns:

bool State whether the extracting has been successful.

Definition at line 342 of file STDAIR_Service.cpp.

References stdair::STDAIR_ServiceContext::getConfigHolder(), and stdair::BomJSONImport::jsonImportConfig().

8.154.3.13 std::string stdair::STDAIR_Service::list (const AirlineCode_T & *iAirlineCode* = "all", const FlightNumber_T & *iFlightNumber* = 0) const

Display the list of flight-dates (contained within the BOM tree) corresponding to the parameters given as input.

Parameters:

const AirlineCode& Airline for which the flight-dates should be displayed. If set to "all" (the default), all the inventories will be displayed.

const FlightNumber_T& Flight number for which all the departure dates should be displayed. If set to 0 (the default), all the flight numbers will be displayed.

Returns:

std::string Output string in which the BOM tree is logged/dumped.

Definition at line 428 of file STDAIR_Service.cpp.

References stdair::STDAIR_ServiceContext::getCloneBomRoot(), and stdair::BomDisplay::list().

8.154.3.14 std::string stdair::STDAIR_Service::listAirportPairDateRange () const

Display the list of aiports pairs and date ranges (contained within the BOM tree)

Parameters:

std::ostream& Output stream in which the airport pairs and date ranges are logged/dumped.

Definition at line 446 of file STDAIR_Service.cpp.

References stdair::STDAIR_ServiceContext::getCloneBomRoot(), and stdair::BomDisplay::listAirportPairDateRange().

8.154.3.15 bool stdair::STDAIR_Service::check (const AirlineCode_T &, const FlightNumber_T &, const Date_T & *iDepartureDate*) const

Check whether the given flight-date is a valid one.

Parameters:

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

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

const stdair::Date_T& Departure date of the flight to check.

Returns:

bool Whether or not the given flight date is valid.

Definition at line 463 of file STDAIR_Service.cpp.

References stdair::STDAIR_ServiceContext::getCloneBomRoot(), and stdair::BomRetriever::retrieveFlightDateFromKeySet().

8.154.3.16 bool stdair::STDAIR_Service::check (const AirportCode_T &, const AirportCode_T &, const Date_T & *iDepartureDate*) const

Check whether the given couple airportpair-date is a valid one.

Parameters:

const stdair::AirportCode_T& Origin airport of the fare rule to check.

const stdair::AirportCode_T& Destination airport of the fare rule to check.

const stdair::Date_T& Departure date of the fare rule to check.

Returns:

bool Whether or not the given airportpair-date couple is a valid one.

Definition at line 485 of file STDAIR_Service.cpp.

References stdair::STDAIR_ServiceContext::getCloneBomRoot(), and stdair::BomRetriever::retrieveDatePeriodListFromKeySet().

8.154.3.17 std::string stdair::STDAIR_Service::configDisplay () const

Display (dump in the returned string) the configuration.

Returns:

std::string Output string in which the configuration is logged/dumped.

Definition at line 508 of file STDAIR_Service.cpp.

References stdair::ConfigHolderStruct::describe(), and stdair::STDAIR_ServiceContext::getConfigHolder().

8.154.3.18 std::string stdair::STDAIR_Service::csvDisplay () const

Recursively display (dump in the returned string) the objects of the persistent BOM tree.

Returns:

std::string Output string in which the persistent BOM tree is logged/dumped.

Definition at line 525 of file STDAIR_Service.cpp.

References stdair::STDAIR_ServiceContext::getPersistentBomRoot().

8.154.3.19 std::string stdair::STDAIR_Service::csvDisplay (const BomRoot &) const

Recursively display (dump in the returned string) the objects of the BOM tree.

Parameters:

const BomRoot& Reference on the BomRoot to display.

Returns:

std::string Output string in which the BOM tree is logged/dumped.

Definition at line 541 of file STDAIR_Service.cpp.

References stdair::BomDisplay::csvDisplay().

8.154.3.20 std::string stdair::STDAIR_Service::csvDisplay (const AirlineCode_T &, const FlightNumber_T &, const Date_T & *iDepartureDate*) const

Recursively display (dump in the returned string) the flight-date corresponding to the parameters given as input.

Parameters:

const AirlineCode_T& Airline code of the flight to display.
const FlightNumber_T& Flight number of the flight to display.
const Date_T& Departure date of the flight to display.

Returns:

std::string Output string in which the BOM tree is logged/dumped.

Definition at line 555 of file STDAIR_Service.cpp.

References stdair::BomDisplay::csvDisplay(), stdair::STDAIR_ServiceContext::getCloneBomRoot(), and stdair::BomRetriever::retrieveFlightDateFromKeySet().

8.154.3.21 std::string stdair::STDAIR_Service::csvDisplay (const TravelSolutionList_T &) const

Display (dump in the returned string) the full list of travel solution structures.

Returns:

std::string Output string in which the list of travel solutions is logged/dumped.

Definition at line 587 of file STDAIR_Service.cpp.

References stdair::BomDisplay::csvDisplay().

8.154.3.22 std::string stdair::STDAIR_Service::csvDisplay (const AirportCode_T &, const AirportCode_T &, const Date_T & *iDepartureDate*) const

Recursively display (dump in the returned string) the fare-rules corresponding to the parameters given as input.

Parameters:

const AirportCode_T& Origin airport of the fare-rules to display

const AirportCode_T& Destination airport of the fare-rules to display.

const Date_T& Departure date of the fare-rules to display.

Returns:

std::string Output string in which the BOM tree is logged/dumped.

Definition at line 598 of file STDAIR_Service.cpp.

References stdair::BomDisplay::csvDisplay(), stdair::STDAIR_ServiceContext::getCloneBomRoot(), and stdair::BomRetriever::retrieveDatePeriodListFromKeySet().

8.154.3.23 BomRoot & stdair::STDAIR_Service::getBomRoot () const

Get a reference on the BomRoot object.

If the service context has not been initialised, that method throws an exception (failing assertion).

Returns:

BomRoot& Reference on the BomRoot.

Definition at line 128 of file STDAIR_Service.cpp.

References stdair::STDAIR_ServiceContext::getCloneBomRoot().

8.154.3.24 BomRoot & stdair::STDAIR_Service::getPersistentBomRoot () const

Get a reference on the BomRoot object.

If the service context has not been initialised, that method throws an exception (failing assertion).

Returns:

BomRoot& Reference on the BomRoot.

Definition at line 138 of file STDAIR_Service.cpp.

References stdair::STDAIR_ServiceContext::getPersistentBomRoot().

8.154.3.25 BasLogParams stdair::STDAIR_Service::getLogParams () const

Get the log parameters.

Returns:

BasLogParams Copy of the structure holding the log parameters.

Definition at line 148 of file STDAIR_Service.cpp.

8.154.3.26 const BasDBParams & stdair::STDAIR_Service::getDBParams () const

Get the database parameters.

Returns:

const BasDBParams& Reference on the structure holding the database parameters.

Definition at line 153 of file STDAIR_Service.cpp.

References stdair::STDAIR_ServiceContext::getDBParams().

8.154.3.27 const ServiceInitialisationType & stdair::STDAIR_Service::getServiceInitialisationType () const

Get the type of initialisation (e.g., not yet, file parsing, sample BOM) which the component (owner of the current STDAIR_Service instance) has gone through.

Returns:

const ServiceInitialisationType& Reference on the type of initialisation (enumeration structure).

Definition at line 163 of file STDAIR_Service.cpp.

References stdair::STDAIR_ServiceContext::getServiceInitialisationType().

8.154.3.28 void stdair::STDAIR_Service::importINIConfig (const ConfigINIFile &)

Import the configuration INI input file (format cfg).

Parameters:

const ConfigINIFile& INI input file.

Definition at line 375 of file STDAIR_Service.cpp.

References stdair::STDAIR_ServiceContext::getConfigHolder(), and stdair::BomINIImport::importINIConfig().

8.154.3.29 void stdair::STDAIR_Service::importConfigValue (const std::string & *iValue*, const std::string & *iPath*)

Create the given specified path in the configuration tree and add the corresponding given value (or replace the value if the path already exists).

Parameters:

const std::string& Value to add in the configuration tree.

const std::string& Path to create (or to look for).

Definition at line 391 of file STDAIR_Service.cpp.

References stdair::ConfigHolderStruct::addValue(), and stdair::STDAIR_ServiceContext::getConfigHolder().

8.154.3.30 template<typename ValueType> bool stdair::STDAIR_Service::exportConfigValue (ValueType & *ioValue*, const std::string & *iPath*)

Look for the specified path in the configuration tree and, if existing, try to extract the corresponding value. The type of the value to extract is a template parameter.

Parameters:

ValueType& Value to add in the configuration tree.

const std::string& Path to look for.

Definition at line 552 of file STDAIR_Service.hpp.

References stdair::ConfigHolderStruct::exportValue(), and stdair::STDAIR_ServiceContext::getConfigHolder().

8.154.3.31 void stdair::STDAIR_Service::updateAirlineFeatures ()

Update the airline features objects thanks to the configuration holder.

Definition at line 408 of file STDAIR_Service.cpp.

References stdair::STDAIR_ServiceContext::getConfigHolder(), stdair::STDAIR_ServiceContext::getPersistentBomRoot(), and stdair::ConfigHolderStruct::updateAirlineFeatures().

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

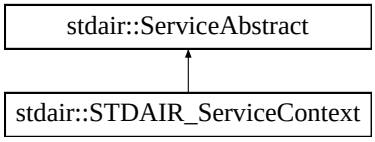
- stdair/STDAIR_Service.hpp
- stdair/service/STDAIR_Service.cpp

8.155 stdair::STDAIR_ServiceContext Class Reference

Class holding the context of the Stdair services.

`#include <stdair/service/STDAIR_ServiceContext.hpp>`

Inheritance diagram for stdair::STDAIR_ServiceContext::



Public Member Functions

- virtual void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

Friends

- class STDAIR_Service
- class FacSTDAIRServiceContext

8.155.1 Detailed Description

Class holding the context of the Stdair services.

Definition at line 25 of file STDAIR_ServiceContext.hpp.

8.155.2 Member Function Documentation

8.155.2.1 virtual void stdair::ServiceAbstract::toStream (std::ostream & *ioOut*) const [inline, virtual, inherited]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Definition at line 28 of file ServiceAbstract.hpp.

8.155.2.2 virtual void stdair::ServiceAbstract::fromStream (std::istream & *ioIn*) [inline, virtual, inherited]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Definition at line 35 of file ServiceAbstract.hpp.

Referenced by operator>>().

8.155.3 Friends And Related Function Documentation

8.155.3.1 friend class STDAIR_Service [friend]

The STDAIR_Service class should be the sole class to get access to ServiceContext content: general users do not want to bother with a context interface.

Definition at line 29 of file STDAIR_ServiceContext.hpp.

8.155.3.2 friend class FacSTDAIRServiceContext [friend]

Definition at line 30 of file STDAIR_ServiceContext.hpp.

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

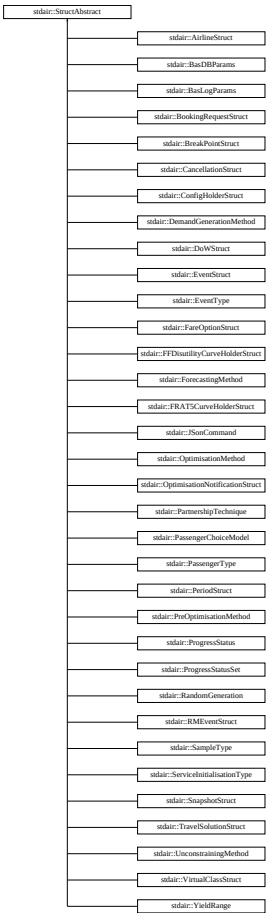
- stdair/service/STDAIR_ServiceContext.hpp

8.156 stdair::StructAbstract Struct Reference

Base class for the light structures.

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

Inheritance diagram for stdair::StructAbstract::



Public Member Functions

- virtual ~StructAbstract ()
- void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)
- virtual const std::string describe () const =0

Protected Member Functions

- StructAbstract ()

8.156.1 Detailed Description

Base class for the light structures.

Definition at line 16 of file StructAbstract.hpp.

8.156.2 Constructor & Destructor Documentation

8.156.2.1 virtual stdair::StructAbstract::~StructAbstract () [inline, virtual]

Destructor.

Definition at line 22 of file StructAbstract.hpp.

8.156.2.2 stdair::StructAbstract::StructAbstract () [inline, protected]

Protected Default Constructor to ensure this class is abtract.

Definition at line 49 of file StructAbstract.hpp.

8.156.3 Member Function Documentation

8.156.3.1 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const [inline]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References describe().

8.156.3.2 virtual void stdair::StructAbstract::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented in stdair::ProgressStatusSet, stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::EventStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 38 of file StructAbstract.hpp.

Referenced by operator>>().

8.156.3.3 virtual const std::string stdair::StructAbstract::describe () const [pure virtual]

Display of the structure.

Implemented in stdair::BasDBParams, stdair::BasLogParams, stdair::DemandGenerationMethod, stdair::EventType, stdair::ForecastingMethod, stdair::JSonCommand, stdair::OptimisationMethod, stdair::PartnershipTechnique, stdair::PassengerChoiceModel, stdair::PassengerType, stdair::PreOptimisationMethod, stdair::ProgressStatus, stdair::ProgressStatusSet, stdair::RandomGeneration, stdair::SampleType, stdair::ServiceInitialisationType, stdair::UnconstrainingMethod, stdair::YieldRange,

stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::DoWStruct, stdair::EventStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::PeriodStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Referenced by toStream().

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

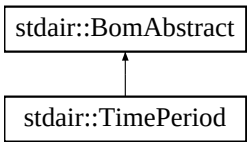
- stdair/basic/StructAbstract.hpp

8.157 stdair::TimePeriod Class Reference

Class representing the actual attributes for a fare time-period.

`#include <stdair/bom/TimePeriod.hpp>`

Inheritance diagram for stdair::TimePeriod::



Public Types

- typedef TimePeriodKey Key_T

Public Member Functions

- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const HolderMap_T & getHolderMap () const
- const Time_T & getTimeRangeStart () const
- const Time_T & getTimeRangeEnd () const
- bool isDepartureTimeValid (const Time_T &) const

Protected Member Functions

- TimePeriod (const Key_T &)
- virtual ~TimePeriod ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent
- HolderMap_T _holderMap

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager

8.157.1 Detailed Description

Class representing the actual attributes for a fare time-period.

Definition at line 18 of file TimePeriod.hpp.

8.157.2 Member Typedef Documentation

8.157.2.1 typedef TimePeriodKey stdair::TimePeriod::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 28 of file TimePeriod.hpp.

8.157.3 Constructor & Destructor Documentation

8.157.3.1 stdair::TimePeriod::TimePeriod (const Key_T &) [protected]

Main constructor.

Definition at line 27 of file TimePeriod.cpp.

8.157.3.2 stdair::TimePeriod::~TimePeriod () [protected, virtual]

Destructor.

Definition at line 32 of file TimePeriod.cpp.

8.157.4 Member Function Documentation

8.157.4.1 void stdair::TimePeriod::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 38 of file TimePeriod.hpp.

References toString().

8.157.4.2 void stdair::TimePeriod::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 47 of file TimePeriod.hpp.

8.157.4.3 std::string stdair::TimePeriod::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 36 of file TimePeriod.cpp.

References describeKey().

Referenced by toStream().

8.157.4.4 const std::string stdair::TimePeriod::describeKey () const [inline]

Get a string describing the key.

Definition at line 58 of file TimePeriod.hpp.

References _key, and stdair::TimePeriodKey::toString().

Referenced by toString().

8.157.4.5 const Key_T& stdair::TimePeriod::getKey () const [inline]

Get the primary key (time range start, time range end).

Definition at line 67 of file TimePeriod.hpp.

References _key.

8.157.4.6 BomAbstract* const stdair::TimePeriod::getParent () const [inline]

Get a reference on the parent object instance.

Definition at line 74 of file TimePeriod.hpp.

References _parent.

8.157.4.7 const HolderMap_T& stdair::TimePeriod::getHolderMap () const [inline]

Get a reference on the children holder.

Definition at line 81 of file TimePeriod.hpp.

References _holderMap.

8.157.4.8 const Time_T& stdair::TimePeriod::getTimeRangeStart () const [inline]

Get the time range start.

Definition at line 88 of file TimePeriod.hpp.

References _key, and stdair::TimePeriodKey::getTimeRangeStart().

Referenced by isDepartureTimeValid().

8.157.4.9 const Time_T& stdair::TimePeriod::getTimeRangeEnd () const [inline]

Get the time range end

Definition at line 95 of file TimePeriod.hpp.

References _key, and stdair::TimePeriodKey::getTimeRangeEnd().

Referenced by isDepartureTimeValid().

8.157.4.10 bool stdair::TimePeriod::isDepartureTimeValid (const Time_T &) const

Check if the given departure time is included in the departure period of the segment path.

Definition at line 44 of file TimePeriod.cpp.

References getTimeRangeEnd(), getTimeRangeStart(), and STDAIR_LOG_DEBUG.

8.157.5 Friends And Related Function Documentation

8.157.5.1 friend class FacBom [friend]

Definition at line 19 of file TimePeriod.hpp.

8.157.5.2 friend class FacCloneBom [friend]

Definition at line 20 of file TimePeriod.hpp.

8.157.5.3 friend class FacBomManager [friend]

Definition at line 21 of file TimePeriod.hpp.

8.157.6 Member Data Documentation

8.157.6.1 Key_T stdair::TimePeriod::_key [protected]

Primary key (flight number and departure date).

Definition at line 133 of file TimePeriod.hpp.

Referenced by describeKey(), getKey(), getTimeRangeEnd(), and getTimeRangeStart().

8.157.6.2 BomAbstract* stdair::TimePeriod::_parent [protected]

Pointer on the parent class (Inventory).

Definition at line 138 of file TimePeriod.hpp.

Referenced by getParent().

8.157.6.3 HolderMap_T stdair::TimePeriod::_holderMap [protected]

Map holding the children.

Definition at line 143 of file TimePeriod.hpp.

Referenced by getHolderMap().

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

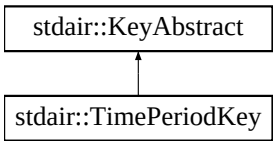
- stdair/bom/TimePeriod.hpp
- stdair/bom/TimePeriod.cpp

8.158 stdair::TimePeriodKey Struct Reference

Key of time-period.

`#include <stdair/bom/TimePeriodKey.hpp>`

Inheritance diagram for stdair::TimePeriodKey::



Public Member Functions

- TimePeriodKey (const Time_T &, const Time_T &)
- TimePeriodKey (const TimePeriodKey &)
- ~TimePeriodKey ()
- const Time_T & getTimeRangeStart () const
- const Time_T & getTimeRangeEnd () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const

8.158.1 Detailed Description

Key of time-period.

Definition at line 15 of file TimePeriodKey.hpp.

8.158.2 Constructor & Destructor Documentation

8.158.2.1 stdair::TimePeriodKey::TimePeriodKey (const Time_T &, const Time_T &)

Main constructor.

Definition at line 21 of file TimePeriodKey.cpp.

8.158.2.2 stdair::TimePeriodKey::TimePeriodKey (const TimePeriodKey &)

Copy constructor.

Definition at line 28 of file TimePeriodKey.cpp.

8.158.2.3 stdair::TimePeriodKey::~TimePeriodKey ()

Destructor.

Definition at line 34 of file TimePeriodKey.cpp.

8.158.3 Member Function Documentation

8.158.3.1 const Time_T& stdair::TimePeriodKey::getTimeRangeStart () const [inline]

Get the time period start.

Definition at line 35 of file TimePeriodKey.hpp.

Referenced by stdair::TimePeriod::getTimeRangeStart().

8.158.3.2 const Time_T& stdair::TimePeriodKey::getTimeRangeEnd () const [inline]

Get the time period end.

Definition at line 42 of file TimePeriodKey.hpp.

Referenced by stdair::TimePeriod::getTimeRangeEnd().

8.158.3.3 void stdair::TimePeriodKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 38 of file TimePeriodKey.cpp.

References toString().

8.158.3.4 void stdair::TimePeriodKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 43 of file TimePeriodKey.cpp.

8.158.3.5 const std::string stdair::TimePeriodKey::toString () const [virtual]

Get the serialised version of the Business Object Key. That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

Reimplemented from stdair::KeyAbstract.

Definition at line 47 of file TimePeriodKey.cpp.

Referenced by stdair::TimePeriod::describeKey(), and toStream().

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

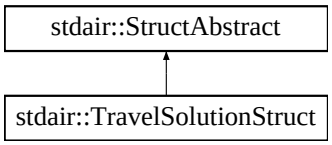
- stdair/bom/TimePeriodKey.hpp
- stdair/bom/TimePeriodKey.cpp

8.159 stdair::TravelSolutionStruct Struct Reference

Structure holding the elements of a travel solution.

`#include <stdair/bom/TravelSolutionStruct.hpp>`

Inheritance diagram for stdair::TravelSolutionStruct::



Public Member Functions

- const SegmentPath_T & getSegmentPath () const
- const ClassAvailabilityMapHolder_T & getClassAvailabilityMapHolder () const
- const ClassObjectIDMapHolder_T & getClassObjectIDMapHolder () const
- const ClassYieldMapHolder_T & getClassYieldMapHolder () const
- const BidPriceVectorHolder_T & getBidPriceVectorHolder () const
- const ClassBpvMapHolder_T & getClassBpvMapHolder () const
- const FareOptionList_T & getFareOptionList () const
- FareOptionList_T & getFareOptionListRef ()
- const FareOptionStruct & getChosenFareOption () const
- void addSegment (const std::string &)
- void addClassAvailabilityMap (const ClassAvailabilityMap_T &)
- void addClassObjectIDMap (const ClassObjectIDMap_T &)
- void addClassYieldMap (const ClassYieldMap_T &)
- void addBidPriceVector (const BidPriceVector_T &)
- void addClassBpvMap (const ClassBpvMap_T &)
- void addFareOption (const FareOptionStruct &)
- void setChosenFareOption (const FareOptionStruct &iChosenFO)
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string describe () const
- const std::string display () const

- const std::string describeSegmentPath () const
- TravelSolutionStruct ()
- ~TravelSolutionStruct ()

8.159.1 Detailed Description

Structure holding the elements of a travel solution.

Definition at line 24 of file TravelSolutionStruct.hpp.

8.159.2 Constructor & Destructor Documentation

8.159.2.1 stdair::TravelSolutionStruct::TravelSolutionStruct ()

Default constructor.

Definition at line 15 of file TravelSolutionStruct.cpp.

8.159.2.2 stdair::TravelSolutionStruct::~TravelSolutionStruct ()

Destructor.

Definition at line 19 of file TravelSolutionStruct.cpp.

8.159.3 Member Function Documentation

8.159.3.1 const SegmentPath_T& stdair::TravelSolutionStruct::getSegmentPath () const [inline]

Get the segment path.

Definition at line 28 of file TravelSolutionStruct.hpp.

8.159.3.2 const ClassAvailabilityMapHolder_T& stdair::TravelSolutionStruct::getClassAvailabilityMapHolder () const [inline]

Get the holder of avalabilities.

Definition at line 33 of file TravelSolutionStruct.hpp.

8.159.3.3 const ClassObjectIDMapHolder_T& stdair::TravelSolutionStruct::getClassObjectIDMapHolder () const [inline]

Get the holder of object ID's.

Definition at line 38 of file TravelSolutionStruct.hpp.

8.159.3.4 const ClassYieldMapHolder_T& stdair::TravelSolutionStruct::getClassYieldMapHolder () const [inline]

Get the holder of yields.

Definition at line 43 of file TravelSolutionStruct.hpp.

8.159.3.5 const BidPriceVectorHolder_T& stdair::TravelSolutionStruct::getBidPriceVectorHolder () const [inline]

Get the holder of bid price vectors.

Definition at line 48 of file TravelSolutionStruct.hpp.

8.159.3.6 const ClassBpvMapHolder_T& stdair::TravelSolutionStruct::getClassBpvMapHolder () const [inline]

Get the holder of class - bid price reference.

Definition at line 53 of file TravelSolutionStruct.hpp.

8.159.3.7 const FareOptionList_T& stdair::TravelSolutionStruct::getFareOptionList () const [inline]

Get the list of fare options.

Definition at line 58 of file TravelSolutionStruct.hpp.

8.159.3.8 FareOptionList_T& stdair::TravelSolutionStruct::getFareOptionListRef () [inline]

Get the non-const list of fare options.

Definition at line 63 of file TravelSolutionStruct.hpp.

8.159.3.9 const FareOptionStruct& stdair::TravelSolutionStruct::getChosenFareOption () const [inline]

Get the chosen fare option.

Definition at line 68 of file TravelSolutionStruct.hpp.

8.159.3.10 void stdair::TravelSolutionStruct::addSegment (const std::string &)

Add a segment key to the segment path.

Definition at line 154 of file TravelSolutionStruct.cpp.

8.159.3.11 void stdair::TravelSolutionStruct::addClassAvailabilityMap (const ClassAvailabilityMap_T &)

Add a class availability map.

Definition at line 160 of file TravelSolutionStruct.cpp.

8.159.3.12 void stdair::TravelSolutionStruct::addClassObjectIDMap (const ClassObjectIDMap_T &)

Add a class object ID map.

Definition at line 166 of file TravelSolutionStruct.cpp.

8.159.3.13 void stdair::TravelSolutionStruct::addClassYieldMap (const ClassYieldMap_T &)

Add a class yield map.

Definition at line 172 of file TravelSolutionStruct.cpp.

8.159.3.14 void stdair::TravelSolutionStruct::addBidPriceVector (const BidPriceVector_T &)

Add a bid price vector.

Definition at line 178 of file TravelSolutionStruct.cpp.

8.159.3.15 void stdair::TravelSolutionStruct::addClassBpvMap (const ClassBpvMap_T &)

Add a class bpv reference map.

Definition at line 184 of file TravelSolutionStruct.cpp.

8.159.3.16 void stdair::TravelSolutionStruct::addFareOption (const FareOptionStruct &)

Add a fare option.

Definition at line 190 of file TravelSolutionStruct.cpp.

8.159.3.17 void stdair::TravelSolutionStruct::setChosenFareOption (const FareOptionStruct & *iChosenFO*) [inline]

Set the chosen fare option.

Definition at line 97 of file TravelSolutionStruct.hpp.

8.159.3.18 void stdair::TravelSolutionStruct::toStream (std::ostream & *ioOut*) const

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::StructAbstract.

Definition at line 23 of file TravelSolutionStruct.cpp.

References describe().

8.159.3.19 void stdair::TravelSolutionStruct::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::StructAbstract.

Definition at line 28 of file TravelSolutionStruct.cpp.

8.159.3.20 const std::string stdair::TravelSolutionStruct::describe () const [virtual]

Display of the structure.

Implements stdair::StructAbstract.

Definition at line 53 of file TravelSolutionStruct.cpp.

References stdair::FareOptionStruct::describe(), stdair::BomKeyManager::extractKeys(), and stdair::ParsedKey::toString().

Referenced by toStream().

8.159.3.21 const std::string stdair::TravelSolutionStruct::display () const

Display of the structure.

Definition at line 95 of file TravelSolutionStruct.cpp.

References stdair::FareOptionStruct::display(), stdair::BomKeyManager::extractKeys(), and stdair::ParsedKey::toString().

8.159.3.22 const std::string stdair::TravelSolutionStruct::describeSegmentPath () const

Display only the segment path.

Definition at line 32 of file TravelSolutionStruct.cpp.

References stdair::BomKeyManager::extractKeys(), and stdair::ParsedKey::toString().

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

- stdair/bom/TravelSolutionStruct.hpp
- stdair/bom/TravelSolutionStruct.cpp

8.160 soci::type_conversion< stdair::AirlineStruct > Struct Template Reference

`#include <stdair/dbadaptor/DbaAirline.hpp>`

Public Types

- typedef values base_type

Static Public Member Functions

- static void from_base (values const &iAirlineValues, indicator, stdair::AirlineStruct &ioAirline)
- static void to_base (const stdair::AirlineStruct &iAirline, values &ioAirlineValues, indicator &ioIndicator)

8.160.1 Detailed Description

template<> struct soci::type_conversion< stdair::AirlineStruct >

Specify how the AirlineStruct struct can be converted to (resp. from) values stored into (resp. retrieved from) database, using the SOCI framework.

Definition at line 25 of file DbaAirline.hpp.

8.160.2 Member Typedef Documentation

8.160.2.1 typedef values soci::type_conversion< stdair::AirlineStruct >::base_type

Definition at line 27 of file DbaAirline.hpp.

8.160.3 Member Function Documentation

8.160.3.1 void soci::type_conversion< stdair::AirlineStruct >::from_base (values const & *iAirlineValues*, indicator, stdair::AirlineStruct & *ioAirline*) [static]

Fill an Airline object from the database values.

Definition at line 17 of file DbaAirline.cpp.

References stdair::AirlineStruct::setAirlineCode(), and stdair::AirlineStruct::setAirlineName().

8.160.3.2 void soci::type_conversion< stdair::AirlineStruct >::to_base (const stdair::AirlineStruct & *iAirline*, values & *ioAirlineValues*, indicator & *ioIndicator*) [static]

Fill the database values from an Airline object.

Definition at line 30 of file DbaAirline.cpp.

References stdair::AirlineStruct::getAirlineCode(), and stdair::AirlineStruct::getAirlineName().

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

- stdair/dbadaptor/DbaAirline.hpp
- stdair/dbadaptor/DbaAirline.cpp

8.161 TypeWithSize< size > Class Template Reference

`#include <stdair/basic/float_utils_google.hpp>`

Public Types

- typedef void UInt

8.161.1 Detailed Description

template<size_t size> class TypeWithSize< size >

Definition at line 54 of file float_utils_google.hpp.

8.161.2 Member Typedef Documentation

8.161.2.1 template<size_t size> typedef void TypeWithSize< size >::UInt

Definition at line 58 of file float_utils_google.hpp.

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

- stdair/basic/float_utils_google.hpp

8.162 TypeWithSize< 4 > Class Template Reference

`#include <stdair/basic/float_utils_google.hpp>`

Public Types

- typedef int Int
- typedef unsigned int UInt

8.162.1 Detailed Description

template<> class TypeWithSize< 4 >

Definition at line 63 of file float_utils_google.hpp.

8.162.2 Member Typedef Documentation

8.162.2.1 typedef int TypeWithSize< 4 >::Int

Definition at line 69 of file float_utils_google.hpp.

8.162.2.2 typedef unsigned int TypeWithSize< 4 >::UInt

Definition at line 70 of file float_utils_google.hpp.

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

- stdair/basic/float_utils_google.hpp

8.163 TypeWithSize< 8 > Class Template Reference

`#include <stdair/basic/float_utils_google.hpp>`

Public Types

- typedef long long Int
- typedef unsigned long long UInt

8.163.1 Detailed Description

template<> class TypeWithSize< 8 >

Definition at line 75 of file float_utils_google.hpp.

8.163.2 Member Typedef Documentation

8.163.2.1 typedef long long TypeWithSize< 8 >::Int

Definition at line 81 of file float_utils_google.hpp.

8.163.2.2 typedef unsigned long long TypeWithSize< 8 >::UInt

Definition at line 82 of file float_utils_google.hpp.

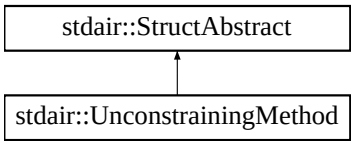
The documentation for this class was generated from the following file:

- stdair/basic/float_utils_google.hpp

8.164 stdair::UnconstrainingMethod Struct Reference

`#include <stdair/basic/UnconstrainingMethod.hpp>`

Inheritance diagram for stdair::UnconstrainingMethod::



Public Types

- EM = 0
- LAST_VALUE
- enum EN_UnconstrainingMethod { EM = 0, LAST_VALUE }

Public Member Functions

- EN_UnconstrainingMethod getMethod () const
- std::string getMethodAsString () const
- const std::string describe () const
- bool operator== (const EN_UnconstrainingMethod &) const
- UnconstrainingMethod (const EN_UnconstrainingMethod &)
- UnconstrainingMethod (const char iMethod)
- UnconstrainingMethod (const UnconstrainingMethod &)
- void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

Static Public Member Functions

- static const std::string & getLabel (const EN_UnconstrainingMethod &)
- static char getMethodLabel (const EN_UnconstrainingMethod &)
- static std::string getMethodLabelAsString (const EN_UnconstrainingMethod &)
- static std::string describeLabels ()

8.164.1 Detailed Description

Enumeration of unconstraining methods.

Definition at line 15 of file UnconstrainingMethod.hpp.

8.164.2 Member Enumeration Documentation

8.164.2.1 enum stdair::UnconstrainingMethod::EN_UnconstrainingMethod

Enumerator:

EM

LAST_VALUE

Definition at line 17 of file UnconstrainingMethod.hpp.

8.164.3 Constructor & Destructor Documentation

8.164.3.1 stdair::UnconstrainingMethod::UnconstrainingMethod (const EN_UnconstrainingMethod &)

Constructor.

Definition at line 36 of file UnconstrainingMethod.cpp.

8.164.3.2 stdair::UnconstrainingMethod::UnconstrainingMethod (const char *iMethod*)

Constructor.

Definition at line 41 of file UnconstrainingMethod.cpp.

References describeLabels(), EM, and LAST_VALUE.

8.164.3.3 stdair::UnconstrainingMethod::UnconstrainingMethod (const UnconstrainingMethod &)

Default copy constructor.

Definition at line 30 of file UnconstrainingMethod.cpp.

8.164.4 Member Function Documentation

8.164.4.1 const std::string & stdair::UnconstrainingMethod::getLabel (const EN_UnconstrainingMethod &) [static]

Get the label as a string (e.g., "Expectation-Maximisation")

Definition at line 58 of file UnconstrainingMethod.cpp.

8.164.4.2 char stdair::UnconstrainingMethod::getMethodLabel (const EN_UnconstrainingMethod &) [static]

Get the label as a single char (e.g., 'T' or 'B').

Definition at line 63 of file UnconstrainingMethod.cpp.

8.164.4.3 std::string stdair::UnconstrainingMethod::getMethodLabelAsString (const EN_UnconstrainingMethod &) [static]

Get the label as a string of a single char (e.g., "T" or "B").

Definition at line 69 of file UnconstrainingMethod.cpp.

8.164.4.4 std::string stdair::UnconstrainingMethod::describeLabels () [static]

List the labels.

Definition at line 76 of file UnconstrainingMethod.cpp.

References LAST_VALUE.

Referenced by UnconstrainingMethod().

8.164.4.5 UnconstrainingMethod::EN_UnconstrainingMethod stdair::UnconstrainingMethod::getMethod () const

Get the enumerated value.

Definition at line 88 of file UnconstrainingMethod.cpp.

Referenced by stdair::AirlineFeature::getUnconstrainingMethod().

8.164.4.6 std::string stdair::UnconstrainingMethod::getMethodAsString () const

Get the enumerated value as a short string (e.g., "T" or "B").

Definition at line 93 of file UnconstrainingMethod.cpp.

8.164.4.7 const std::string stdair::UnconstrainingMethod::describe () const [virtual]

Give a description of the structure (e.g., "Expectation-Maximisation").

Implements stdair::StructAbstract.

Definition at line 100 of file UnconstrainingMethod.cpp.

8.164.4.8 bool stdair::UnconstrainingMethod::operator== (const EN_UnconstrainingMethod &) const

Comparison operator.

Definition at line 108 of file UnconstrainingMethod.cpp.

8.164.4.9 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const [inline, inherited]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References stdair::StructAbstract::describe().

8.164.4.10 virtual void stdair::StructAbstract::fromStream (std::istream & *ioIn*) [inline, virtual, inherited]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented in stdair::ProgressStatusSet, stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::EventStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 38 of file StructAbstract.hpp.

Referenced by operator>>().

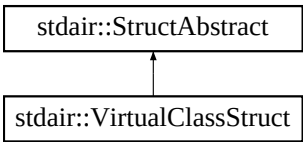
The documentation for this struct was generated from the following files:

- stdair/basic/UnconstrainingMethod.hpp
- stdair/basic/UnconstrainingMethod.cpp

8.165 stdair::VirtualClassStruct Struct Reference

`#include <stdair/bom/VirtualClassStruct.hpp>`

Inheritance diagram for stdair::VirtualClassStruct::



Public Member Functions

- const BookingClassList_T & getBookingClassList () const
- const Yield_T & getYield () const
- const MeanValue_T & getMean () const
- const StdDevValue_T & getStdDev () const
- const BookingLimit_T & getCumulatedBookingLimit () const
- const ProtectionLevel_T & getCumulatedProtection () const
- const GeneratedDemandVector_T getGeneratedDemandVector () const
- void setYield (const Yield_T &iYield)
- void setMean (const MeanValue_T &iMean)
- void setStdDev (const StdDevValue_T &iStdDev)
- void setCumulatedBookingLimit (const BookingLimit_T &iBL)
- void setCumulatedProtection (const ProtectionLevel_T &iP)
- void addBookingClass (BookingClass &iBookingClass)
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)

- const std::string describe () const
- VirtualClassStruct (const VirtualClassStruct &)
- VirtualClassStruct (const BookingClassList_T &)
- ~VirtualClassStruct ()

8.165.1 Detailed Description

Structure holding the elements of a virtual class.

Definition at line 24 of file VirtualClassStruct.hpp.

8.165.2 Constructor & Destructor Documentation

8.165.2.1 stdair::VirtualClassStruct::VirtualClassStruct (const VirtualClassStruct &)

Default copy constructor.

Definition at line 19 of file VirtualClassStruct.cpp.

8.165.2.2 stdair::VirtualClassStruct::VirtualClassStruct (const BookingClassList_T &)

Constructor.

Definition at line 26 of file VirtualClassStruct.cpp.

8.165.2.3 stdair::VirtualClassStruct::~VirtualClassStruct ()

Destructor.

Definition at line 31 of file VirtualClassStruct.cpp.

8.165.3 Member Function Documentation

8.165.3.1 const BookingClassList_T& stdair::VirtualClassStruct::getBookingClassList () const [inline]

Get the list of booking class.

Definition at line 28 of file VirtualClassStruct.hpp.

8.165.3.2 const Yield_T& stdair::VirtualClassStruct::getYield () const [inline]

Get the yield (average price paid for that virtual class).

Definition at line 33 of file VirtualClassStruct.hpp.

8.165.3.3 const MeanValue_T& stdair::VirtualClassStruct::getMean () const [inline]

Get the mean value of the demand distribution.

Definition at line 38 of file VirtualClassStruct.hpp.

8.165.3.4 const StdDevValue_T& stdair::VirtualClassStruct::getStdDev () const [inline]

Get the standard deviation of the demand distribution.

Definition at line 43 of file VirtualClassStruct.hpp.

8.165.3.5 const BookingLimit_T& stdair::VirtualClassStruct::getCumulatedBookingLimit () const [inline]

Get the booking limit of the class.

Definition at line 48 of file VirtualClassStruct.hpp.

8.165.3.6 const ProtectionLevel_T& stdair::VirtualClassStruct::getCumulatedProtection () const [inline]

Get the protection level of the class.

Definition at line 53 of file VirtualClassStruct.hpp.

8.165.3.7 const GeneratedDemandVector_T stdair::VirtualClassStruct::getGeneratedDemandVector () const

Get the generated demand sample vector for Monte-Carlo method.

Definition at line 54 of file VirtualClassStruct.cpp.

References stdair::BookingClass::getGeneratedDemandVector().

8.165.3.8 void stdair::VirtualClassStruct::setYield (const Yield_T & *iYield*) [inline]

Set the yield (average price paid for that virtual class).

Definition at line 63 of file VirtualClassStruct.hpp.

8.165.3.9 void stdair::VirtualClassStruct::setMean (const MeanValue_T & *iMean*) [inline]

Set the mean value of the demand distribution.

Definition at line 68 of file VirtualClassStruct.hpp.

8.165.3.10 void stdair::VirtualClassStruct::setStdDev (const StdDevValue_T & *iStdDev*) [inline]

Set the standard deviation of the demand distribution.

Definition at line 73 of file VirtualClassStruct.hpp.

8.165.3.11 void stdair::VirtualClassStruct::setCumulatedBookingLimit (const BookingLimit_T & *iBL*) [inline]

Set the booking limit of the class.

Definition at line 78 of file VirtualClassStruct.hpp.

8.165.3.12 void stdair::VirtualClassStruct::setCumulatedProtection (const ProtectionLevel_T & *iP*) [inline]

Set the protection level of the class.

Definition at line 83 of file VirtualClassStruct.hpp.

8.165.3.13 void stdair::VirtualClassStruct::addBookingClass (BookingClass & *iBookingClass*) [inline]

Add a booking class to the list of booking classes. Note: it is not a link Parent/Child so we don't use the FacBom. The Virtual Classes are not bom objects because the optimiser needs to build them before each optimisation.

Definition at line 92 of file VirtualClassStruct.hpp.

8.165.3.14 void stdair::VirtualClassStruct::toStream (std::ostream & *ioOut*) const

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::StructAbstract.

Definition at line 36 of file VirtualClassStruct.cpp.

References describe().

8.165.3.15 void stdair::VirtualClassStruct::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::StructAbstract.

Definition at line 41 of file VirtualClassStruct.cpp.

8.165.3.16 const std::string stdair::VirtualClassStruct::describe () const [virtual]

Display of the structure.

Implements stdair::StructAbstract.

Definition at line 45 of file VirtualClassStruct.cpp.

Referenced by toStream().

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

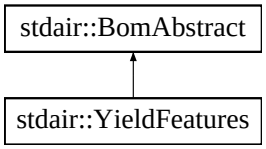
- stdair/bom/VirtualClassStruct.hpp
- stdair/bom/VirtualClassStruct.cpp

8.166 stdair::YieldFeatures Class Reference

Class representing the actual attributes for a yield date-period.

`#include <stdair/bom/YieldFeatures.hpp>`

Inheritance diagram for stdair::YieldFeatures::



Public Types

- typedef YieldFeaturesKey Key_T

Public Member Functions

- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const HolderMap_T & getHolderMap () const
- const CabinCode_T & getCabinCode () const
- const TripType_T & getTripType () const
- bool isTripTypeValid (const TripType_T &) const

Protected Member Functions

- YieldFeatures (const Key_T &)
- virtual ~YieldFeatures ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent
- HolderMap_T _holderMap

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager

8.166.1 Detailed Description

Class representing the actual attributes for a yield date-period.

Definition at line 19 of file YieldFeatures.hpp.

8.166.2 Member Typedef Documentation

8.166.2.1 typedef YieldFeaturesKey stdair::YieldFeatures::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 29 of file YieldFeatures.hpp.

8.166.3 Constructor & Destructor Documentation

8.166.3.1 stdair::YieldFeatures::YieldFeatures (const Key_T &) [protected]

Main constructor.

Definition at line 28 of file YieldFeatures.cpp.

8.166.3.2 stdair::YieldFeatures::~YieldFeatures () [protected, virtual]

Destructor.

Definition at line 33 of file YieldFeatures.cpp.

8.166.4 Member Function Documentation

8.166.4.1 void stdair::YieldFeatures::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 38 of file YieldFeatures.hpp.

References toString().

8.166.4.2 void stdair::YieldFeatures::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 47 of file YieldFeatures.hpp.

8.166.4.3 std::string stdair::YieldFeatures::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 37 of file YieldFeatures.cpp.

References describeKey().

Referenced by toStream().

8.166.4.4 const std::string stdair::YieldFeatures::describeKey () const [inline]

Get a string describing the key.

Definition at line 58 of file YieldFeatures.hpp.

References _key, and stdair::YieldFeaturesKey::toString().

Referenced by toString().

8.166.4.5 const Key_T& stdair::YieldFeatures::getKey () const [inline]

Get the primary key (trip type, cabin code).

Definition at line 67 of file YieldFeatures.hpp.

References _key.

8.166.4.6 BomAbstract* const stdair::YieldFeatures::getParent () const [inline]

Get a reference on the parent object instance.

Definition at line 74 of file YieldFeatures.hpp.

References _parent.

8.166.4.7 const HolderMap_T& stdair::YieldFeatures::getHolderMap () const [inline]

Get a reference on the children holder.

Definition at line 81 of file YieldFeatures.hpp.

References _holderMap.

8.166.4.8 const CabinCode_T& stdair::YieldFeatures::getCabinCode () const [inline]

Get the cabin code.

Definition at line 88 of file YieldFeatures.hpp.

References _key, and stdair::YieldFeaturesKey::getCabinCode().

8.166.4.9 const TripType_T& stdair::YieldFeatures::getTripType () const [inline]

Get the trip type.

Definition at line 95 of file YieldFeatures.hpp.

References _key, and stdair::YieldFeaturesKey::getTripType().

Referenced by isTripTypeValid().

8.166.4.10 bool stdair::YieldFeatures::isTripTypeValid (const TripType_T &) const

Check whether the fare rule trip type corresponds to the booking request trip type.

Definition at line 45 of file YieldFeatures.cpp.

References getTripType(), stdair::TRIP_TYPE_INBOUND, stdair::TRIP_TYPE_OUTBOUND, and stdair::TRIP_TYPE_ROUND_TRIP.

8.166.5 Friends And Related Function Documentation

8.166.5.1 friend class FacBom [friend]

Definition at line 20 of file YieldFeatures.hpp.

8.166.5.2 friend class FacCloneBom [friend]

Definition at line 21 of file YieldFeatures.hpp.

8.166.5.3 friend class FacBomManager [friend]

Definition at line 22 of file YieldFeatures.hpp.

8.166.6 Member Data Documentation

8.166.6.1 Key_T stdair::YieldFeatures::_key [protected]

Primary key (flight number and departure date).

Definition at line 138 of file YieldFeatures.hpp.

Referenced by describeKey(), getCabinCode(), getKey(), and getTripType().

8.166.6.2 BomAbstract* stdair::YieldFeatures::_parent [protected]

Pointer on the parent class.

Definition at line 143 of file YieldFeatures.hpp.

Referenced by getParent().

8.166.6.3 HolderMap_T stdair::YieldFeatures::_holderMap [protected]

Map holding the children.

Definition at line 148 of file YieldFeatures.hpp.

Referenced by getHolderMap().

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

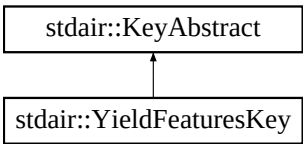
- stdair/bom/YieldFeatures.hpp
- stdair/bom/YieldFeatures.cpp

8.167 stdair::YieldFeaturesKey Struct Reference

Key of date-period.

`#include <stdair/bom/YieldFeaturesKey.hpp>`

Inheritance diagram for stdair::YieldFeaturesKey::



Public Member Functions

- YieldFeaturesKey (const TripType_T &, const CabinCode_T &)
- YieldFeaturesKey (const YieldFeaturesKey &)
- ~YieldFeaturesKey ()
- const TripType_T & getTripType () const
- const CabinCode_T & getCabinCode () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const

8.167.1 Detailed Description

Key of date-period.

Definition at line 18 of file YieldFeaturesKey.hpp.

8.167.2 Constructor & Destructor Documentation

8.167.2.1 stdair::YieldFeaturesKey::YieldFeaturesKey (const TripType_T &, const CabinCode_T &)

Main constructor.

Definition at line 21 of file YieldFeaturesKey.cpp.

8.167.2.2 stdair::YieldFeaturesKey::YieldFeaturesKey (const YieldFeaturesKey &)

Copy constructor.

Definition at line 27 of file YieldFeaturesKey.cpp.

8.167.2.3 stdair::YieldFeaturesKey::~YieldFeaturesKey ()

Destructor.

Definition at line 32 of file YieldFeaturesKey.cpp.

8.167.3 Member Function Documentation

8.167.3.1 const TripType_T& stdair::YieldFeaturesKey::getTripType () const [inline]

Get the fare trip type.

Definition at line 44 of file YieldFeaturesKey.hpp.

Referenced by stdair::YieldFeatures::getTripType().

8.167.3.2 const CabinCode_T& stdair::YieldFeaturesKey::getCabinCode () const [inline]

Get the cabin.

Definition at line 51 of file YieldFeaturesKey.hpp.

Referenced by stdair::YieldFeatures::getCabinCode().

8.167.3.3 void stdair::YieldFeaturesKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 36 of file YieldFeaturesKey.cpp.

References toString().

8.167.3.4 void stdair::YieldFeaturesKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 41 of file YieldFeaturesKey.cpp.

8.167.3.5 const std::string stdair::YieldFeaturesKey::toString () const [virtual]

Get the serialised version of the Business Object Key. That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

Reimplemented from stdair::KeyAbstract.

Definition at line 45 of file YieldFeaturesKey.cpp.

Referenced by stdair::YieldFeatures::describeKey(), and toStream().

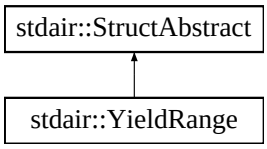
The documentation for this struct was generated from the following files:

- stdair/bom/YieldFeaturesKey.hpp
- stdair/bom/YieldFeaturesKey.cpp

8.168 stdair::YieldRange Class Reference

`#include <stdair/basic/YieldRange.hpp>`

Inheritance diagram for stdair::YieldRange::



Public Member Functions

- YieldRange ()
- YieldRange (const YieldRange &)
- YieldRange (const Yield_T iUpperYield)
- YieldRange (const Yield_T iUpperYield, const Yield_T iAverageYield)
- YieldRange (const Yield_T iUpperYield, const Yield_T iAverageYield, const Yield_T iLowerYield)
- virtual ~YieldRange ()
- Yield_T getUpperYield () const
- Yield_T getAverageYield () const
- Yield_T getLowerYield () const
- void setUpperYield (const Yield_T iUpperYield)
- void setAverageYield (const Yield_T iAverageYield)
- void setLowerYield (const Yield_T iLowerYield)
- void toStream (std::ostream &) const
- void fromStream (std::istream &)
- const std::string describe () const

8.168.1 Detailed Description

Class representing a range of yields.

Typically, bookings are priced according to rules (e.g., fare rules), leading to slight variations of revenues for a given product. The "yield range" captures the extent of revenues earned for a given product.

When no average and lower yields are defined, they are assumed to be equal to the upper yield.

Note that the lower yield is generally not defined, as it corresponds to the upper yield of the lower yield range.

Definition at line 23 of file YieldRange.hpp.

8.168.2 Constructor & Destructor Documentation

8.168.2.1 stdair::YieldRange::YieldRange ()

Constructors.

Definition at line 13 of file YieldRange.cpp.

8.168.2.2 stdair::YieldRange::YieldRange (const YieldRange &)

Definition at line 20 of file YieldRange.cpp.

8.168.2.3 stdair::YieldRange::YieldRange (const Yield_T *iUpperYield*)

Definition at line 27 of file YieldRange.cpp.

8.168.2.4 stdair::YieldRange::YieldRange (const Yield_T *iUpperYield*, const Yield_T *iAverageYield*)

Definition at line 33 of file YieldRange.cpp.

8.168.2.5 stdair::YieldRange::YieldRange (const Yield_T *iUpperYield*, const Yield_T *iAverageYield*, const Yield_T *iLowerYield*)

Definition at line 40 of file YieldRange.cpp.

8.168.2.6 stdair::YieldRange::~YieldRange () [virtual]

Constructors.

Definition at line 48 of file YieldRange.cpp.

8.168.3 Member Function Documentation

8.168.3.1 Yield_T stdair::YieldRange::getUpperYield () const [inline]

Getter for the upper yield of the range.

Definition at line 39 of file YieldRange.hpp.

8.168.3.2 Yield_T stdair::YieldRange::getAverageYield () const [inline]

Getter for the average yield of the range.

Definition at line 43 of file YieldRange.hpp.

8.168.3.3 Yield_T stdair::YieldRange::getLowerYield () const [inline]

Getter for the lower yield of the range.

Definition at line 47 of file YieldRange.hpp.

8.168.3.4 void stdair::YieldRange::setUpperYield (const Yield_T *iUpperYield*) [inline]

Setter for the upper yield of the range.

Definition at line 53 of file YieldRange.hpp.

8.168.3.5 void stdair::YieldRange::setAverageYield (const Yield_T *iAverageYield*) [inline]

Setter for the average yield of the range.

Definition at line 57 of file YieldRange.hpp.

8.168.3.6 void stdair::YieldRange::setLowerYield (const Yield_T *iLowerYield*) [inline]

Setter for the lower yield of the range.

Definition at line 61 of file YieldRange.hpp.

8.168.3.7 void stdair::YieldRange::toStream (std::ostream &) const

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::StructAbstract.

Definition at line 52 of file YieldRange.cpp.

8.168.3.8 void stdair::YieldRange::fromStream (std::istream &) [virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::StructAbstract.

Definition at line 58 of file YieldRange.cpp.

8.168.3.9 const std::string stdair::YieldRange::describe () const [virtual]

Display of the structure.

Implements stdair::StructAbstract.

Definition at line 62 of file YieldRange.cpp.

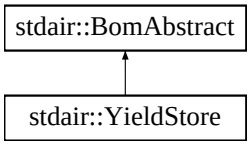
The documentation for this class was generated from the following files:

- stdair/basic/YieldRange.hpp
- stdair/basic/YieldRange.cpp

8.169 stdair::YieldStore Class Reference

`#include <stdair/bom/YieldStore.hpp>`

Inheritance diagram for stdair::YieldStore::



Public Types

- typedef YieldStoreKey Key_T

Public Member Functions

- void toStream (std::ostream &ioOut) const
- BomAbstract *const getParent () const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- const Key_T & getKey () const
- const AirlineCode_T & getAirlineCode () const

Protected Member Functions

- YieldStore (const Key_T &)
- YieldStore (const YieldStore &)
- ~YieldStore ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent

Friends

- class FacBom
- class FacBomManager

8.169.1 Detailed Description

Class representing the actual attributes for an airline YieldStore.

Definition at line 18 of file YieldStore.hpp.

8.169.2 Member Typedef Documentation

8.169.2.1 typedef YieldStoreKey stdair::YieldStore::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 25 of file YieldStore.hpp.

8.169.3 Constructor & Destructor Documentation

8.169.3.1 stdair::YieldStore::YieldStore (const Key_T &) [protected]

Default constructors.

Definition at line 13 of file YieldStore.cpp.

8.169.3.2 stdair::YieldStore::YieldStore (const YieldStore &) [protected]

8.169.3.3 stdair::YieldStore::~YieldStore () [protected]

Destructor.

Definition at line 17 of file YieldStore.cpp.

8.169.4 Member Function Documentation

8.169.4.1 void stdair::YieldStore::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 31 of file YieldStore.hpp.

References toString().

8.169.4.2 BomAbstract* const stdair::YieldStore::getParent () const [inline]

Get the parent object.

Definition at line 34 of file YieldStore.hpp.

References _parent.

8.169.4.3 void stdair::YieldStore::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 38 of file YieldStore.hpp.

8.169.4.4 std::string stdair::YieldStore::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 21 of file YieldStore.cpp.

References _key, and stdair::YieldStoreKey::toString().

Referenced by toStream().

8.169.4.5 const std::string stdair::YieldStore::describeKey () const [inline]

Get a string describing the key.

Definition at line 44 of file YieldStore.hpp.

References _key, and stdair::YieldStoreKey::toString().

8.169.4.6 const Key_T& stdair::YieldStore::getKey () const [inline]

Get the YieldStore key.

Definition at line 49 of file YieldStore.hpp.

References _key.

8.169.4.7 const AirlineCode_T& stdair::YieldStore::getAirlineCode () const [inline]

Get the airline code.

Definition at line 52 of file YieldStore.hpp.

References _key, and stdair::YieldStoreKey::getAirlineCode().

8.169.5 Friends And Related Function Documentation

8.169.5.1 friend class FacBom [friend]

Definition at line 19 of file YieldStore.hpp.

8.169.5.2 friend class FacBomManager [friend]

Definition at line 20 of file YieldStore.hpp.

8.169.6 Member Data Documentation

8.169.6.1 Key_T stdair::YieldStore::_key [protected]

The key of both structure and objects.

Definition at line 66 of file YieldStore.hpp.

Referenced by describeKey(), getAirlineCode(), getKey(), and toString().

8.169.6.2 BomAbstract* stdair::YieldStore::_parent [protected]

Definition at line 67 of file YieldStore.hpp.

Referenced by getParent().

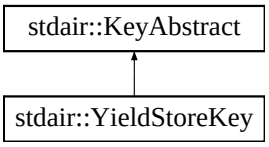
The documentation for this class was generated from the following files:

- stdair/bom/YieldStore.hpp
- stdair/bom/YieldStore.cpp

8.170 stdair::YieldStoreKey Struct Reference

`#include <stdair/bom/YieldStoreKey.hpp>`

Inheritance diagram for stdair::YieldStoreKey::



Public Member Functions

- YieldStoreKey (const AirlineCode_T &iAirlineCode)
- YieldStoreKey (const YieldStoreKey &)
- ~YieldStoreKey ()
- const AirlineCode_T & getAirlineCode () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const

8.170.1 Detailed Description

Key of YieldStore.

Definition at line 14 of file YieldStoreKey.hpp.

8.170.2 Constructor & Destructor Documentation

8.170.2.1 stdair::YieldStoreKey::YieldStoreKey (const AirlineCode_T & *iAirlineCode*)

Constructors.

Definition at line 10 of file YieldStoreKey.cpp.

8.170.2.2 stdair::YieldStoreKey::YieldStoreKey (const YieldStoreKey &)

Definition at line 14 of file YieldStoreKey.cpp.

8.170.2.3 stdair::YieldStoreKey::~YieldStoreKey ()

Destructor.

Definition at line 19 of file YieldStoreKey.cpp.

8.170.3 Member Function Documentation

8.170.3.1 const AirlineCode_T& stdair::YieldStoreKey::getAirlineCode () const [inline]

Get the airline code.

Definition at line 30 of file YieldStoreKey.hpp.

Referenced by stdair::YieldStore::getAirlineCode().

8.170.3.2 void stdair::YieldStoreKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 23 of file YieldStoreKey.cpp.

References toString().

8.170.3.3 void stdair::YieldStoreKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 28 of file YieldStoreKey.cpp.

8.170.3.4 const std::string stdair::YieldStoreKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from stdair::KeyAbstract.

Definition at line 32 of file YieldStoreKey.cpp.

Referenced by stdair::YieldStore::describeKey(), toStream(), and stdair::YieldStore::toString().

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

- stdair/bom/YieldStoreKey.hpp
- stdair/bom/YieldStoreKey.cpp

9 StdAir File Documentation

9.1 batches/stdair.cpp File Reference

9.2 doc/local/authors.doc File Reference

9.3 doc/local/codingrules.doc File Reference

9.4 doc/local/copyright.doc File Reference

9.5 doc/local/documentation.doc File Reference

9.6 doc/local/features.doc File Reference

9.7 doc/local/help_wanted.doc File Reference

9.8 doc/local/howto_release.doc File Reference

9.9 doc/local/index.doc File Reference

9.10 doc/local/installation.doc File Reference

9.11 doc/local/linking.doc File Reference

9.12 doc/local/test.doc File Reference

9.13 doc/local/users_guide.doc File Reference

9.14 doc/local/verification.doc File Reference

9.15 doc/tutorial/tutorial.doc File Reference

9.16 stdair/basic/BasChronometer.cpp File Reference

```
#include <cassert>
#include <stdair/basic/BasChronometer.hpp>
```

Namespaces

- namespace stdair

9.17 stdair/basic/BasChronometer.hpp File Reference

```
#include <boost/date_time/posix_time/posix_time.hpp>
```

Namespaces

- namespace stdair

Classes

- struct stdair::BasChronometer

9.18 stdair/basic/BasConst.cpp File Reference

```
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/basic/BasConst_Event.hpp>
#include <stdair/basic/BasConst_Request.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_BookingClass.hpp>
#include <stdair/basic/BasConst_Yield.hpp>
#include <stdair/basic/BasConst_DefaultObject.hpp>
#include <stdair/basic/BasConst_Period_BOM.hpp>
#include <stdair/basic/BasConst_TravelSolution.hpp>
#include <stdair/basic/BasConst_SellUpCurves.hpp>
```

Namespaces

- namespace stdair

Functions

- const std::string stdair::DEFAULT_BOM_ROOT_KEY (" – ROOT – ")
- const double stdair::DEFAULT_EPSILON_VALUE (0.0001)
- const unsigned int stdair::DEFAULT_FLIGHT_SPEED (900)
- const NbOfFlightDates_T stdair::DEFAULT_NB_OF_FLIGHTDATES (0.0)
- const Duration_T stdair::NULL_BOOST_TIME_DURATION (-1,-1,-1)
- const Duration_T stdair::DEFAULT_NULL_DURATION (0, 0, 0)
- const unsigned int stdair::DEFAULT_NB_OF_DAYS_IN_A_YEAR (365)
- const unsigned int stdair::DEFAULT_NUMBER_OF_SUBDIVISIONS (1000)
- const DayDuration_T stdair::DEFAULT_DAY_DURATION (0)
- const DatePeriod_T stdair::BOOST_DEFAULT_DATE_PERIOD (Date_T(2007, 1, 1), Date_T(2007, 1, 1))
- const DOW_String_T stdair::DEFAULT_DOW_STRING ("0000000")
- const DateOffset_T stdair::DEFAULT_DATE_OFFSET (0)
- const Date_T stdair::DEFAULT_DATE (2010, boost::gregorian::Jan, 1)
- const DateTime_T stdair::DEFAULT_DATETIME (DEFAULT_DATE, NULL_BOOST_TIME_DURATION)
- const Duration_T stdair::DEFAULT_EPSILON_DURATION (0, 0, 0, 1)
- const Count_T stdair::SECONDS_IN_ONE_DAY (86400)
- const Count_T stdair::MILLISECONDS_IN_ONE_SECOND (1000)
- const RandomSeed_T stdair::DEFAULT_RANDOM_SEED (120765987)
- const AirportCode_T stdair::AIRPORT_LHR ("LHR")
- const AirportCode_T stdair::AIRPORT_SYD ("SYD")

- const CityCode_T stdair::POS_LHR ("LHR")
- const Date_T stdair::DATE_20110115 (2011, boost::gregorian::Jan, 15)
- const Date_T stdair::DATE_20111231 (2011, boost::gregorian::Dec, 31)
- const DayDuration_T stdair::NO_ADVANCE_PURCHASE (0)
- const SaturdayStay_T stdair::SATURDAY_STAY (true)
- const SaturdayStay_T stdair::NO_SATURDAY_STAY (false)
- const ChangeFees_T stdair::CHANGE_FEES (true)
- const ChangeFees_T stdair::NO_CHANGE_FEES (false)
- const NonRefundable_T stdair::NON_REFUNDABLE (true)
- const NonRefundable_T stdair::NO_NON_REFUNDABLE (false)
- const SaturdayStay_T stdair::DEFAULT_BOM_TREE_SATURDAY_STAY (true)
- const ChangeFees_T stdair::DEFAULT_BOM_TREE_CHANGE_FEES (true)
- const NonRefundable_T stdair::DEFAULT_BOM_TREE_NON_REFUNDABLE (true)
- const DayDuration_T stdair::NO_STAY_DURATION (0)
- const AirlineCode_T stdair::AIRLINE_CODE_BA ("BA")
- const CabinCode_T stdair::CABIN_Y ("Y")
- const ClassCode_T stdair::CLASS_CODE_Y ("Y")
- const ClassCode_T stdair::CLASS_CODE_Q ("Q")
- const AirportCode_T stdair::AIRPORT_SIN ("SIN")
- const AirportCode_T stdair::AIRPORT_BKK ("BKK")
- const CityCode_T stdair::POS_SIN ("SIN")
- const CabinCode_T stdair::CABIN_ECO ("Eco")
- const FrequentFlyer_T stdair::FREQUENT_FLYER_MEMBER ("M")
- const FamilyCode_T stdair::DEFAULT_FAMILY_CODE ("0")
- const PolicyCode_T stdair::DEFAULT_POLICY_CODE ("0")
- const NestingStructureCode_T stdair::DEFAULT_NESTING_STRUCTURE_CODE ("DEFAULT")
- const NestingStructureCode_T stdair::DISPLAY_NESTING_STRUCTURE_CODE ("Display Nesting")
- const NestingStructureCode_T stdair::YIELD_BASED_NESTING_STRUCTURE_CODE ("Yield-Based Nesting")
- const NestingNodeCode_T stdair::DEFAULT_NESTING_NODE_CODE ("0")
- const NbOfAirlines_T stdair::DEFAULT_NBOFAIRLINES (0)
- const FlightPathCode_T stdair::DEFAULT_FLIGHTPATH_CODE ("")
- const Distance_T stdair::DEFAULT_DISTANCE_VALUE (0)
- const ClassCode_T stdair::DEFAULT_CLOSED_CLASS_CODE ("CC")
- const NbOfBookings_T stdair::DEFAULT_CLASS_NB_OF_BOOKINGS (0)
- const NbOfBookings_T stdair::DEFAULT_CLASS_TOTAL_NB_OF_BOOKINGS (0)
- const NbOfBookings_T stdair::DEFAULT_CLASS_UNCONSTRAINED_DEMAND (0)
- const NbOfBookings_T stdair::DEFAULT_CLASS_REMAINING_DEMAND_MEAN (0)
- const NbOfBookings_T stdair::DEFAULT_CLASS_REMAINING_DEMAND_STANDARD_-DEVIATION (0)
- const NbOfCancellations_T stdair::DEFAULT_CLASS_NB_OF_CANCELLATIONS (0)
- const NbOfNoShows_T stdair::DEFAULT_CLASS_NB_OF_NOSHOWS (0)
- const CabinCapacity_T stdair::DEFAULT_CABIN_CAPACITY (100.0)
- const CommittedSpace_T stdair::DEFAULT_COMMITTED_SPACE (0.0)
- const BlockSpace_T stdair::DEFAULT_BLOCK_SPACE (0.0)
- const Availability_T stdair::DEFAULT_NULL_AVAILABILITY (0.0)
- const Availability_T stdair::DEFAULT_AVAILABILITY (9.0)
- const Availability_T stdair::MAXIMAL_AVAILABILITY (9999.0)
- const CensorshipFlag_T stdair::DEFAULT_CLASS_CENSORSHIPFLAG (false)

- const BookingLimit_T stdair::DEFAULT_CLASS_BOOKING_LIMIT (9999.0)
- const AuthorizationLevel_T stdair::DEFAULT_CLASS_AUTHORIZATION_LEVEL (9999.0)
- const AuthorizationLevel_T stdair::DEFAULT_CLASS_MAX_AUTHORIZATION_LEVEL (9999.0)
- const AuthorizationLevel_T stdair::DEFAULT_CLASS_MIN_AUTHORIZATION_LEVEL (0.0)
- const OverbookingRate_T stdair::DEFAULT_CLASS_OVERBOOKING_RATE (0.0)
- const BookingRatio_T stdair::DEFAULT_OND_BOOKING_RATE (0.0)
- const Fare_T stdair::DEFAULT_FARE_VALUE (0.0)
- const Yield_T stdair::DEFAULT_CLASS_YIELD_VALUE (0.0)
- const Revenue_T stdair::DEFAULT_REVENUE_VALUE (0.0)
- const Percentage_T stdair::DEFAULT_LOAD_FACTOR_VALUE (100.0)
- const Yield_T stdair::DEFAULT_YIELD_VALUE (0.0)
- const Yield_T stdair::DEFAULT_YIELD_MAX_VALUE (std::numeric_limits< double >::max())
- const NbOfBookings_T stdair::DEFAULT_YIELD_NB_OF_BOOKINGS (0.0)
- const Identity_T stdair::DEFAULT_BOOKING_NUMBER (0)
- const NbOfCancellations_T stdair::DEFAULT_YIELD_NB_OF_CANCELLATIONS (0.0)
- const NbOfNoShows_T stdair::DEFAULT_YIELD_NB_OF_NOSHOWS (0.0)
- const Availability_T stdair::DEFAULT_YIELD_AVAILABILITY (0.0)
- const CensorshipFlag_T stdair::DEFAULT_YIELD_CENSORSHIPFLAG (false)
- const BookingLimit_T stdair::DEFAULT_YIELD_BOOKING_LIMIT (0.0)
- const OverbookingRate_T stdair::DEFAULT_YIELD_OVERBOOKING_RATE (0.0)
- const Fare_T stdair::DEFAULT_OND_FARE_VALUE (0.0)
- const Count_T stdair::DEFAULT_PROGRESS_STATUS (0)
- const Percentage_T stdair::MAXIMUM_PROGRESS_STATUS (100)
- const Date_T stdair::DEFAULT_EVENT_OLDEST_DATE (2008, boost::gregorian::Jan, 1)
- const DateTime_T stdair::DEFAULT_EVENT_OLDEST_DATETIME (DEFAULT_EVENT_-OLDEST_DATE, NULL_BOOST_TIME_DURATION)
- const PartySize_T stdair::DEFAULT_PARTY_SIZE (1)
- const DayDuration_T stdair::DEFAULT_STAY_DURATION (7)
- const WTP_T stdair::DEFAULT_WTP (1000.0)
- const Date_T stdair::DEFAULT_PREFERRED_DEPARTURE_DATE (DEFAULT_DEPARTURE_-DATE)
- const Duration_T stdair::DEFAULT_PREFERRED_DEPARTURE_TIME (8, 0, 0)
- const DateOffset_T stdair::DEFAULT_ADVANCE_PURCHASE (22)
- const Date_T stdair::DEFAULT_REQUEST_DATE (DEFAULT_PREFERRED_DEPARTURE_-DATE-DEFAULT_ADVANCE_PURCHASE)
- const Duration_T stdair::DEFAULT_REQUEST_TIME (8, 0, 0)
- const DateTime_T stdair::DEFAULT_REQUEST_DATE_TIME (DEFAULT_REQUEST_DATE, DEFAULT_REQUEST_TIME)
- const CabinCode_T stdair::DEFAULT_PREFERRED_CABIN ("M")
- const CityCode_T stdair::DEFAULT_POS ("ALL")
- const ChannelLabel_T stdair::DEFAULT_CHANNEL ("DC")
- const ChannelLabel_T stdair::CHANNEL_DN ("DN")
- const ChannelLabel_T stdair::CHANNEL_IN ("IN")
- const TripType_T stdair::TRIP_TYPE_ONE_WAY ("OW")
- const TripType_T stdair::TRIP_TYPE_ROUND_TRIP ("RT")
- const TripType_T stdair::TRIP_TYPE_INBOUND ("RI")
- const TripType_T stdair::TRIP_TYPE_OUTBOUND ("RO")
- const FrequentFlyer_T stdair::DEFAULT_FF_TIER ("N")
- const PriceValue_T stdair::DEFAULT_VALUE_OF_TIME (100.0)

- const IntDuration_T stdair::HOUR_CONVERTED_IN_SECONDS (3600)
- const Duration_T stdair::DEFAULT_MINIMAL_CONNECTION_TIME (0, 30, 0)
- const Duration_T stdair::DEFAULT_MAXIMAL_CONNECTION_TIME (24, 0, 0)
- const MatchingIndicator_T stdair::DEFAULT_MATCHING_INDICATOR (0.0)
- const PriceCurrency_T stdair::DEFAULT_CURRENCY ("EUR")
- const AvailabilityStatus_T stdair::DEFAULT_AVAILABILITY_STATUS (false)
- const AirlineCode_T stdair::DEFAULT_AIRLINE_CODE ("XX")
- const AirlineCode_T stdair::DEFAULT_NULL_AIRLINE_CODE ("")
- const FlightNumber_T stdair::DEFAULT_FLIGHT_NUMBER (9999)
- const FlightNumber_T stdair::DEFAULT_FLIGHT_NUMBER_FF (255)
- const TableID_T stdair::DEFAULT_TABLE_ID (9999)
- const Date_T stdair::DEFAULT_DEPARTURE_DATE (1900, boost::gregorian::Jan, 1)
- const AirportCode_T stdair::DEFAULT_AIRPORT_CODE ("XXX")
- const AirportCode_T stdair::DEFAULT_NULL_AIRPORT_CODE ("")
- const AirportCode_T stdair::DEFAULT_ORIGIN ("XXX")
- const AirportCode_T stdair::DEFAULT_DESTINATION ("YYY")
- const CabinCode_T stdair::DEFAULT_CABIN_CODE ("X")
- const FamilyCode_T stdair::DEFAULT_FARE_FAMILY_CODE ("EcoSaver")
- const FamilyCode_T stdair::DEFAULT_NULL_FARE_FAMILY_CODE ("NoFF")
- const ClassCode_T stdair::DEFAULT_CLASS_CODE ("X")
- const ClassCode_T stdair::DEFAULT_NULL_CLASS_CODE ("")
- const BidPrice_T stdair::DEFAULT_BID_PRICE (0.0)
- const unsigned short stdair::MAXIMAL_NUMBER_OF_LEGS_IN_FLIGHT (7)
- const unsigned short stdair::MAXIMAL_NUMBER_OF_SEGMENTS_IN_OND (3)
- const SeatIndex_T stdair::DEFAULT_SEAT_INDEX (1)
- const NbOfSeats_T stdair::DEFAULT_NULL_BOOKING_NUMBER (0)
- const CapacityAdjustment_T stdair::DEFAULT_NULL_CAPACITY_ADJUSTMENT (0)
- const UPR_T stdair::DEFAULT_NULL_UPR (0)
- const std::string stdair::DEFAULT_FARE_FAMILY_VALUE_TYPE ("FF")
- const std::string stdair::DEFAULT_SEGMENT_CABIN_VALUE_TYPE ("SC")
- const std::string stdair::DEFAULT_KEY_FLD_DELIMITER (";")
- const std::string stdair::DEFAULT_KEY_SUB_FLD_DELIMITER (",")
- const boost::char_separator< char > stdair::DEFAULT_KEY_TOKEN_DELIMITER (";, ")

Variables

- const std::string stdair::DOW_STR [ ]
- const UnconstrainingMethod stdair::DEFAULT_UNCONSTRAINING_METHOD ('E')
- const PartnershipTechnique stdair::DEFAULT_PARTNERSHIP_TECHNIQUE ('N')
- const ForecastingMethod stdair::DEFAULT_FORECASTING_METHOD ('Q')
- const PreOptimisationMethod stdair::DEFAULT_PREOPTIMISATION_METHOD ('N')
- const OptimisationMethod stdair::DEFAULT_OPTIMISATION_METHOD ('M')
- const CensorshipFlagList_T stdair::DEFAULT_CLASS_CENSORSHIPFLAG_LIST
- const Date_T stdair::DEFAULT_DICO_STUDIED_DATE
- const AirlineCodeList_T stdair::DEFAULT_AIRLINE_CODE_LIST
- const ClassList_StringList_T stdair::DEFAULT_CLASS_CODE_LIST
- const BidPriceVector_T stdair::DEFAULT_BID_PRICE_VECTOR = std::vector<BidPrice_T>()
- const int stdair::DEFAULT_MAX_DTD = 365
- const DCPList_T stdair::DEFAULT_DCP_LIST = DefaultDCPList::init()
- const FRAT5Curve_T stdair::FRAT5_CURVE_A

- const FRAT5Curve_T stdair::FRAT5_CURVE_B
- const FRAT5Curve_T stdair::FRAT5_CURVE_C
- const FRAT5Curve_T stdair::FRAT5_CURVE_D
- const FFDisutilityCurve_T stdair::FF_DISUTILITY_CURVE_A
- const FFDisutilityCurve_T stdair::FF_DISUTILITY_CURVE_B
- const FFDisutilityCurve_T stdair::FF_DISUTILITY_CURVE_C
- const FFDisutilityCurve_T stdair::FF_DISUTILITY_CURVE_D
- const FFDisutilityCurve_T stdair::FF_DISUTILITY_CURVE_E
- const FFDisutilityCurve_T stdair::FF_DISUTILITY_CURVE_F
- const DTDFratMap_T stdair::DEFAULT_DTD_FRAT5COEF_MAP
- const DTDProbMap_T stdair::DEFAULT_DTD_PROB_MAP
- const OnDStringList_T stdair::DEFAULT_OND_STRING_LIST
- const std::string stdair::DISPLAY_LEVEL_STRING_ARRAY [51]

9.19 stdair/basic/BasConst_BomDisplay.hpp File Reference

```
#include <string>
#include <boost/tokenizer.hpp>
```

Namespaces

- namespace stdair

Variables

- const std::string stdair::DISPLAY_LEVEL_STRING_ARRAY [51]
- const std::string stdair::DEFAULT_KEY_FLD_DELIMITER
- const std::string stdair::DEFAULT_KEY_SUB_FLD_DELIMITER
- const boost::char_separator< char > stdair::DEFAULT_KEY_TOKEN_DELIMITER

9.20 stdair/basic/BasConst_BookingClass.hpp File Reference

```
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/stdair_fare_types.hpp>
```

Namespaces

- namespace stdair

Variables

- const Distance_T stdair::DEFAULT_DISTANCE_VALUE
- const ClassCode_T stdair::DEFAULT_CLOSED_CLASS_CODE
- const NbOfBookings_T stdair::DEFAULT_CLASS_NB_OF_BOOKINGS
- const NbOfBookings_T stdair::DEFAULT_CLASS_TOTAL_NB_OF_BOOKINGS
- const NbOfBookings_T stdair::DEFAULT_CLASS_UNCONSTRAINED_DEMAND
- const NbOfBookings_T stdair::DEFAULT_CLASS_REMAINING_DEMAND_MEAN
- const NbOfBookings_T stdair::DEFAULT_CLASS_REMAINING_DEMAND_STANDARD_-DEVIATION
- const NbOfCancellations_T stdair::DEFAULT_CLASS_NB_OF_CANCELLATIONS
- const NbOfNoShows_T stdair::DEFAULT_CLASS_NB_OF_NOSHOWS
- const CabinCapacity_T stdair::DEFAULT_CABIN_CAPACITY
- const CommittedSpace_T stdair::DEFAULT_COMMITTED_SPACE
- const BlockSpace_T stdair::DEFAULT_BLOCK_SPACE
- const Availability_T stdair::DEFAULT_NULL_AVAILABILITY
- const Availability_T stdair::DEFAULT_AVAILABILITY
- const CensorshipFlag_T stdair::DEFAULT_CLASS_CENSORSHIPFLAG
- const CensorshipFlagList_T stdair::DEFAULT_CLASS_CENSORSHIPFLAG_LIST
- const BookingLimit_T stdair::DEFAULT_CLASS_BOOKING_LIMIT
- const AuthorizationLevel_T stdair::DEFAULT_CLASS_AUTHORIZATION_LEVEL
- const AuthorizationLevel_T stdair::DEFAULT_CLASS_MAX_AUTHORIZATION_LEVEL
- const AuthorizationLevel_T stdair::DEFAULT_CLASS_MIN_AUTHORIZATION_LEVEL
- const OverbookingRate_T stdair::DEFAULT_CLASS_OVERBOOKING_RATE
- const Fare_T stdair::DEFAULT_FARE_VALUE
- const Revenue_T stdair::DEFAULT_REVENUE_VALUE
- const PriceCurrency_T stdair::DEFAULT_CURRENCY
- const Percentage_T stdair::DEFAULT_LOAD_FACTOR_VALUE
- const DayDuration_T stdair::DEFAULT_DAY_DURATION
- const double stdair::DEFAULT_EPSILON_VALUE

9.21 stdair/basic/BasConst_DefaultObject.hpp File Reference

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

Namespaces

- namespace stdair

Variables

- const AirportCode_T stdair::AIRPORT_LHR
- const AirportCode_T stdair::AIRPORT_SYD
- const CityCode_T stdair::POS_LHR
- const DayDuration_T stdair::NO_ADVANCE_PURCHASE
- const SaturdayStay_T stdair::SATURDAY_STAY
- const SaturdayStay_T stdair::NO_SATURDAY_STAY
- const ChangeFees_T stdair::CHANGE_FEES
- const ChangeFees_T stdair::NO_CHANGE_FEES

- const NonRefundable_T stdair::NON_REFUNDABLE
- const NonRefundable_T stdair::NO_NON_REFUNDABLE
- const DayDuration_T stdair::NO_STAY_DURATION
- const CabinCode_T stdair::CABIN_Y
- const AirlineCode_T stdair::AIRLINE_CODE_BA
- const ClassCode_T stdair::CLASS_CODE_Y
- const ClassCode_T stdair::CLASS_CODE_Q
- const AirportCode_T stdair::AIRPORT_SIN
- const AirportCode_T stdair::AIRPORT_BKK
- const CityCode_T stdair::POS_SIN
- const CabinCode_T stdair::CABIN_ECO
- const FrequentFlyer_T stdair::FREQUENT_FLYER_MEMBER

9.22 stdair/basic/BasConst_Event.hpp File Reference

```
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/stdair_event_types.hpp>
```

Namespaces

- namespace stdair

Variables

- const Count_T stdair::DEFAULT_PROGRESS_STATUS
- const Date_T stdair::DEFAULT_EVENT_OLDEST_DATE
- const DateTime_T stdair::DEFAULT_EVENT_OLDEST_DATETIME
- const Percentage_T stdair::MAXIMUM_PROGRESS_STATUS

9.23 stdair/basic/BasConst_General.hpp File Reference

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

Namespaces

- namespace stdair

Variables

- const std::string stdair::DEFAULT_BOM_ROOT_KEY
- const double stdair::DEFAULT_EPSILON_VALUE
- const CabinCapacity_T stdair::DEFAULT_CABIN_CAPACITY
- const NbOfFlightDates_T stdair::DEFAULT_NB_OF_FLIGHTDATES
- const NbOfBookings_T stdair::DEFAULT_CLASS_NB_OF_BOOKINGS
- const Distance_T stdair::DEFAULT_DISTANCE_VALUE

- const unsigned int stdair::DEFAULT_FLIGHT_SPEED
- const Fare_T stdair::DEFAULT_FARE_VALUE
- const PriceCurrency_T stdair::DEFAULT_CURRENCY
- const Revenue_T stdair::DEFAULT_REVENUE_VALUE
- const BookingRatio_T stdair::DEFAULT_OND_BOOKING_RATE
- const Count_T stdair::SECONDS_IN_ONE_DAY
- const Count_T stdair::MILLISECONDS_IN_ONE_SECOND
- const Date_T stdair::DEFAULT_DATE
- const DateTime_T stdair::DEFAULT_DATETIME
- const Duration_T stdair::DEFAULT_EPSILON_DURATION
- const RandomSeed_T stdair::DEFAULT_RANDOM_SEED
- const Duration_T stdair::NULL_BOOST_TIME_DURATION
- const Duration_T stdair::DEFAULT_NULL_DURATION
- const Fare_T stdair::DEFAULT_CLASS_FARE_VALUE
- const NbOfAirlines_T stdair::DEFAULT_NBOFAIRLINES
- const unsigned int stdair::DEFAULT_NB_OF_DAYS_IN_A_YEAR
- const NbOfBookings_T stdair::DEFAULT_CLASS_NB_OF_BOOKINGS
- const ChannelLabel_T stdair::DEFAULT_CHANNEL
- const OnDStringList_T stdair::DEFAULT_OND_STRING_LIST
- const unsigned int stdair::DEFAULT_NUMBER_OF_SUBDIVISIONS

9.24 stdair/basic/BasConst_Inventory.hpp File Reference

```
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/basic/ForecastingMethod.hpp>
#include <stdair/basic/UnconstrainingMethod.hpp>
#include <stdair/basic/PreOptimisationMethod.hpp>
#include <stdair/basic/OptimisationMethod.hpp>
#include <stdair/basic/PartnershipTechnique.hpp>
```

Namespaces

- namespace stdair

Classes

- struct stdair::DefaultDCPList
- struct stdair::DefaultDtdFratMap
- struct stdair::DefaultDtdProbMap

Variables

- const AirlineCode_T stdair::DEFAULT_AIRLINE_CODE
- const AirlineCode_T stdair::DEFAULT_NULL_AIRLINE_CODE
- const AirlineCodeList_T stdair::DEFAULT_AIRLINE_CODE_LIST

- const FlightNumber_T stdair::DEFAULT_FLIGHT_NUMBER
- const FlightNumber_T stdair::DEFAULT_FLIGHT_NUMBER_FF
- const TableID_T stdair::DEFAULT_TABLE_ID
- const Date_T stdair::DEFAULT_DEPARTURE_DATE
- const AirportCode_T stdair::DEFAULT_AIRPORT_CODE
- const AirportCode_T stdair::DEFAULT_NULL_AIRPORT_CODE
- const AirportCode_T stdair::DEFAULT_ORIGIN
- const AirportCode_T stdair::DEFAULT_DESTINATION
- const CabinCode_T stdair::DEFAULT_CABIN_CODE
- const FamilyCode_T stdair::DEFAULT_FARE_FAMILY_CODE
- const FamilyCode_T stdair::DEFAULT_NULL_FARE_FAMILY_CODE
- const PolicyCode_T stdair::DEFAULT_POLICY_CODE
- const NestingStructureCode_T stdair::DEFAULT_NESTING_STRUCTURE_CODE
- const NestingStructureCode_T stdair::DISPLAY_NESTING_STRUCTURE_CODE
- const NestingStructureCode_T stdair::YIELD_BASED_NESTING_STRUCTURE_CODE
- const NestingNodeCode_T stdair::DEFAULT_NESTING_NODE_CODE
- const ClassCode_T stdair::DEFAULT_CLASS_CODE
- const ClassCode_T stdair::DEFAULT_NULL_CLASS_CODE
- const ClassList_StringList_T stdair::DEFAULT_CLASS_CODE_LIST
- const BidPrice_T stdair::DEFAULT_BID_PRICE
- const BidPriceVector_T stdair::DEFAULT_BID_PRICE_VECTOR
- const unsigned short stdair::MAXIMAL_NUMBER_OF_LEGS_IN_FLIGHT
- const unsigned short stdair::MAXIMAL_NUMBER_OF_SEGMENTS_IN_OND
- const Availability_T stdair::MAXIMAL_AVAILABILITY
- const SeatIndex_T stdair::DEFAULT_SEAT_INDEX
- const NbOfSeats_T stdair::DEFAULT_NULL_BOOKING_NUMBER
- const CapacityAdjustment_T stdair::DEFAULT_NULL_CAPACITY_ADJUSTMENT
- const UPR_T stdair::DEFAULT_NULL_UPR
- const std::string stdair::DEFAULT_FARE_FAMILY_VALUE_TYPE
- const std::string stdair::DEFAULT_SEGMENT_CABIN_VALUE_TYPE
- const int stdair::DEFAULT_MAX_DTD
- const DCPList_T stdair::DEFAULT_DCP_LIST
- const DTDFratMap_T stdair::DEFAULT_DTD_FRAT5COEF_MAP
- const DTDProbMap_T stdair::DEFAULT_DTD_PROB_MAP
- const ForecastingMethod stdair::DEFAULT_FORECASTING_METHOD
- const UnconstrainingMethod stdair::DEFAULT_UNCONSTRAINING_METHOD
- const PreOptimisationMethod stdair::DEFAULT_PREOPTIMISATION_METHOD
- const OptimisationMethod stdair::DEFAULT_OPTIMISATION_METHOD
- const PartnershipTechnique stdair::DEFAULT_PARTNERSHIP_TECHNIQUE

9.25 stdair/basic/BasConst_Period_BOM.hpp File Reference

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

Namespaces

- namespace stdair

Variables

- const DatePeriod_T stdair::BOOST_DEFAULT_DATE_PERIOD
- const std::string stdair::DOW_STR [ ]
- const DOW_String_T stdair::DEFAULT_DOW_STRING
- const DateOffset_T stdair::DEFAULT_DATE_OFFSET
- const DayDuration_T stdair::DEFAULT_DAY_DURATION

9.26 stdair/basic/BasConst_Request.hpp File Reference

```
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
```

Namespaces

- namespace stdair

Variables

- const PartySize_T stdair::DEFAULT_PARTY_SIZE
- const DayDuration_T stdair::DEFAULT_STAY_DURATION
- const WTP_T stdair::DEFAULT_WTP
- const CityCode_T stdair::DEFAULT_POS
- const Date_T stdair::DEFAULT_PREFERRED_DEPARTURE_DATE
- const Duration_T stdair::DEFAULT_PREFERRED_DEPARTURE_TIME
- const DateOffset_T stdair::DEFAULT_ADVANCE_PURCHASE
- const Date_T stdair::DEFAULT_REQUEST_DATE
- const Duration_T stdair::DEFAULT_REQUEST_TIME
- const DateTime_T stdair::DEFAULT_REQUEST_DATE_TIME
- const CabinCode_T stdair::DEFAULT_PREFERRED_CABIN
- const ChannelLabel_T stdair::DEFAULT_CHANNEL
- const ChannelLabel_T stdair::CHANNEL_DN
- const ChannelLabel_T stdair::CHANNEL_IN
- const TripType_T stdair::TRIP_TYPE_ONE_WAY
- const TripType_T stdair::TRIP_TYPE_ROUND_TRIP
- const TripType_T stdair::TRIP_TYPE_INBOUND
- const TripType_T stdair::TRIP_TYPE_OUTBOUND
- const FrequentFlyer_T stdair::DEFAULT_FF_TIER
- const PriceValue_T stdair::DEFAULT_VALUE_OF_TIME
- const IntDuration_T stdair::HOUR_CONVERTED_IN_SECONDS

9.27 stdair/basic/BasConst_SellUpCurves.hpp File Reference

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

Namespaces

- namespace stdair

Classes

- struct stdair::DefaultMap

Variables

- const FRAT5Curve_T stdair::FRAT5_CURVE_A
- const FRAT5Curve_T stdair::FRAT5_CURVE_B
- const FRAT5Curve_T stdair::FRAT5_CURVE_C
- const FRAT5Curve_T stdair::FRAT5_CURVE_D
- const FFDisutilityCurve_T stdair::FF_DISUTILITY_CURVE_A
- const FFDisutilityCurve_T stdair::FF_DISUTILITY_CURVE_B
- const FFDisutilityCurve_T stdair::FF_DISUTILITY_CURVE_C
- const FFDisutilityCurve_T stdair::FF_DISUTILITY_CURVE_D
- const FFDisutilityCurve_T stdair::FF_DISUTILITY_CURVE_E
- const FFDisutilityCurve_T stdair::FF_DISUTILITY_CURVE_F

9.28 stdair/basic/BasConst_TravelSolution.hpp File Reference

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

Namespaces

- namespace stdair

Variables

- const Distance_T stdair::DEFAULT_DISTANCE_VALUE
- const Duration_T stdair::DEFAULT_MINIMAL_CONNECTION_TIME
- const Duration_T stdair::DEFAULT_MAXIMAL_CONNECTION_TIME
- const Duration_T stdair::NULL_BOOST_TIME_DURATION
- const FlightPathCode_T stdair::DEFAULT_FLIGHTPATH_CODE
- const Availability_T stdair::DEFAULT_CLASS_AVAILABILITY
- const AvailabilityStatus_T stdair::DEFAULT_AVAILABILITY_STATUS
- const unsigned short stdair::DEFAULT_NUMBER_OF_REQUIRED_SEATS
- const MatchingIndicator_T stdair::DEFAULT_MATCHING_INDICATOR
- const Revenue_T stdair::DEFAULT_REVENUE_VALUE
- const AirlineCode_T stdair::DEFAULT_DICO_STUDIED_AIRLINE
- const Date_T stdair::DEFAULT_DICO_STUDIED_DATE

9.29 stdair/basic/BasConst_Yield.hpp File Reference

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

Namespaces

- namespace stdair

Variables

- const Yield_T stdair::DEFAULT_YIELD_VALUE
- const Yield_T stdair::DEFAULT_YIELD_MAX_VALUE

9.30 stdair/basic/BasDBParams.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasDBParams.hpp>
```

Namespaces

- namespace stdair

9.31 stdair/basic/BasDBParams.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_db.hpp>
#include <stdair/basic/StructAbstract.hpp>
```

Namespaces

- namespace stdair

Classes

- struct stdair::BasDBParams

  Structure holding the parameters for connection to a database.

9.32 stdair/basic/BasFileMgr.cpp File Reference

```
#include <cassert>
#include <boost/version.hpp>
#include <boost/filesystem/path.hpp>
#include <boost/filesystem/operations.hpp>
#include <stdair/basic/BasFileMgr.hpp>
```

Namespaces

- namespace stdair

9.33 stdair/basic/BasFileMgr.hpp File Reference

```
#include <string>
```

Namespaces

- namespace stdair

Classes

- struct stdair::BasFileMgr

9.34 stdair/basic/BasLogParams.cpp File Reference

```
#include <cassert>
#include <iostream>
#include <sstream>
#include <stdair/basic/BasLogParams.hpp>
```

Namespaces

- namespace stdair

9.35 stdair/basic/BasLogParams.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_log.hpp>
#include <stdair/basic/StructAbstract.hpp>
```

Namespaces

- namespace stdair

Classes

- struct stdair::BasLogParams

  Structure holding parameters for logging.

9.36 stdair/basic/BasParserHelperTypes.hpp File Reference

```
#include <string>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/service/Logger.hpp>
```

Namespaces

- namespace stdair

Classes

- struct stdair::date_time_element< MIN, MAX >

Typedefs

- typedef date_time_element< 0, 23 > stdair::hour_t
- typedef date_time_element< 0, 59 > stdair::minute_t
- typedef date_time_element< 0, 59 > stdair::second_t
- typedef date_time_element< 1900, 2100 > stdair::year_t
- typedef date_time_element< 1, 12 > stdair::month_t
- typedef date_time_element< 1, 31 > stdair::day_t

Functions

- template<int MIN, int MAX> date_time_element< MIN, MAX > stdair::operator * (const date_time_element< MIN, MAX > &o1, const date_time_element< MIN, MAX > &o2)
- template<int MIN, int MAX> date_time_element< MIN, MAX > stdair::operator+ (const date_time_element< MIN, MAX > &o1, const date_time_element< MIN, MAX > &o2)

9.37 stdair/basic/BasParserTypes.hpp File Reference

```
#include <string>
#include <boost/spirit/include/qi.hpp>
#include <boost/spirit/include/phoenix_core.hpp>
#include <boost/spirit/include/phoenix_operator.hpp>
#include <boost/spirit/include/support_multi_pass.hpp>
#include <stdair/basic/BasParserHelperTypes.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef std::istreambuf_iterator< char > stdair::base_iterator_t
- typedef boost::spirit::multi_pass< base_iterator_t > stdair::iterator_t
- typedef boost::spirit::qi::int_parser< unsigned int, 10, 1, 1 > stdair::int1_p_t
- typedef boost::spirit::qi::uint_parser< int, 10, 2, 2 > stdair::uint2_p_t
- typedef boost::spirit::qi::uint_parser< int, 10, 4, 4 > stdair::uint4_p_t
- typedef boost::spirit::qi::uint_parser< int, 10, 1, 4 > stdair::uint1_4_p_t
- typedef boost::spirit::qi::uint_parser< hour_t, 10, 2, 2 > stdair::hour_p_t
- typedef boost::spirit::qi::uint_parser< minute_t, 10, 2, 2 > stdair::minute_p_t
- typedef boost::spirit::qi::uint_parser< second_t, 10, 2, 2 > stdair::second_p_t
- typedef boost::spirit::qi::uint_parser< year_t, 10, 4, 4 > stdair::year_p_t
- typedef boost::spirit::qi::uint_parser< month_t, 10, 2, 2 > stdair::month_p_t
- typedef boost::spirit::qi::uint_parser< day_t, 10, 2, 2 > stdair::day_p_t

9.38 stdair/basic/BasTypes.hpp File Reference

```
#include <string>
```

Namespaces

- namespace stdair

9.39 stdair/basic/ContinuousAttributeLite.hpp File Reference

```
#include <cassert>
#include <iosfwd>
#include <string>
#include <vector>
#include <map>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/DictionaryManager.hpp>
```

Namespaces

- namespace stdair

Classes

- struct stdair::ContinuousAttributeLite< T >

  Class modeling the distribution of values that can be taken by a continuous attribute.

9.40 stdair/basic/DemandGenerationMethod.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/DemandGenerationMethod.hpp>
```

Namespaces

- namespace stdair

9.41 stdair/basic/DemandGenerationMethod.hpp File Reference

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

Namespaces

- namespace stdair

Classes

- struct stdair::DemandGenerationMethod

  Enumeration of demand (booking request) generation methods.

9.42 stdair/basic/DictionaryManager.cpp File Reference

```
#include <stdair/basic/DictionaryManager.hpp>
#include <stdair/basic/BasConst_General.hpp>
```

Namespaces

- namespace stdair

9.43 stdair/basic/DictionaryManager.hpp File Reference

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

Namespaces

- namespace stdair

Classes

- class stdair::DictionaryManager

  Class wrapper of dictionary business methods.

Typedefs

- typedef unsigned short stdair::DictionaryKey_T

9.44 stdair/basic/EventType.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/EventType.hpp>
```

Namespaces

- namespace stdair

9.45 stdair/basic/EventType.hpp File Reference

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

Namespaces

- namespace stdair

Classes

- struct stdair::EventType

9.46 stdair/basic/float_utils.hpp File Reference

```
#include <stdair/basic/float_utils_google.hpp>
```

Namespaces

- namespace stdair

9.47 stdair/basic/float_utils_google.hpp File Reference

Classes

- class TypeWithSize< size >
- class TypeWithSize< 4 >
- class TypeWithSize< 8 >
- class FloatingPoint< RawType >

9.48 stdair/basic/ForecastingMethod.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/ForecastingMethod.hpp>
```

Namespaces

- namespace stdair

9.49 stdair/basic/ForecastingMethod.hpp File Reference

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

Namespaces

- namespace stdair

Classes

- struct stdair::ForecastingMethod

9.50 stdair/basic/JSonCommand.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/JSonCommand.hpp>
```

Namespaces

- namespace stdair

9.51 stdair/basic/JSonCommand.hpp File Reference

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

Namespaces

- namespace stdair

Classes

- struct stdair::JSonCommand
  Enumeration of json commands.

9.52 stdair/basic/OptimisationMethod.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/OptimisationMethod.hpp>
```

Namespaces

- namespace stdair

9.53 stdair/basic/OptimisationMethod.hpp File Reference

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

Namespaces

- namespace stdair

Classes

- struct stdair::OptimisationMethod

9.54 stdair/basic/PartnershipTechnique.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/PartnershipTechnique.hpp>
```

Namespaces

- namespace stdair

9.55 stdair/basic/PartnershipTechnique.hpp File Reference

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

Namespaces

- namespace stdair

Classes

- struct stdair::PartnershipTechnique

  Enumeration of partnership techniques.

9.56 stdair/basic/PassengerChoiceModel.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/PassengerChoiceModel.hpp>
```

Namespaces

- namespace stdair

9.57 stdair/basic/PassengerChoiceModel.hpp File Reference

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

Namespaces

- namespace stdair

Classes

- struct stdair::PassengerChoiceModel

9.58 stdair/basic/PassengerType.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/PassengerType.hpp>
```

Namespaces

- namespace stdair

9.59 stdair/basic/PassengerType.hpp File Reference

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

Namespaces

- namespace stdair

Classes

- struct stdair::PassengerType

9.60 stdair/basic/PreOptimisationMethod.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/PreOptimisationMethod.hpp>
```

Namespaces

- namespace stdair

9.61 stdair/basic/PreOptimisationMethod.hpp File Reference

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

Namespaces

- namespace stdair

Classes

- struct stdair::PreOptimisationMethod

9.62 stdair/basic/ProgressStatus.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/BasConst_Event.hpp>
#include <stdair/basic/ProgressStatus.hpp>
```

Namespaces

- namespace stdair

9.63 stdair/basic/ProgressStatus.hpp File Reference

```
#include <string>
#include <boost/progress.hpp>
#include <stdair/basic/BasConst_Event.hpp>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/service/Logger.hpp>
```

Namespaces

- namespace stdair

Classes

- struct stdair::ProgressStatus

9.64 stdair/basic/ProgressStatusSet.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/ProgressStatusSet.hpp>
```

Namespaces

- namespace stdair

9.65 stdair/basic/ProgressStatusSet.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_event_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/basic/EventType.hpp>
#include <stdair/basic/ProgressStatus.hpp>
```

Namespaces

- namespace stdair

Classes

- struct stdair::ProgressStatusSet

9.66 stdair/basic/RandomGeneration.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/version.hpp>
#include <stdair/basic/RandomGeneration.hpp>
```

Namespaces

- namespace stdair

9.67 stdair/basic/RandomGeneration.hpp File Reference

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

Namespaces

- namespace stdair

Classes

- struct stdair::RandomGeneration

  Class holding a random generator.

9.68 stdair/basic/SampleType.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/SampleType.hpp>
```

Namespaces

- namespace stdair

9.69 stdair/basic/SampleType.hpp File Reference

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

Namespaces

- namespace stdair

Classes

- struct stdair::SampleType
  Enumeration of BOM sample types.

9.70 stdair/basic/ServiceInitialisationType.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/ServiceInitialisationType.hpp>
```

Namespaces

- namespace stdair

9.71 stdair/basic/ServiceInitialisationType.hpp File Reference

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

Namespaces

- namespace stdair

Classes

- struct stdair::ServiceInitialisationType

  Enumeration of service initialisation types.

9.72 stdair/basic/StructAbstract.hpp File Reference

```
#include <iosfwd>
#include <string>
```

Namespaces

- namespace stdair

Classes

- struct stdair::StructAbstract

  Base class for the light structures.

Functions

- template<class charT, class traits> std::basic_ostream< charT, traits > & operator<< (std::basic_ostream< charT, traits > &ioOut, const stdair::StructAbstract &iStruct)
- template<class charT, class traits> std::basic_istream< charT, traits > & operator>> (std::basic_istream< charT, traits > &ioIn, stdair::StructAbstract &ioStruct)

9.72.1 Function Documentation

9.72.1.1 template<class charT, class traits> std::basic_ostream<charT, traits>& operator<< (std::basic_ostream< charT, traits > & *ioOut*, const stdair::StructAbstract & *iStruct*) [inline]

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (p653) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 61 of file StructAbstract.hpp.

9.72.1.2 template<class charT, class traits> std::basic_istream<charT, traits>& operator>> (std::basic_istream< charT, traits > & *ioIn*, stdair::StructAbstract & *ioStruct*) [inline]

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (pp655-657) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 89 of file StructAbstract.hpp.

References stdair::StructAbstract::fromStream().

9.73 stdair/basic/UnconstrainingMethod.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/UnconstrainingMethod.hpp>
```

Namespaces

- namespace stdair

9.74 stdair/basic/UnconstrainingMethod.hpp File Reference

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

Namespaces

- namespace stdair

Classes

- struct stdair::UnconstrainingMethod

9.75 stdair/basic/YieldRange.cpp File Reference

```
#include <limits>
#include <sstream>
#include <stdair/basic/YieldRange.hpp>
```

Namespaces

- namespace stdair

9.76 stdair/basic/YieldRange.hpp File Reference

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

Namespaces

- namespace stdair

Classes

- class stdair::YieldRange

9.77 stdair/bom/AirlineClassList.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/AirlineClassList.hpp>
```

Namespaces

- namespace stdair

9.78 stdair/bom/AirlineClassList.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/AirlineClassListKey.hpp>
#include <stdair/bom/AirlineClassListTypes.hpp>
```

Namespaces

- namespace boost
- namespace boost::serialization
- namespace stdair

Classes

- class stdair::AirlineClassList

  Class representing the actual attributes for a segment-features.

9.79 stdair/bom/AirlineClassListKey.cpp File Reference

```
#include <cassert>
#include <sstream>
```

```
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/AirlineClassListKey.hpp>
```

Namespaces

- namespace stdair

Functions

- template void stdair::AirlineClassListKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void stdair::AirlineClassListKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)

9.80 stdair/bom/AirlineClassListKey.hpp File Reference

```
#include <iosfwd>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

Namespaces

- namespace boost
- namespace boost::serialization
- namespace stdair

Classes

- struct stdair::AirlineClassListKey

  Key of airport-pair.

9.81 stdair/bom/AirlineClassListTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef std::list< AirlineClassList * > stdair::AirlineClassListList_T
- typedef std::map< const MapKey_T, AirlineClassList * > stdair::AirlineClassListMap_T
- typedef std::pair< MapKey_T, AirlineClassList * > stdair::AirlineClassListWithKey_T
- typedef std::list< AirlineClassListWithKey_T > stdair::AirlineClassListDetailedList_T

9.82 stdair/bom/AirlineFeature.cpp File Reference

```
#include <cassert>
#include <stdair/stdair_types.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/AirlineFeature.hpp>
```

Namespaces

- namespace stdair

9.83 stdair/bom/AirlineFeature.hpp File Reference

```
#include <stdair/stdair_rm_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 <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/AirlineFeatureKey.hpp>
#include <stdair/bom/AirlineFeatureTypes.hpp>
```

Namespaces

- namespace stdair

Classes

- class stdair::AirlineFeature

  Class representing various configuration parameters (e.g., revenue management methods such EMSRb or Monte-Carlo) for a given airline for the simulation.

9.84 stdair/bom/AirlineFeatureKey.cpp File Reference

```
#include <sstream>
#include <stdair/bom/AirlineFeatureKey.hpp>
```

Namespaces

- namespace stdair

9.85 stdair/bom/AirlineFeatureKey.hpp File Reference

```
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

Namespaces

- namespace stdair

Classes

- struct stdair::AirlineFeatureKey

9.86 stdair/bom/AirlineFeatureTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef std::list< AirlineFeature * > stdair::AirlineFeatureList_T
- typedef std::map< const MapKey_T, AirlineFeature * > stdair::AirlineFeatureMap_T

9.87 stdair/bom/AirlineStruct.cpp File Reference

```
#include <cassert>
#include <istream>
#include <ostream>
#include <sstream>
#include <stdair/bom/AirlineStruct.hpp>
```

Namespaces

- namespace stdair

9.88 stdair/bom/AirlineStruct.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <vector>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
```

Namespaces

- namespace stdair

Classes

- struct stdair::AirlineStruct

9.89 stdair/bom/AirportPair.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/AirportPair.hpp>
```

Namespaces

- namespace stdair

9.90 stdair/bom/AirportPair.hpp File Reference

```
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/AirportPairKey.hpp>
#include <stdair/bom/AirportPairTypes.hpp>
```

Namespaces

- namespace stdair

Classes

- class stdair::AirportPair

  Class representing the actual attributes for an airport-pair.

9.91 stdair/bom/AirportPairKey.cpp File Reference

```
#include <ostream>
#include <sstream>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/AirportPairKey.hpp>
```

Namespaces

- namespace stdair

9.92 stdair/bom/AirportPairKey.hpp File Reference

```
#include <stdair/bom/KeyAbstract.hpp>
#include <stdair/stdair_basic_types.hpp>
```

Namespaces

- namespace stdair

Classes

- struct stdair::AirportPairKey

  Key of airport-pair.

9.93 stdair/bom/AirportPairTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef std::list< AirportPair * > stdair::AirportPairList_T
- typedef std::map< const MapKey_T, AirportPair * > stdair::AirportPairMap_T
- typedef std::pair< MapKey_T, AirportPair * > stdair::AirportPairWithKey_T
- typedef std::list< AirportPairWithKey_T > stdair::AirportPairDetailedList_T

9.94 stdair/bom/BomAbstract.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <map>
#include <typeinfo>
```

Namespaces

- namespace stdair

Classes

- class stdair::BomAbstract

  Base class for the Business Object Model (BOM) layer.

Typedefs

- typedef std::map< const std::type_info *, BomAbstract * > stdair::HolderMap_T

Functions

- template<class charT, class traits> std::basic_ostream< charT, traits > & operator<< (std::basic_ostream< charT, traits > &ioOut, const stdair::BomAbstract &iBom)
- template<class charT, class traits> std::basic_istream< charT, traits > & operator>> (std::basic_istream< charT, traits > &ioIn, stdair::BomAbstract &ioBom)

9.94.1 Function Documentation

9.94.1.1 template<class charT, class traits> std::basic_ostream<charT, traits>& operator<< (std::basic_ostream< charT, traits > & *ioOut*, const stdair::BomAbstract & *iBom*) [inline]

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (p653) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 74 of file BomAbstract.hpp.

9.94.1.2 template<class charT, class traits> std::basic_istream<charT, traits>& operator>> (std::basic_istream< charT, traits > & *ioIn*, stdair::BomAbstract & *ioBom*) [inline]

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (pp655-657) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 102 of file BomAbstract.hpp.

References stdair::BomAbstract::fromStream().

9.95 stdair/bom/BomArchive.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/tmpdir.hpp>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/base_object.hpp>
#include <boost/serialization/utility.hpp>
#include <boost/serialization/list.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/BookingRequestStruct.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomArchive.hpp>
```

Namespaces

- namespace stdair

9.96 stdair/bom/BomArchive.hpp File Reference

```
#include <iosfwd>
```

Namespaces

- namespace stdair

Classes

- class stdair::BomArchive

  Utility class to archive/restore BOM objects with Boost serialisation.

9.97 stdair/bom/BomDisplay.cpp File Reference

9.98 stdair/bom/BomDisplay.hpp File Reference

```
#include <iosfwd>
#include <stdair/bom/TravelSolutionTypes.hpp>
#include <stdair/bom/DatePeriodTypes.hpp>
```

Namespaces

- namespace stdair

Classes

- class stdair::BomDisplay

  Utility class to display StdAir objects with a pretty format.

9.99 stdair/bom/BomHolder.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <list>
#include <map>
#include <stdair/bom/key_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/BomHolderKey.hpp>
```

Namespaces

- namespace stdair

Classes

- class stdair::BomHolder< BOM >

  Class representing the holder of BOM object containers (list and map).

9.100 stdair/bom/BomHolderKey.cpp File Reference

```
#include <ostream>
#include <sstream>
#include <stdair/bom/BomHolderKey.hpp>
```

Namespaces

- namespace stdair

9.101 stdair/bom/BomHolderKey.hpp File Reference

```
#include <stdair/bom/KeyAbstract.hpp>
```

Namespaces

- namespace stdair

Classes

- struct stdair::BomHolderKey

9.102 stdair/bom/BomID.hpp File Reference

```
#include <iosfwd>
#include <string>
```

Namespaces

- namespace stdair

Classes

- struct stdair::BomID< BOM >

  Class wrapper of bom ID (e.g. pointer to object).

9.103 stdair/bom/BomIDTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
#include <stdair/bom/BomID.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef BomID< BookingClass > stdair::BookingClassID_T
- typedef std::list< BookingClassID_T > stdair::BookingClassIDList_T

9.104 stdair/bom/BomINIImport.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/bom/BomINIImport.hpp>
#include <stdair/bom/ConfigHolderStruct.hpp>
#include <stdair/service/Logger.hpp>
```

Namespaces

- namespace bpt
- namespace stdair

Typedefs

- typedef char bpt::ptree

9.105 stdair/bom/BomINIImport.hpp File Reference

```
#include <string>
#include <stdair/stdair_file.hpp>
```

Namespaces

- namespace stdair

Classes

- class stdair::BomINIImport

  Utility class to import StdAir objects in a INI format.

9.106 stdair/bom/BomJSONExport.cpp File Reference

```
#include <cassert>
#include <ostream>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/Bucket.hpp>
#include <stdair/bom/EventStruct.hpp>
#include <stdair/bom/EventTypes.hpp>
#include <stdair/bom/BookingRequestStruct.hpp>
#include <stdair/bom/BreakPointStruct.hpp>
#include <stdair/bom/BomJSONExport.hpp>
```

Namespaces

- namespace stdair

9.107 stdair/bom/BomJSONExport.hpp File Reference

```
#include <iosfwd>
#include <stdair/bom/TravelSolutionTypes.hpp>
```

Namespaces

- namespace bpt
- namespace stdair

Classes

- class stdair::BomJSONExport

  Utility class to export StdAir objects in a JSON format.

Typedefs

- typedef char bpt::ptree

9.108 stdair/bom/BomJSONImport.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/bom/BomJSONImport.hpp>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/stdair_json.hpp>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/ConfigHolderStruct.hpp>
```

Namespaces

- namespace bpt
- namespace stdair

Typedefs

- typedef char bpt::ptree

9.109 stdair/bom/BomJSONImport.hpp File Reference

```
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/basic/JSonCommand.hpp>
#include <stdair/basic/EventType.hpp>
#include <stdair/bom/BreakPointStruct.hpp>
```

Namespaces

- namespace stdair

Classes

- class stdair::BomJSONImport

  Utility class to import StdAir objects in a JSON format.

9.110 stdair/bom/BomKeyManager.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/tokenizer.hpp>
#include <boost/lexical_cast.hpp>
#include <boost/date_time/gregorian/parsers.hpp>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/InventoryKey.hpp>
#include <stdair/bom/FlightDateKey.hpp>
#include <stdair/bom/SegmentDateKey.hpp>
#include <stdair/bom/LegDateKey.hpp>
#include <stdair/bom/ParsedKey.hpp>
#include <stdair/bom/BomKeyManager.hpp>
#include <stdair/service/Logger.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef boost::tokenizer< boost::char_separator< char > > stdair::Tokeniser_T

9.111 stdair/bom/BomKeyManager.hpp File Reference

```
#include <iosfwd>
#include <stdair/stdair_basic_types.hpp>
```

Namespaces

- namespace stdair

Classes

- class stdair::BomKeyManager

  Utility class to extract key structures from strings.

9.112 stdair/bom/BomManager.hpp File Reference

```
#include <iosfwd>
```

```
#include <string>
#include <list>
#include <map>
#include <boost/static_assert.hpp>
#include <boost/type_traits/is_same.hpp>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/BomHolder.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/AirlineFeature.hpp>
```

Namespaces

- namespace stdair

Classes

- class stdair::BomManager
  Utility class for StdAir-based objects.

9.113 stdair/bom/BomRetriever.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/BomKeyManager.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/AirlineFeature.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/FareFamily.hpp>
```

```
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/BomRetriever.hpp>
#include <stdair/bom/ParsedKey.hpp>
#include <stdair/bom/AirportPair.hpp>
#include <stdair/service/Logger.hpp>
```

Namespaces

- namespace stdair

9.114 stdair/bom/BomRetriever.hpp File Reference

```
#include <iosfwd>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/bom/DatePeriod.hpp>
```

Namespaces

- namespace stdair

Classes

- class stdair::BomRetriever

  Utility class to retrieve StdAir objects.

9.115 stdair/bom/BomRoot.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/InventoryKey.hpp>
#include <stdair/bom/Inventory.hpp>
```

Namespaces

- namespace stdair

9.116 stdair/bom/BomRoot.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/BomRootKey.hpp>
#include <stdair/bom/FRAT5CurveHolderStruct.hpp>
#include <stdair/bom/FFDisutilityCurveHolderStruct.hpp>
```

Namespaces

- namespace boost
- namespace boost::serialization
- namespace stdair

Classes

- class stdair::BomRoot

  Class representing the actual attributes for the Bom root.

9.117 stdair/bom/BomRootKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/BomRootKey.hpp>
```

Namespaces

- namespace stdair

Functions

- template void stdair::BomRootKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void stdair::BomRootKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)

9.118 stdair/bom/BomRootKey.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/bom/KeyAbstract.hpp>
```

Namespaces

- namespace boost
- namespace boost::serialization
- namespace stdair

Classes

- struct stdair::BomRootKey

  Key of the BOM structure root.

9.119 stdair/bom/BookingClass.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/RandomGeneration.hpp>
#include <stdair/bom/BookingClass.hpp>
```

Namespaces

- namespace stdair

9.120 stdair/bom/BookingClass.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/stdair_rm_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/BookingClassKey.hpp>
#include <stdair/bom/BookingClassTypes.hpp>
```

Namespaces

- namespace stdair

Classes

- class stdair::BookingClass

9.121 stdair/bom/BookingClassKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BookingClassKey.hpp>
```

Namespaces

- namespace stdair

9.122 stdair/bom/BookingClassKey.hpp File Reference

```
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

Namespaces

- namespace stdair

Classes

- struct stdair::BookingClassKey

9.123 stdair/bom/BookingClassTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef std::list< BookingClass * > stdair::BookingClassList_T
- typedef std::map< const MapKey_T, BookingClass * > stdair::BookingClassMap_T

9.124 stdair/bom/BookingRequestStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/date_time/gregorian/formatters.hpp>
#include <boost/date_time/posix_time/posix_time.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_Request.hpp>
#include <stdair/bom/BookingRequestStruct.hpp>
```

Namespaces

- namespace stdair

Functions

- void stdair::intDisplay (std::ostream &oStream, const int &iInt)

9.125 stdair/bom/BookingRequestStruct.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/BookingRequestTypes.hpp>
```

Namespaces

- namespace stdair

Classes

- struct stdair::BookingRequestStruct

  Structure holding the elements of a booking request.

9.126 stdair/bom/BookingRequestTypes.hpp File Reference

```
#include <boost/shared_ptr.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef boost::shared_ptr< BookingRequestStruct > stdair::BookingRequestPtr_T
- typedef std::string stdair::DemandGeneratorKey_T

9.127 stdair/bom/BreakPointStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/BreakPointStruct.hpp>
```

Namespaces

- namespace stdair

9.128 stdair/bom/BreakPointStruct.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/BreakPointTypes.hpp>
```

Namespaces

- namespace stdair

Classes

- struct stdair::BreakPointStruct

9.129 stdair/bom/BreakPointTypes.hpp File Reference

```
#include <list>
#include <boost/shared_ptr.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef boost::shared_ptr< BreakPointStruct > stdair::BreakPointPtr_T
- typedef std::list< BreakPointStruct > stdair::BreakPointList_T

9.130 stdair/bom/Bucket.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/Bucket.hpp>
```

Namespaces

- namespace stdair

9.131 stdair/bom/Bucket.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/BucketKey.hpp>
#include <stdair/bom/BucketTypes.hpp>
```

Namespaces

- namespace boost
- namespace boost::serialization
- namespace stdair

Classes

- class stdair::Bucket

  Class representing the actual attributes for an airline booking class.

9.132 stdair/bom/BucketKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/bom/BucketKey.hpp>
```

Namespaces

- namespace stdair

Functions

- template void stdair::BucketKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void stdair::BucketKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)

9.133 stdair/bom/BucketKey.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

Namespaces

- namespace boost
- namespace boost::serialization
- namespace stdair

Classes

- struct stdair::BucketKey

  Key of booking-class.

9.134 stdair/bom/BucketTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef std::list< Bucket * > stdair::BucketList_T
- typedef std::map< const MapKey_T, Bucket * > stdair::BucketMap_T

9.135 stdair/bom/CancellationStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_BookingClass.hpp>
#include <stdair/bom/CancellationStruct.hpp>
#include <stdair/bom/BookingClass.hpp>
```

Namespaces

- namespace stdair

9.136 stdair/bom/CancellationStruct.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <vector>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/BookingClassTypes.hpp>
#include <stdair/bom/TravelSolutionTypes.hpp>
#include <stdair/bom/BomIDTypes.hpp>
```

Namespaces

- namespace stdair

Classes

- struct stdair::CancellationStruct
  Structure holding the elements of a travel solution.

9.137 stdair/bom/CancellationTypes.hpp File Reference

```
#include <boost/shared_ptr.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef boost::shared_ptr< CancellationStruct > stdair::CancellationPtr_T

9.138 stdair/bom/ConfigHolderStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/ForecastingMethod.hpp>
#include <stdair/basic/UnconstrainingMethod.hpp>
#include <stdair/basic/PartnershipTechnique.hpp>
#include <stdair/basic/PreOptimisationMethod.hpp>
#include <stdair/basic/OptimisationMethod.hpp>
#include <stdair/bom/AirlineFeature.hpp>
#include <stdair/bom/ConfigHolderStruct.hpp>
#include <stdair/bom/BomRetriever.hpp>
#include <stdair/service/Logger.hpp>
```

Namespaces

- namespace stdair

9.139 stdair/bom/ConfigHolderStruct.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <boost/static_assert.hpp>
#include <boost/type_traits/is_same.hpp>
#include <stdair/stdair_file.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/ConfigHolderTypes.hpp>
```

Namespaces

- namespace bpt
- namespace stdair

Classes

- struct stdair::ConfigHolderStruct

Typedefs

- typedef char bpt::ptree

9.140 stdair/bom/ConfigHolderTypes.hpp File Reference

```
#include <list>
#include <boost/shared_ptr.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef boost::shared_ptr< ConfigHolderStruct > stdair::ConfigHolderPtr_T

9.141 stdair/bom/DatePeriod.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_Period_BOM.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/DatePeriod.hpp>
```

Namespaces

- namespace stdair

9.142 stdair/bom/DatePeriod.hpp File Reference

```
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/DatePeriodKey.hpp>
#include <stdair/bom/DatePeriodTypes.hpp>
```

Namespaces

- namespace stdair

Classes

- class stdair::DatePeriod

  Class representing the actual attributes for a fare date-period.

9.143 stdair/bom/DatePeriodKey.cpp File Reference

```
#include <ostream>
#include <sstream>
#include <boost/date_time/gregorian/formatters.hpp>
#include <stdair/basic/BasConst_Period_BOM.hpp>
#include <stdair/bom/DatePeriodKey.hpp>
```

Namespaces

- namespace stdair

9.144 stdair/bom/DatePeriodKey.hpp File Reference

```
#include <stdair/bom/KeyAbstract.hpp>
#include <stdair/stdair_date_time_types.hpp>
```

Namespaces

- namespace stdair

Classes

- struct stdair::DatePeriodKey

  Key of date-period.

9.145 stdair/bom/DatePeriodTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef std::list< DatePeriod * > stdair::DatePeriodList_T
- typedef std::map< const MapKey_T, DatePeriod * > stdair::DatePeriodMap_T
- typedef std::pair< MapKey_T, DatePeriod * > stdair::DatePeriodWithKey_T
- typedef std::list< DatePeriodWithKey_T > stdair::DatePeriodDetailedList_T

9.146 stdair/bom/DoWStruct.cpp File Reference

```
#include <sstream>
#include <cassert>
#include <stdair/basic/BasConst_Period_BOM.hpp>
#include <stdair/bom/DoWStruct.hpp>
```

Namespaces

- namespace stdair

9.147 stdair/bom/DoWStruct.hpp File Reference

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

Namespaces

- namespace stdair

Classes

- struct stdair::DoWStruct

9.148 stdair/bom/EventStruct.cpp File Reference

```
#include <cassert>
#include <boost/shared_ptr.hpp>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/BasConst_Event.hpp>
#include <stdair/bom/BookingRequestStruct.hpp>
#include <stdair/bom/OptimisationNotificationStruct.hpp>
#include <stdair/bom/SnapshotStruct.hpp>
#include <stdair/bom/CancellationStruct.hpp>
#include <stdair/bom/RMEventStruct.hpp>
#include <stdair/bom/BreakPointStruct.hpp>
#include <stdair/bom/EventStruct.hpp>
```

Namespaces

- namespace stdair

9.149 stdair/bom/EventStruct.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/stdair_event_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/basic/EventType.hpp>
#include <stdair/bom/EventTypes.hpp>
#include <stdair/bom/BookingRequestTypes.hpp>
#include <stdair/bom/OptimisationNotificationTypes.hpp>
#include <stdair/bom/SnapshotTypes.hpp>
#include <stdair/bom/CancellationTypes.hpp>
#include <stdair/bom/RMEventTypes.hpp>
#include <stdair/bom/BreakPointTypes.hpp>
```

Namespaces

- namespace stdair

Classes

- struct stdair::EventStruct

9.150 stdair/bom/EventTypes.hpp File Reference

```
#include <map>
#include <boost/shared_ptr.hpp>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/stdair_event_types.hpp>
#include <stdair/basic/ProgressStatus.hpp>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef std::pair< const LongDuration_T, EventStruct > stdair::EventListElement_T

- typedef std::map< const LongDuration_T, EventStruct > stdair::EventList_T

9.151 stdair/bom/FareFamily.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/FareFamily.hpp>
```

Namespaces

- namespace stdair

9.152 stdair/bom/FareFamily.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_rm_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/FareFamilyKey.hpp>
#include <stdair/bom/FareFamilyTypes.hpp>
```

Namespaces

- namespace boost
- namespace boost::serialization
- namespace stdair

Classes

- class stdair::FareFamily

  Class representing the actual attributes for a family fare.

9.153 stdair/bom/FareFamilyKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
```

```
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/FareFamilyKey.hpp>
```

Namespaces

- namespace stdair

Functions

- template void stdair::FareFamilyKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void stdair::FareFamilyKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)

9.154 stdair/bom/FareFamilyKey.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

Namespaces

- namespace boost
- namespace boost::serialization
- namespace stdair

Classes

- struct stdair::FareFamilyKey

  Key of a given fare family, made of a fare family code.

9.155 stdair/bom/FareFamilyTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef std::list< FareFamily * > stdair::FareFamilyList_T
- typedef std::map< const MapKey_T, FareFamily * > stdair::FareFamilyMap_T

9.156 stdair/bom/FareFeatures.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_DefaultObject.hpp>
#include <stdair/basic/BasConst_Request.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/FareFeatures.hpp>
```

Namespaces

- namespace stdair

9.157 stdair/bom/FareFeatures.hpp File Reference

```
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/FareFeaturesKey.hpp>
#include <stdair/bom/FareFeaturesTypes.hpp>
```

Namespaces

- namespace stdair

Classes

- class stdair::FareFeatures

  Class representing the actual attributes for a fare date-period.

9.158 stdair/bom/FareFeaturesKey.cpp File Reference

```
#include <ostream>
#include <sstream>
#include <stdair/basic/BasConst_DefaultObject.hpp>
#include <stdair/basic/BasConst_Request.hpp>
#include <stdair/bom/FareFeaturesKey.hpp>
```

Namespaces

- namespace stdair

9.159 stdair/bom/FareFeaturesKey.hpp File Reference

```
#include <stdair/bom/KeyAbstract.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/stdair_inventory_types.hpp>
```

Namespaces

- namespace stdair

Classes

- struct stdair::FareFeaturesKey

  Key of date-period.

9.160 stdair/bom/FareFeaturesTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef std::list< FareFeatures * > stdair::FareFeaturesList_T
- typedef std::map< const MapKey_T, FareFeatures * > stdair::FareFeaturesMap_T
- typedef std::pair< MapKey_T, FareFeatures * > stdair::FareFeaturesWithKey_T
- typedef std::list< FareFeaturesWithKey_T > stdair::FareFeaturesDetailedList_T

9.161 stdair/bom/FareOptionStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_BookingClass.hpp>
#include <stdair/bom/FareOptionStruct.hpp>
```

Namespaces

- namespace stdair

9.162 stdair/bom/FareOptionStruct.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/BookingClassTypes.hpp>
```

Namespaces

- namespace stdair

Classes

- struct stdair::FareOptionStruct

  Structure holding the elements of a fare option.

9.163 stdair/bom/FareOptionTypes.hpp File Reference

```
#include <list>
#include <map>
#include <stdair/stdair_types.hpp>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef std::list< FareOptionStruct > stdair::FareOptionList_T

9.164 stdair/bom/FFDisutilityCurveHolderStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/FFDisutilityCurveHolderStruct.hpp>
```

Namespaces

- namespace stdair

9.165 stdair/bom/FFDisutilityCurveHolderStruct.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_rm_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
```

Namespaces

- namespace stdair

Classes

- struct stdair::FFDisutilityCurveHolderStruct

Typedefs

- typedef std::map< const std::string, FFDisutilityCurve_T > stdair::FFDisutilityCurveHolder_T

9.166 stdair/bom/FlightDate.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/SegmentDate.hpp>
```

Namespaces

- namespace stdair

9.167 stdair/bom/FlightDate.hpp File Reference

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

```
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/FlightDateKey.hpp>
#include <stdair/bom/FlightDateTypes.hpp>
```

Namespaces

- namespace boost
- namespace boost::serialization
- namespace stdair

Classes

- class stdair::FlightDate

  Class representing the actual attributes for an airline flight-date.

9.168 stdair/bom/FlightDateKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/date_time/gregorian/formatters.hpp>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/FlightDateKey.hpp>
```

Namespaces

- namespace stdair

Functions

- template void stdair::FlightDateKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void stdair::FlightDateKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)

9.169 stdair/bom/FlightDateKey.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_basic_types.hpp>
```

```
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

Namespaces

- namespace boost
- namespace boost::serialization
- namespace stdair

Classes

- struct stdair::FlightDateKey

  Key of a given flight-date, made of a flight number and a departure date.

9.170 stdair/bom/FlightDateTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef std::list< FlightDate * > stdair::FlightDateList_T
- typedef std::map< const MapKey_T, FlightDate * > stdair::FlightDateMap_T

9.171 stdair/bom/FlightPeriod.cpp File Reference

```
#include <cassert>
#include <stdair/bom/FlightPeriod.hpp>
```

Namespaces

- namespace stdair

9.172 stdair/bom/FlightPeriod.hpp File Reference

```
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/FlightPeriodKey.hpp>
#include <stdair/bom/FlightPeriodTypes.hpp>
```

Namespaces

- namespace stdair

Classes

- class stdair::FlightPeriod

9.173 stdair/bom/FlightPeriodKey.cpp File Reference

```
#include <stdair/bom/FlightPeriodKey.hpp>
```

Namespaces

- namespace stdair

9.174 stdair/bom/FlightPeriodKey.hpp File Reference

```
#include <stdair/bom/KeyAbstract.hpp>
#include <stdair/bom/PeriodStruct.hpp>
```

Namespaces

- namespace stdair

Classes

- struct stdair::FlightPeriodKey

9.175 stdair/bom/FlightPeriodTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef std::list< FlightPeriod * > stdair::FlightPeriodList_T
- typedef std::map< const MapKey_T, FlightPeriod * > stdair::FlightPeriodMap_T

9.176 stdair/bom/FRAT5CurveHolderStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/FRAT5CurveHolderStruct.hpp>
```

Namespaces

- namespace stdair

9.177 stdair/bom/FRAT5CurveHolderStruct.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_rm_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
```

Namespaces

- namespace stdair

Classes

- struct stdair::FRAT5CurveHolderStruct

Typedefs

- typedef std::map< const std::string, FRAT5Curve_T > stdair::FRAT5CurveHolder_T

9.178 stdair/bom/Inventory.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/FlightDate.hpp>
```

Namespaces

- namespace stdair

9.179 stdair/bom/Inventory.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/AirlineFeature.hpp>
#include <stdair/bom/InventoryKey.hpp>
#include <stdair/bom/InventoryTypes.hpp>
```

Namespaces

- namespace boost
- namespace boost::serialization
- namespace stdair

Classes

- class stdair::Inventory

  Class representing the actual attributes for an airline inventory.

9.180 stdair/bom/InventoryKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/InventoryKey.hpp>
```

Namespaces

- namespace stdair

Functions

- template void stdair::InventoryKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void stdair::InventoryKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)

9.181 stdair/bom/InventoryKey.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

Namespaces

- namespace boost
- namespace boost::serialization
- namespace stdair

Classes

- struct stdair::InventoryKey

  Key of a given inventory, made of the airline code.

9.182 stdair/bom/InventoryTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef std::list< Inventory * > stdair::InventoryList_T
- typedef std::map< const MapKey_T, Inventory * > stdair::InventoryMap_T

9.183 stdair/bom/key_types.hpp File Reference

```
#include <string>
#include <list>
```

Namespaces

- namespace stdair

Typedefs

- typedef std::string stdair::MapKey_T
- typedef std::list< std::string > stdair::KeyList_T

9.184 stdair/bom/KeyAbstract.hpp File Reference

```
#include <iosfwd>
#include <string>
```

Namespaces

- namespace stdair

Classes

- struct stdair::KeyAbstract

  Base class for the keys of Business Object Model (BOM) layer.

Functions

- template<class charT, class traits> std::basic_ostream< charT, traits > & operator<< (std::basic_ostream< charT, traits > &ioOut, const stdair::KeyAbstract &iKey)
- template<class charT, class traits> std::basic_istream< charT, traits > & operator>> (std::basic_istream< charT, traits > &ioIn, stdair::KeyAbstract &ioKey)

9.184.1 Function Documentation

9.184.1.1 template<class charT, class traits> std::basic_ostream<charT, traits>& operator<< (std::basic_ostream< charT, traits > & *ioOut*, const stdair::KeyAbstract & *iKey*) [inline]

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (p653) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 74 of file KeyAbstract.hpp.

9.184.1.2 template<class charT, class traits> std::basic_istream<charT, traits>& operator>> (std::basic_istream< charT, traits > & *ioIn*, stdair::KeyAbstract & *ioKey*) [inline]

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (pp655-657) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 102 of file KeyAbstract.hpp.

References stdair::KeyAbstract::fromStream().

9.185 stdair/bom/LegCabin.cpp File Reference

```
#include <cassert>
```

```
#include <sstream>
#include <stdair/basic/BasConst_BookingClass.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/LegCabin.hpp>
```

Namespaces

- namespace stdair

9.186 stdair/bom/LegCabin.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/LegCabinKey.hpp>
#include <stdair/bom/LegCabinTypes.hpp>
#include <stdair/bom/VirtualClassStruct.hpp>
#include <stdair/bom/VirtualClassTypes.hpp>
```

Namespaces

- namespace stdair

Classes

- class stdair::LegCabin

  Class representing the actual attributes for an airline leg-cabin.

9.187 stdair/bom/LegCabinKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
```

```
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/LegCabinKey.hpp>
```

Namespaces

- namespace stdair

9.188 stdair/bom/LegCabinKey.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

Namespaces

- namespace boost
- namespace boost::serialization
- namespace stdair

Classes

- struct stdair::LegCabinKey

  Key of a given leg-cabin, made of a cabin code (only).

9.189 stdair/bom/LegCabinTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef std::list< LegCabin * > stdair::LegCabinList_T
- typedef std::map< const MapKey_T, LegCabin * > stdair::LegCabinMap_T

9.190 stdair/bom/LegDate.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/LegDate.hpp>
```

Namespaces

- namespace stdair

9.191 stdair/bom/LegDate.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/LegDateKey.hpp>
#include <stdair/bom/LegDateTypes.hpp>
```

Namespaces

- namespace stdair

Classes

- class stdair::LegDate

9.192 stdair/bom/LegDateKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/LegDateKey.hpp>
```

Namespaces

- namespace stdair

9.193 stdair/bom/LegDateKey.hpp File Reference

```
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

Namespaces

- namespace stdair

Classes

- struct stdair::LegDateKey

9.194 stdair/bom/LegDateTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef std::list< LegDate * > stdair::LegDateList_T
- typedef std::map< const MapKey_T, LegDate * > stdair::LegDateMap_T

9.195 stdair/bom/NestingNode.cpp File Reference

```
#include <sstream>
#include <cassert>
#include <iomanip>
#include <iostream>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/BookingClassTypes.hpp>
#include <stdair/bom/NestingNode.hpp>
```

Namespaces

- namespace stdair

9.196 stdair/bom/NestingNode.hpp File Reference

```
#include <cmath>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_rm_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/BookingClassTypes.hpp>
#include <stdair/bom/NestingNodeKey.hpp>
```

Namespaces

- namespace boost
- namespace boost::serialization
- namespace stdair

Classes

- class stdair::NestingNode

9.197 stdair/bom/NestingNodeKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/NestingNodeKey.hpp>
```

Namespaces

- namespace stdair

Functions

- template void stdair::NestingNodeKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void stdair::NestingNodeKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)

9.198 stdair/bom/NestingNodeKey.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

Namespaces

- namespace boost
- namespace boost::serialization
- namespace stdair

Classes

- struct stdair::NestingNodeKey

  Key of a given policy, made of a policy code.

9.199 stdair/bom/NestingNodeTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef std::list< NestingNode * > stdair::NestingNodeList_T
- typedef std::map< const MapKey_T, NestingNode * > stdair::NestingNodeMap_T

9.200 stdair/bom/NestingStructureKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/NestingStructureKey.hpp>
```

Namespaces

- namespace stdair

Functions

- template void stdair::NestingStructureKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void stdair::NestingStructureKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)

9.201 stdair/bom/NestingStructureKey.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

Namespaces

- namespace boost
- namespace boost::serialization
- namespace stdair

Classes

- struct stdair::NestingStructureKey

  Key of a given policy, made of a policy code.

9.202 stdair/bom/OnDDate.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/OnDDate.hpp>
```

Namespaces

- namespace stdair

9.203 stdair/bom/OnDDate.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/stdair_rm_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/OnDDateKey.hpp>
#include <stdair/bom/OnDDateTypes.hpp>
```

Namespaces

- namespace boost
- namespace boost::serialization
- namespace stdair

Classes

- class stdair::OnDDate

  Class representing the actual attributes for an airline flight-date.

9.204 stdair/bom/OnDDateKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/date_time/gregorian/formatters.hpp>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/OnDDateKey.hpp>
#include <stdair/bom/BomKeyManager.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/SegmentDate.hpp>
```

```
#include <stdair/bom/BomDisplay.hpp>
```

Namespaces

- namespace stdair

Functions

- template void stdair::OnDDateKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void stdair::OnDDateKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)

9.205 stdair/bom/OnDDateKey.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

Namespaces

- namespace stdair

Classes

- struct stdair::OnDDateKey

  Key of a given O&D-date, made of a list of OnD strings. a OnD string contains the airline code, the flight number, the date and the segment (origin and destination).

9.206 stdair/bom/OnDDateTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/stdair_demand_types.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef std::list< OnDDate * > stdair::OnDDateList_T
- typedef std::map< const MapKey_T, OnDDate * > stdair::OnDDateMap_T
- typedef std::pair< std::string, YieldDemandPair_T > stdair::StringDemandStructPair_T
- typedef std::map< std::string, YieldDemandPair_T > stdair::StringDemandStructMap_T
- typedef std::map< std::string, CabinClassPairList_T > stdair::StringCabinClassPairListMap_T
- typedef std::pair< std::string, CabinClassPairList_T > stdair::StringCabinClassPair_T
- typedef std::map< CabinCode_T, WTPDemandPair_T > stdair::CabinForecastMap_T
- typedef std::pair< CabinCode_T, WTPDemandPair_T > stdair::CabinForecastPair_T

9.207 stdair/bom/OptimisationNotificationStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/bom/OptimisationNotificationStruct.hpp>
```

Namespaces

- namespace stdair

9.208 stdair/bom/OptimisationNotificationStruct.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/OptimisationNotificationTypes.hpp>
```

Namespaces

- namespace stdair

Classes

- struct stdair::OptimisationNotificationStruct

9.209 stdair/bom/OptimisationNotificationTypes.hpp File Reference

```
#include <boost/shared_ptr.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef boost::shared_ptr< OptimisationNotificationStruct > stdair::OptimisationNotificationPtr_T

9.210 stdair/bom/ParsedKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/tokenizer.hpp>
#include <boost/lexical_cast.hpp>
#include <boost/date_time/gregorian/parsers.hpp>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/InventoryKey.hpp>
#include <stdair/bom/FlightDateKey.hpp>
#include <stdair/bom/SegmentDateKey.hpp>
#include <stdair/bom/LegDateKey.hpp>
#include <stdair/bom/ParsedKey.hpp>
#include <stdair/service/Logger.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef boost::tokenizer< boost::char_separator< char > > stdair::Tokeniser_T

Functions

- const boost::char_separator< char > stdair::TokeniserDashSeparator ("-")
- const boost::char_separator< char > stdair::TokeniserTimeSeparator (":")

9.211 stdair/bom/ParsedKey.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

Namespaces

- namespace stdair

Classes

- struct stdair::ParsedKey

9.212 stdair/bom/PeriodStruct.cpp File Reference

```
#include <sstream>
#include <cassert>
#include <stdair/basic/BasConst_Period_BOM.hpp>
#include <stdair/bom/PeriodStruct.hpp>
```

Namespaces

- namespace stdair

9.213 stdair/bom/PeriodStruct.hpp File Reference

```
#include <string>
#include <vector>
#include <stdair/stdair_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/DoWStruct.hpp>
```

Namespaces

- namespace stdair

Classes

- struct stdair::PeriodStruct

9.214 stdair/bom/Policy.cpp File Reference

```
#include <sstream>
#include <cassert>
#include <iomanip>
#include <iostream>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomManager.hpp>
```

```
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/BookingClassTypes.hpp>
#include <stdair/bom/Policy.hpp>
```

Namespaces

- namespace stdair

9.215 stdair/bom/Policy.hpp File Reference

```
#include <cmath>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_rm_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/BookingClassTypes.hpp>
#include <stdair/bom/PolicyKey.hpp>
```

Namespaces

- namespace boost
- namespace boost::serialization
- namespace stdair

Classes

- class stdair::Policy

9.216 stdair/bom/PolicyKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/PolicyKey.hpp>
```

Namespaces

- namespace stdair

Functions

- template void stdair::PolicyKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void stdair::PolicyKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)

9.217 stdair/bom/PolicyKey.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

Namespaces

- namespace boost
- namespace boost::serialization
- namespace stdair

Classes

- struct stdair::PolicyKey

  Key of a given policy, made of a policy code.

9.218 stdair/bom/PolicyTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef std::list< Policy * > stdair::PolicyList_T
- typedef std::map< const MapKey_T, Policy * > stdair::PolicyMap_T

9.219 stdair/bom/PosChannel.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_Request.hpp>
#include <stdair/service/Logger.hpp>
```

```
#include <stdair/bom/PosChannel.hpp>
```

Namespaces

- namespace stdair

9.220 stdair/bom/PosChannel.hpp File Reference

```
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/PosChannelKey.hpp>
#include <stdair/bom/PosChannelTypes.hpp>
```

Namespaces

- namespace stdair

Classes

- class stdair::PosChannel

  Class representing the actual attributes for a fare point of sale.

9.221 stdair/bom/PosChannelKey.cpp File Reference

```
#include <ostream>
#include <sstream>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/basic/BasConst_Request.hpp>
#include <stdair/bom/PosChannelKey.hpp>
```

Namespaces

- namespace stdair

9.222 stdair/bom/PosChannelKey.hpp File Reference

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

Namespaces

- namespace stdair

Classes

- struct stdair::PosChannelKey
  Key of point of sale and channel.

9.223 stdair/bom/PosChannelTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef std::list< PosChannel * > stdair::PosChannelList_T
- typedef std::map< const MapKey_T, PosChannel * > stdair::PosChannelMap_T
- typedef std::pair< MapKey_T, PosChannel * > stdair::PosChannelWithKey_T
- typedef std::list< PosChannelWithKey_T > stdair::PosChannelDetailedList_T

9.224 stdair/bom/RMEventStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/bom/RMEventStruct.hpp>
```

Namespaces

- namespace stdair

9.225 stdair/bom/RMEventStruct.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/RMEventTypes.hpp>
```

Namespaces

- namespace stdair

Classes

- struct stdair::RMEventStruct

9.226 stdair/bom/RMEventTypes.hpp File Reference

```
#include <list>
#include <boost/shared_ptr.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef boost::shared_ptr< RMEventStruct > stdair::RMEventPtr_T
- typedef std::list< RMEventStruct > stdair::RMEventList_T

9.227 stdair/bom/SegmentCabin.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_BookingClass.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_Yield.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/BookingClassTypes.hpp>
#include <stdair/bom/Policy.hpp>
```

Namespaces

- namespace stdair

9.228 stdair/bom/SegmentCabin.hpp File Reference

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

```
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/SegmentCabinKey.hpp>
#include <stdair/bom/SegmentCabinTypes.hpp>
#include <stdair/bom/PolicyTypes.hpp>
```

Namespaces

- namespace boost
- namespace boost::serialization
- namespace stdair

Classes

- class stdair::SegmentCabin

  Class representing the actual attributes for an airline segment-cabin.

9.229 stdair/bom/SegmentCabinKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/SegmentCabinKey.hpp>
```

Namespaces

- namespace stdair

Functions

- template void stdair::SegmentCabinKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void stdair::SegmentCabinKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)

9.230 stdair/bom/SegmentCabinKey.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

Namespaces

- namespace boost
- namespace boost::serialization
- namespace stdair

Classes

- struct stdair::SegmentCabinKey

  Key of a given segment-cabin, made of a cabin code (only).

9.231 stdair/bom/SegmentCabinTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef std::list< SegmentCabin * > stdair::SegmentCabinList_T
- typedef std::map< const MapKey_T, SegmentCabin * > stdair::SegmentCabinMap_T

9.232 stdair/bom/SegmentDate.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_BookingClass.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/SegmentCabin.hpp>
```

Namespaces

- namespace stdair

9.233 stdair/bom/SegmentDate.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/SegmentDateKey.hpp>
#include <stdair/bom/SegmentDateTypes.hpp>
```

Namespaces

- namespace boost
- namespace boost::serialization
- namespace stdair

Classes

- class stdair::SegmentDate

  Class representing the actual attributes for an airline segment-date.

Typedefs

- typedef std::list< std::string > stdair::RoutingLegKeyList_T

9.234 stdair/bom/SegmentDateKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/SegmentDateKey.hpp>
```

Namespaces

- namespace stdair

Functions

- template void stdair::SegmentDateKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)

- template void stdair::SegmentDateKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)

9.235 stdair/bom/SegmentDateKey.hpp File Reference

```
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

Namespaces

- namespace boost
- namespace boost::serialization
- namespace stdair

Classes

- struct stdair::SegmentDateKey

  Key of a given segment-date, made of an origin and a destination airports.

9.236 stdair/bom/SegmentDateTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef std::list< SegmentDate * > stdair::SegmentDateList_T
- typedef std::map< const MapKey_T, SegmentDate * > stdair::SegmentDateMap_T

9.237 stdair/bom/SegmentPeriod.cpp File Reference

```
#include <cassert>
#include <stdair/basic/BasConst_BookingClass.hpp>
#include <stdair/bom/SegmentPeriod.hpp>
```

Namespaces

- namespace stdair

9.238 stdair/bom/SegmentPeriod.hpp File Reference

```
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/SegmentPeriodKey.hpp>
#include <stdair/bom/SegmentPeriodTypes.hpp>
```

Namespaces

- namespace stdair

Classes

- class stdair::SegmentPeriod

9.239 stdair/bom/SegmentPeriodKey.cpp File Reference

```
#include <sstream>
#include <stdair/bom/SegmentPeriodKey.hpp>
```

Namespaces

- namespace stdair

9.240 stdair/bom/SegmentPeriodKey.hpp File Reference

```
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

Namespaces

- namespace stdair

Classes

- struct stdair::SegmentPeriodKey

9.241 stdair/bom/SegmentPeriodTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef std::list< SegmentPeriod * > stdair::SegmentPeriodList_T
- typedef std::map< const MapKey_T, SegmentPeriod * > stdair::SegmentPeriodMap_T
- typedef std::pair< MapKey_T, SegmentPeriod * > stdair::SegmentPeriodWithKey_T
- typedef std::list< SegmentPeriodWithKey_T > stdair::SegmentPeriodDetailedList_T

9.242 stdair/bom/SegmentSnapshotTable.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/multi_array.hpp>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/SegmentSnapshotTable.hpp>
```

Namespaces

- namespace stdair

9.243 stdair/bom/SegmentSnapshotTable.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/SegmentSnapshotTableKey.hpp>
#include <stdair/bom/SegmentSnapshotTableTypes.hpp>
```

Namespaces

- namespace boost
- namespace boost::serialization
- namespace stdair

Classes

- class stdair::SegmentSnapshotTable

  Class representing the actual attributes for an airline segment data tables.

9.244 stdair/bom/SegmentSnapshotTableKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/SegmentSnapshotTableKey.hpp>
```

Namespaces

- namespace stdair

Functions

- template void stdair::SegmentSnapshotTableKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void stdair::SegmentSnapshotTableKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)

9.245 stdair/bom/SegmentSnapshotTableKey.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

Namespaces

- namespace boost
- namespace boost::serialization
- namespace stdair

Classes

- struct stdair::SegmentSnapshotTableKey

  Key of a given guillotine block, made of a guillotine number.

9.246 stdair/bom/SegmentSnapshotTableTypes.hpp File Reference

```
#include <map>
```

```
#include <list>
#include <boost/multi_array.hpp>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef std::list< SegmentSnapshotTable * > stdair::SegmentSnapshotTableList_T
- typedef std::map< const MapKey_T, SegmentSnapshotTable * > stdair::SegmentSnapshotTableMap_T
- typedef std::map< const SegmentCabin *, SegmentDataID_T > stdair::SegmentCabinIndexMap_T
- typedef std::map< const MapKey_T, ClassIndex_T > stdair::ClassIndexMap_T

9.247 stdair/bom/SimpleNestingStructure.cpp File Reference

```
#include <sstream>
#include <cassert>
#include <iomanip>
#include <iostream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/SimpleNestingStructure.hpp>
#include <stdair/bom/NestingNode.hpp>
#include <stdair/bom/NestingNodeTypes.hpp>
#include <stdair/service/Logger.hpp>
```

Namespaces

- namespace stdair

9.248 stdair/bom/SimpleNestingStructure.hpp File Reference

```
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/NestingNodeTypes.hpp>
#include <stdair/bom/SimpleNestingStructureTypes.hpp>
#include <stdair/bom/NestingStructureKey.hpp>
```

Namespaces

- namespace boost
- namespace boost::serialization
- namespace stdair

Classes

- class stdair::SimpleNestingStructure

9.249 stdair/bom/SimpleNestingStructureTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef std::list< SimpleNestingStructure * > stdair::SimpleNestingStructureList_T
- typedef std::map< const MapKey_T, SimpleNestingStructure * > stdair::SimpleNestingStructure-Map_T

9.250 stdair/bom/SnapshotStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/bom/SnapshotStruct.hpp>
```

Namespaces

- namespace stdair

9.251 stdair/bom/SnapshotStruct.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/SnapshotTypes.hpp>
```

Namespaces

- namespace stdair

Classes

- struct stdair::SnapshotStruct

9.252 stdair/bom/SnapshotTypes.hpp File Reference

```
#include <boost/shared_ptr.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef boost::shared_ptr< SnapshotStruct > stdair::SnapshotPtr_T

9.253 stdair/bom/TimePeriod.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/TimePeriod.hpp>
```

Namespaces

- namespace stdair

9.254 stdair/bom/TimePeriod.hpp File Reference

```
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/TimePeriodKey.hpp>
#include <stdair/bom/TimePeriodTypes.hpp>
```

Namespaces

- namespace stdair

Classes

- class stdair::TimePeriod

  Class representing the actual attributes for a fare time-period.

9.255 stdair/bom/TimePeriodKey.cpp File Reference

```
#include <ostream>
#include <sstream>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/TimePeriodKey.hpp>
```

Namespaces

- namespace stdair

9.256 stdair/bom/TimePeriodKey.hpp File Reference

```
#include <stdair/bom/KeyAbstract.hpp>
#include <stdair/stdair_date_time_types.hpp>
```

Namespaces

- namespace stdair

Classes

- struct stdair::TimePeriodKey

  Key of time-period.

9.257 stdair/bom/TimePeriodTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef std::list< TimePeriod * > stdair::TimePeriodList_T
- typedef std::map< const MapKey_T, TimePeriod * > stdair::TimePeriodMap_T
- typedef std::pair< MapKey_T, TimePeriod * > stdair::TimePeriodWithKey_T
- typedef std::list< TimePeriodWithKey_T > stdair::TimePeriodDetailedList_T

9.258 stdair/bom/TravelSolutionStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_BookingClass.hpp>
#include <stdair/bom/TravelSolutionStruct.hpp>
#include <stdair/bom/BomKeyManager.hpp>
#include <stdair/bom/ParsedKey.hpp>
```

Namespaces

- namespace stdair

9.259 stdair/bom/TravelSolutionStruct.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <vector>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/BookingClassTypes.hpp>
#include <stdair/bom/FareOptionStruct.hpp>
#include <stdair/bom/FareOptionTypes.hpp>
#include <stdair/bom/TravelSolutionTypes.hpp>
```

Namespaces

- namespace stdair

Classes

- struct stdair::TravelSolutionStruct

  Structure holding the elements of a travel solution.

9.260 stdair/bom/TravelSolutionTypes.hpp File Reference

```
#include <list>
#include <map>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/key_types.hpp>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/BomIDTypes.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef std::list< TravelSolutionStruct > stdair::TravelSolutionList_T
- typedef KeyList_T stdair::SegmentPath_T
- typedef std::list< SegmentPath_T > stdair::SegmentPathList_T
- typedef std::map< const ClassCode_T, Availability_T > stdair::ClassAvailabilityMap_T
- typedef std::list< ClassAvailabilityMap_T > stdair::ClassAvailabilityMapHolder_T
- typedef std::map< const ClassCode_T, BookingClassID_T > stdair::ClassObjectIDMap_T
- typedef std::list< ClassObjectIDMap_T > stdair::ClassObjectIDMapHolder_T
- typedef std::map< const ClassCode_T, YieldValue_T > stdair::ClassYieldMap_T
- typedef std::list< ClassYieldMap_T > stdair::ClassYieldMapHolder_T
- typedef std::list< BidPriceVector_T > stdair::BidPriceVectorHolder_T
- typedef std::map< const ClassCode_T, const BidPriceVector_T * > stdair::ClassBpvMap_T
- typedef std::list< ClassBpvMap_T > stdair::ClassBpvMapHolder_T

9.261 stdair/bom/VirtualClassStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/bom/VirtualClassStruct.hpp>
#include <stdair/bom/BookingClass.hpp>
```

Namespaces

- namespace stdair

9.262 stdair/bom/VirtualClassStruct.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <vector>
```

```
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/stdair_rm_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/BookingClassTypes.hpp>
```

Namespaces

- namespace stdair

Classes

- struct stdair::VirtualClassStruct

9.263 stdair/bom/VirtualClassTypes.hpp File Reference

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

Namespaces

- namespace stdair

Typedefs

- typedef std::list< VirtualClassStruct > stdair::VirtualClassList_T
- typedef std::map< const YieldLevel_T, VirtualClassStruct > stdair::VirtualClassMap_T

9.264 stdair/bom/YieldFeatures.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_Request.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/YieldFeatures.hpp>
```

Namespaces

- namespace stdair

9.265 stdair/bom/YieldFeatures.hpp File Reference

```
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/YieldFeaturesKey.hpp>
#include <stdair/bom/YieldFeaturesTypes.hpp>
```

Namespaces

- namespace stdair

Classes

- class stdair::YieldFeatures

  Class representing the actual attributes for a yield date-period.

9.266 stdair/bom/YieldFeaturesKey.cpp File Reference

```
#include <ostream>
#include <sstream>
#include <stdair/basic/BasConst_Request.hpp>
#include <stdair/bom/YieldFeaturesKey.hpp>
```

Namespaces

- namespace stdair

9.267 stdair/bom/YieldFeaturesKey.hpp File Reference

```
#include <stdair/bom/KeyAbstract.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/stdair_inventory_types.hpp>
```

Namespaces

- namespace stdair

Classes

- struct stdair::YieldFeaturesKey

  Key of date-period.

9.268 stdair/bom/YieldFeaturesTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef std::list< YieldFeatures * > stdair::YieldFeaturesList_T
- typedef std::map< const MapKey_T, YieldFeatures * > stdair::YieldFeaturesMap_T
- typedef std::pair< MapKey_T, YieldFeatures * > stdair::YieldFeaturesWithKey_T
- typedef std::list< YieldFeaturesWithKey_T > stdair::YieldFeaturesDetailedList_T

9.269 stdair/bom/YieldStore.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/bom/YieldStore.hpp>
```

Namespaces

- namespace stdair

9.270 stdair/bom/YieldStore.hpp File Reference

```
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/YieldStoreKey.hpp>
#include <stdair/bom/YieldStoreTypes.hpp>
```

Namespaces

- namespace stdair

Classes

- class stdair::YieldStore

9.271 stdair/bom/YieldStoreKey.cpp File Reference

```
#include <stdair/bom/YieldStoreKey.hpp>
```

Namespaces

- namespace stdair

9.272 stdair/bom/YieldStoreKey.hpp File Reference

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

Namespaces

- namespace stdair

Classes

- struct stdair::YieldStoreKey

9.273 stdair/bom/YieldStoreTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef std::list< YieldStore * > stdair::YieldStoreList_T
- typedef std::map< const MapKey_T, YieldStore * > stdair::YieldStoreMap_T

9.274 stdair/command/CmdAbstract.cpp File Reference

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

Namespaces

- namespace stdair

9.275 stdair/command/CmdAbstract.hpp File Reference

Namespaces

- namespace stdair

Classes

- class stdair::CmdAbstract

9.276 stdair/command/CmdBomManager.cpp File Reference

9.277 stdair/command/CmdBomManager.hpp File Reference

```
#include <iosfwd>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/SampleType.hpp>
#include <stdair/bom/TravelSolutionTypes.hpp>
#include <stdair/command/CmdAbstract.hpp>
```

Namespaces

- namespace stdair

Classes

- class stdair::CmdBomManager

9.278 stdair/command/CmdBomSerialiser.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/list.hpp>
#include <boost/serialization/map.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/FlightDate.hpp>
```

```
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/Bucket.hpp>
#include <stdair/factory/FacBomManager.hpp>
#include <stdair/factory/FacBom.hpp>
#include <stdair/command/CmdBomSerialiser.hpp>
#include <stdair/service/Logger.hpp>
```

Namespaces

- namespace stdair

Functions

- template<class Archive, class BOM_OBJECT1, class BOM_OBJECT2> void stdair::serialiseHelper (BOM_OBJECT1 &ioObject1, Archive &ioArchive, const unsigned int iFileVersion)
- template void stdair::BomRoot::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void stdair::BomRoot::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void stdair::Inventory::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void stdair::Inventory::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void stdair::FlightDate::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void stdair::FlightDate::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void stdair::SegmentDate::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void stdair::SegmentDate::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void stdair::SegmentCabin::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void stdair::SegmentCabin::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)

9.279 stdair/command/CmdBomSerialiser.hpp File Reference

```
#include <iosfwd>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/TravelSolutionTypes.hpp>
#include <stdair/command/CmdAbstract.hpp>
```

Namespaces

- namespace stdair

Classes

- class stdair::CmdBomSerialiser

9.280 stdair/command/CmdCloneBomManager.cpp File Reference

9.281 stdair/command/CmdCloneBomManager.hpp File Reference

```
#include <iosfwd>
#include <stdair/command/CmdAbstract.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/AirlineFeature.hpp>
#include <stdair/bom/OnDDate.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/Bucket.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/AirportPair.hpp>
#include <stdair/bom/PosChannel.hpp>
#include <stdair/bom/DatePeriod.hpp>
#include <stdair/bom/TimePeriod.hpp>
#include <stdair/bom/FareFeatures.hpp>
#include <stdair/bom/YieldFeatures.hpp>
#include <stdair/bom/AirlineClassList.hpp>
#include <stdair/bom/SegmentPeriod.hpp>
#include <stdair/bom/FlightPeriod.hpp>
```

Namespaces

- namespace stdair

Classes

- class stdair::CmdCloneBomManager

9.282 stdair/command/DBManagerForAirlines.cpp File Reference

```
#include <cassert>
#include <soci.h>
#include <mysql/soci-mysql.h>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/bom/AirlineStruct.hpp>
#include <stdair/dbadaptor/DbaAirline.hpp>
#include <stdair/command/DBManagerForAirlines.hpp>
#include <stdair/service/Logger.hpp>
```

Namespaces

- namespace stdair

9.283 stdair/command/DBManagerForAirlines.hpp File Reference

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

Namespaces

- namespace stdair

Classes

- class stdair::DBManagerForAirlines

9.284 stdair/dbadaptor/DbaAbstract.cpp File Reference

```
#include <stdair/dbadaptor/DbaAbstract.hpp>
```

Namespaces

- namespace stdair

9.285 stdair/dbadaptor/DbaAbstract.hpp File Reference

```
#include <iosfwd>
```

Namespaces

- namespace stdair

Classes

- class stdair::DbaAbstract

Functions

- template<class charT, class traits> std::basic_ostream< charT, traits > & operator<< (std::basic_ostream< charT, traits > &ioOut, const stdair::DbaAbstract &iDba)
- template<class charT, class traits> std::basic_istream< charT, traits > & operator>> (std::basic_istream< charT, traits > &ioIn, stdair::DbaAbstract &ioDba)

9.285.1 Function Documentation

9.285.1.1 template<class charT, class traits> std::basic_ostream<charT, traits>& operator<< (std::basic_ostream< charT, traits > & *ioOut*, const stdair::DbaAbstract & *iDba*) `[inline]`

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (p653) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 41 of file DbaAbstract.hpp.

9.285.1.2 template<class charT, class traits> std::basic_istream<charT, traits>& operator>> (std::basic_istream< charT, traits > & *ioIn*, stdair::DbaAbstract & *ioDba*) `[inline]`

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (pp655-657) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 69 of file DbaAbstract.hpp.

References stdair::DbaAbstract::fromStream().

9.286 stdair/dbadaptor/DbaAirline.cpp File Reference

```
#include <exception>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/AirlineStruct.hpp>
#include <stdair/dbadaptor/DbaAirline.hpp>
#include <stdair/service/Logger.hpp>
```

Namespaces

- namespace soci
- namespace stdair

9.287 stdair/dbadaptor/DbaAirline.hpp File Reference

```
#include <soci/soci.h>
```

Namespaces

- namespace stdair
- namespace soci

Classes

- struct soci::type_conversion< stdair::AirlineStruct >

9.288 stdair/factory/FacAbstract.cpp File Reference

```
#include <cassert>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/factory/FacAbstract.hpp>
```

Namespaces

- namespace stdair

9.289 stdair/factory/FacAbstract.hpp File Reference

Namespaces

- namespace stdair

Classes

- class stdair::FacAbstract

9.290 stdair/factory/FacBom.hpp File Reference

```
#include <cassert>
#include <string>
#include <list>
#include <stdair/factory/FacAbstract.hpp>
#include <stdair/service/FacSupervisor.hpp>
#include <stdair/service/Logger.hpp>
```

Namespaces

- namespace stdair

Classes

- class stdair::FacBom< BOM >

  Base class for Factory layer.

9.291 stdair/factory/FacBomManager.cpp File Reference

```
#include <cassert>
#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/SimpleNestingStructure.hpp>
#include <stdair/bom/NestingNode.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/factory/FacBomManager.hpp>
#include <stdair/service/Logger.hpp>
```

Namespaces

- namespace stdair

9.292 stdair/factory/FacBomManager.hpp File Reference

```
#include <iosfwd>
#include <list>
#include <map>
#include <boost/static_assert.hpp>
#include <boost/type_traits/is_same.hpp>
#include <stdair/bom/BomHolder.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/factory/FacAbstract.hpp>
#include <stdair/factory/FacBom.hpp>
#include <stdair/bom/SegmentDate.hpp>
```

Namespaces

- namespace stdair

Classes

- class stdair::FacBomManager
  Utility class for linking StdAir-based objects.

9.293 stdair/factory/FacCloneBom.hpp File Reference

```
#include <cassert>
#include <string>
#include <list>
#include <stdair/factory/FacAbstract.hpp>
#include <stdair/service/FacSupervisor.hpp>
#include <stdair/service/Logger.hpp>
```

Namespaces

- namespace stdair

Classes

- class stdair::FacCloneBom< BOM >
  Base class for Factory layer.

9.294 stdair/service/DBSessionManager.cpp File Reference

```
#include <cassert>
#include <string>
#include <sstream>
#include <soci.h>
#include <mysql/soci-mysql.h>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/BasDBParams.hpp>
#include <stdair/service/DBSessionManager.hpp>
#include <stdair/service/Logger.hpp>
```

Namespaces

- namespace stdair

9.295 stdair/service/DBSessionManager.hpp File Reference

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

Namespaces

- namespace stdair

Classes

- class stdair::DBSessionManager

9.296 stdair/service/FacServiceAbstract.cpp File Reference

```
#include <cassert>
#include <stdair/service/ServiceAbstract.hpp>
#include <stdair/service/FacServiceAbstract.hpp>
```

Namespaces

- namespace stdair

9.297 stdair/service/FacServiceAbstract.hpp File Reference

```
#include <vector>
```

Namespaces

- namespace stdair

Classes

- class stdair::FacServiceAbstract

9.298 stdair/service/FacSTDAIRServiceContext.cpp File Reference

```
#include <cassert>
#include <stdair/service/FacSupervisor.hpp>
#include <stdair/service/FacSTDAIRServiceContext.hpp>
#include <stdair/service/STDAIR_ServiceContext.hpp>
```

Namespaces

- namespace stdair

9.299 stdair/service/FacSTDAIRServiceContext.hpp File Reference

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

Namespaces

- namespace stdair

Classes

- class stdair::FacSTDAIRServiceContext
  Factory for Bucket.

9.300 stdair/service/FacSupervisor.cpp File Reference

```
#include <cassert>
#include <stdair/factory/FacAbstract.hpp>
#include <stdair/service/FacServiceAbstract.hpp>
#include <stdair/service/FacSupervisor.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/service/DBSessionManager.hpp>
```

Namespaces

- namespace stdair

9.301 stdair/service/FacSupervisor.hpp File Reference

```
#include <iosfwd>
#include <list>
```

Namespaces

- namespace stdair

Classes

- class stdair::FacSupervisor

9.302 stdair/service/Logger.cpp File Reference

```
#include <iostream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/service/Logger.hpp>
```

Namespaces

- namespace stdair

9.303 stdair/service/Logger.hpp File Reference

```
#include <cassert>
#include <sstream>
#include <string>
#include <stdair/stdair_log.hpp>
#include <stdair/basic/BasLogParams.hpp>
```

Namespaces

- namespace stdair

Classes

- class stdair::Logger

Defines

- #define STDAIR_LOG_CORE(iLevel, iToBeLogged)
- #define STDAIR_LOG_CRITICAL(iToBeLogged) STDAIR_LOG_CORE (stdair::LOG::CRITICAL, iToBeLogged)
- #define STDAIR_LOG_ERROR(iToBeLogged) STDAIR_LOG_CORE (stdair::LOG::ERROR, iToBeLogged)
- #define STDAIR_LOG_NOTIFICATION(iToBeLogged) STDAIR_LOG_CORE (stdair::LOG::NOTIFICATION, iToBeLogged)
- #define STDAIR_LOG_WARNING(iToBeLogged) STDAIR_LOG_CORE (stdair::LOG::WARNING, iToBeLogged)
- #define STDAIR_LOG_DEBUG(iToBeLogged) STDAIR_LOG_CORE (stdair::LOG::DEBUG, iToBeLogged)
- #define STDAIR_LOG_VERBOSE(iToBeLogged) STDAIR_LOG_CORE (stdair::LOG::VERBOSE, iToBeLogged)

9.303.1 Define Documentation

9.303.1.1 #define STDAIR_LOG_CORE(iLevel, iToBeLogged)

Value:

```
{ std::ostringstream ostr; ostr << iToBeLogged; \
    stdair::Logger::instance().log (iLevel, __LINE__, __FILE__, ostr.str()); }
```

Definition at line 16 of file Logger.hpp.

9.303.1.2 #define STDAIR_LOG_CRITICAL(iToBeLogged) STDAIR_LOG_CORE (stdair::LOG::CRITICAL, iToBeLogged)

Definition at line 20 of file Logger.hpp.

9.303.1.3 #define STDAIR_LOG_ERROR(iToBeLogged) STDAIR_LOG_CORE (stdair::LOG::ERROR, iToBeLogged)

Definition at line 23 of file Logger.hpp.

Referenced by stdair::ParsedKey::getBoardingTime(), stdair::ParsedKey::getFlightDateKey(), stdair::ParsedKey::getInventoryKey(), stdair::ParsedKey::getLegKey(), stdair::BomManager::getObject(), stdair::ParsedKey::getSegmentKey(), and stdair::ConfigHolderStruct::updateAirlineFeatures().

9.303.1.4 #define STDAIR_LOG_NOTIFICATION(iToBeLogged) STDAIR_LOG_CORE (stdair::LOG::NOTIFICATION, iToBeLogged)

Definition at line 26 of file Logger.hpp.

9.303.1.5 #define STDAIR_LOG_WARNING(iToBeLogged) STDAIR_LOG_CORE (stdair::LOG::WARNING, iToBeLogged)

Definition at line 29 of file Logger.hpp.

9.303.1.6 #define STDAIR_LOG_DEBUG(iToBeLogged) STDAIR_LOG_CORE (stdair::LOG::DEBUG, iToBeLogged)

Definition at line 32 of file Logger.hpp.

Referenced by stdair::FRAT5CurveHolderStruct::addCurve(), stdair::FFDisutilityCurveHolderStruct::addCurve(), stdair::ParsedKey::getBoardingTime(), stdair::FFDisutilityCurveHolderStruct::getFFDisutilityCurve(), stdair::ParsedKey::getFlightDateKey(), stdair::FRAT5CurveHolderStruct::getFRAT5Curve(), stdair::ParsedKey::getInventoryKey(), stdair::ParsedKey::getLegKey(), stdair::ParsedKey::getSegmentKey(), stdair::BomINIImport::importINIConfig(), stdair::TimePeriod::isDepartureTimeValid(), and stdair::BomRetriever::retrieveSegmentDateFromLongKey().

9.303.1.7 #define STDAIR_LOG_VERBOSE(iToBeLogged) STDAIR_LOG_CORE (stdair::LOG::VERBOSE, iToBeLogged)

Definition at line 35 of file Logger.hpp.

9.304 stdair/service/ServiceAbstract.cpp File Reference

```
#include <stdair/service/ServiceAbstract.hpp>
```

Namespaces

- namespace stdair

9.305 stdair/service/ServiceAbstract.hpp File Reference

```
#include <iosfwd>
```

Namespaces

- namespace stdair

Classes

- class stdair::ServiceAbstract

Functions

- template<class charT, class traits> std::basic_ostream< charT, traits > & operator<< (std::basic_ostream< charT, traits > &ioOut, const stdair::ServiceAbstract &iService)
- template<class charT, class traits> std::basic_istream< charT, traits > & operator>> (std::basic_istream< charT, traits > &ioIn, stdair::ServiceAbstract &ioService)

9.305.1 Function Documentation

9.305.1.1 template<class charT, class traits> std::basic_ostream<charT, traits>& operator<< (std::basic_ostream< charT, traits > & *ioOut*, const stdair::ServiceAbstract & *iService*) `[inline]`

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (p653) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 58 of file ServiceAbstract.hpp.

9.305.1.2 template<class charT, class traits> std::basic_istream<charT, traits>& operator>> (std::basic_istream< charT, traits > & *ioIn*, stdair::ServiceAbstract & *ioService*) `[inline]`

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (pp655-657) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 86 of file ServiceAbstract.hpp.

References stdair::ServiceAbstract::fromStream().

9.306 stdair/service/STDAIR_Service.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/stdair_types.hpp>
#include <stdair/stdair_json.hpp>
#include <stdair/basic/BasChronometer.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomRetriever.hpp>
#include <stdair/bom/BomJSONExport.hpp>
#include <stdair/bom/BomJSONImport.hpp>
#include <stdair/bom/BomINIImport.hpp>
#include <stdair/bom/BomDisplay.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/EventStruct.hpp>
```

```
#include <stdair/bom/BookingRequestStruct.hpp>
#include <stdair/bom/DatePeriod.hpp>
#include <stdair/command/CmdBomManager.hpp>
#include <stdair/command/CmdCloneBomManager.hpp>
#include <stdair/service/FacSupervisor.hpp>
#include <stdair/service/FacSTDAIRServiceContext.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/service/DBSessionManager.hpp>
#include <stdair/STDAIR_Service.hpp>
```

Namespaces

- namespace bpt
- namespace stdair

Typedefs

- typedef char bpt::ptree

9.307 stdair/service/STDAIR_ServiceContext.cpp File Reference

9.308 stdair/service/STDAIR_ServiceContext.hpp File Reference

```
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/basic/BasLogParams.hpp>
#include <stdair/basic/BasDBParams.hpp>
#include <stdair/bom/ConfigHolderStruct.hpp>
#include <stdair/basic/ServiceInitialisationType.hpp>
#include <stdair/service/ServiceAbstract.hpp>
```

Namespaces

- namespace stdair

Classes

- class stdair::STDAIR_ServiceContext

  Class holding the context of the Stdair services.

9.309 stdair/stdair_basic_types.hpp File Reference

```
#include <string>
#include <list>
```

Namespaces

- namespace stdair

Typedefs

- typedef std::string stdair::LocationCode_T
- typedef unsigned long int stdair::Distance_T
- typedef LocationCode_T stdair::AirportCode_T
- typedef LocationCode_T stdair::CityCode_T
- typedef std::string stdair::KeyDescription_T
- typedef std::string stdair::AirlineCode_T
- typedef unsigned short stdair::FlightNumber_T
- typedef unsigned short stdair::TableID_T
- typedef std::string stdair::CabinCode_T
- typedef std::string stdair::FamilyCode_T
- typedef std::string stdair::PolicyCode_T
- typedef std::string stdair::NestingStructureCode_T
- typedef std::string stdair::NestingNodeCode_T
- typedef std::string stdair::ClassCode_T
- typedef unsigned long stdair::Identity_T
- typedef std::string stdair::TripType_T
- typedef double stdair::MonetaryValue_T
- typedef double stdair::RealNumber_T
- typedef double stdair::Percentage_T
- typedef double stdair::PriceValue_T
- typedef double stdair::YieldValue_T
- typedef std::string stdair::PriceCurrency_T
- typedef double stdair::Revenue_T
- typedef double stdair::Multiplier_T
- typedef double stdair::NbOfSeats_T
- typedef unsigned int stdair::Count_T
- typedef short stdair::PartySize_T
- typedef double stdair::NbOfRequests_T
- typedef NbOfRequests_T stdair::NbOfBookings_T
- typedef NbOfRequests_T stdair::NbOfCancellations_T
- typedef unsigned short stdair::NbOfTravelSolutions_T
- typedef std::string stdair::ClassList_String_T
- typedef unsigned short stdair::NbOfSegments_T
- typedef unsigned short stdair::NbOfAirlines_T
- typedef double stdair::Availability_T
- typedef double stdair::Fare_T
- typedef bool stdair::Flag_T
- typedef unsigned int stdair::UnsignedIndex_T

- typedef unsigned int stdair::NbOfClasses_T
- typedef unsigned int stdair::NbOfFareFamilies_T
- typedef std::string stdair::Filename_T
- typedef std::string stdair::FileAddress_T
- typedef float stdair::ProgressPercentage_T

9.310 stdair/stdair_date_time_types.hpp File Reference

```
#include <string>
#include <boost/date_time/gregorian/gregorian.hpp>
#include <boost/date_time/posix_time/posix_time.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef boost::posix_time::time_duration stdair::Duration_T
- typedef boost::gregorian::date stdair::Date_T
- typedef boost::posix_time::time_duration stdair::Time_T
- typedef boost::posix_time::ptime stdair::DateTime_T
- typedef boost::gregorian::date_period stdair::DatePeriod_T
- typedef std::string stdair::DOW_String_T
- typedef boost::gregorian::date_duration stdair::DateOffset_T
- typedef int stdair::DayDuration_T
- typedef bool stdair::SaturdayStay_T
- typedef long int stdair::IntDuration_T
- typedef long long int stdair::LongDuration_T
- typedef float stdair::FloatDuration_T

9.311 stdair/stdair_db.hpp File Reference

```
#include <string>
```

Namespaces

- namespace soci
- namespace stdair

Typedefs

- typedef soci::session stdair::DBSession_T
- typedef soci::statement stdair::DBRequestStatement_T
- typedef std::string stdair::DBConnectionName_T

9.312 stdair/stdair_demand_types.hpp File Reference

```
#include <string>
#include <vector>
#include <map>
#include <boost/random/linear_congruential.hpp>
#include <boost/random/uniform_real.hpp>
#include <boost/random/variate_generator.hpp>
#include <boost/date_time/gregorian/gregorian.hpp>
#include <boost/date_time/posix_time/posix_time.hpp>
#include <boost/tuple/tuple.hpp>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/stdair_inventory_types.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef bool stdair::ChangeFees_T
- typedef bool stdair::NonRefundable_T
- typedef bool stdair::SaturdayStay_T
- typedef double stdair::SaturdayStayRatio_T
- typedef double stdair::ChangeFeesRatio_T
- typedef double stdair::NonRefundableRatio_T
- typedef double stdair::Disutility_T
- typedef std::string stdair::PassengerType_T
- typedef std::string stdair::DistributionPatternId_T
- typedef std::string stdair::CancellationRateCurveId_T
- typedef std::string stdair::AirlinePreferenceId_T
- typedef std::pair< Percentage_T, Percentage_T > stdair::CancellationNoShowRatePair_T
- typedef std::string stdair::CharacteristicsPatternId_T
- typedef std::string stdair::CharacteristicsIndex_T
- typedef double stdair::WTP_T
- typedef boost::tuples::tuple< double, WTP_T > stdair::CharacteristicsWTP_tuple_T
- typedef std::pair< WTP_T, MeanStdDevPair_T > stdair::WTPDemandPair_T
- typedef NbOfRequests_T stdair::NbOfCancellations_T
- typedef NbOfRequests_T stdair::NbOfNoShows_T
- typedef double stdair::MatchingIndicator_T
- typedef std::string stdair::DemandStreamKeyStr_T
- typedef std::string stdair::ChannelLabel_T
- typedef std::string stdair::FrequentFlyer_T
- typedef std::string stdair::RequestStatus_T

- typedef std::map< Identity_T, Identity_T > stdair::BookingTSIDMap_T
- typedef std::pair< CabinCode_T, ClassCode_T > stdair::CabinClassPair_T
- typedef std::list< CabinClassPair_T > stdair::CabinClassPairList_T
- typedef double stdair::ProportionFactor_T
- typedef std::list< ProportionFactor_T > stdair::ProportionFactorList_T
- typedef std::string stdair::OnDString_T
- typedef std::list< OnDString_T > stdair::OnDStringList_T

9.313 stdair/stdair_event_types.hpp File Reference

```
#include <string>
```

Namespaces

- namespace stdair

Typedefs

- typedef std::string stdair::EventName_T
- typedef double stdair::NbOfEvents_T
- typedef std::string stdair::EventGeneratorKey_T

9.314 stdair/stdair_exceptions.hpp File Reference

```
#include <string>
```

Namespaces

- namespace stdair

Classes

- class stdair::RootException

  Root of the stdair exceptions.

- class stdair::FileNotFoundException
- class stdair::NonInitialisedLogServiceException
- class stdair::NonInitialisedServiceException
- class stdair::NonInitialisedContainerException
- class stdair::NonInitialisedRelationShipException
- class stdair::MemoryAllocationException
- class stdair::ObjectLinkingException
- class stdair::DocumentNotFoundException
- class stdair::ParserException
- class stdair::SerialisationException
- class stdair::KeyNotFoundException
- class stdair::CodeConversionException
- class stdair::CodeDuplicationException

- class stdair::KeyDuplicationException
- class stdair::ObjectCreationgDuplicationException
- class stdair::ObjectNotFoundException
- class stdair::ParsingFileFailedException
- class stdair::SQLDatabaseException
- class stdair::NonInitialisedDBSessionManagerException
- class stdair::SQLDatabaseConnectionImpossibleException
- class stdair::EventException
- class stdair::SimpleNestingStructException
- class stdair::BookingClassListEmptyInNestingStructException

9.315 stdair/stdair_fare_types.hpp File Reference

Namespaces

- namespace stdair

Typedefs

- typedef double stdair::NbOfFareRules_T

9.316 stdair/stdair_file.hpp File Reference

```
#include <string>
#include <boost/utility.hpp>
#include <stdair/stdair_basic_types.hpp>
```

Namespaces

- namespace stdair

Classes

- class stdair::RootFilePath

  Root of the input and output files.

- class stdair::InputFilePath
- class stdair::ScheduleFilePath
- class stdair::ODFilePath
- class stdair::FRAT5FilePath
- class stdair::FFDisutilityFilePath
- class stdair::ConfigINIFile

9.317 stdair/stdair_inventory_types.hpp File Reference

```
#include <string>
#include <vector>
#include <map>
#include <list>
#include <boost/multi_array.hpp>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef std::string stdair::NetworkID_T
- typedef std::vector< AirlineCode_T > stdair::AirlineCodeList_T
- typedef std::vector< ClassList_String_T > stdair::ClassList_StringList_T
- typedef std::vector< ClassCode_T > stdair::ClassCodeList_T
- typedef unsigned short stdair::SubclassCode_T
- typedef std::string stdair::FlightPathCode_T
- typedef std::map< CabinCode_T, ClassList_String_T > stdair::CabinBookingClassMap_T
- typedef std::string stdair::CurveKey_T
- typedef double stdair::CabinCapacity_T
- typedef double stdair::NbOfFlightDates_T
- typedef double stdair::CommittedSpace_T
- typedef double stdair::UPR_T
- typedef double stdair::BookingLimit_T
- typedef double stdair::AuthorizationLevel_T
- typedef double stdair::CapacityAdjustment_T
- typedef double stdair::BlockSpace_T
- typedef bool stdair::AvailabilityStatus_T
- typedef std::vector< Availability_T > stdair::BucketAvailabilities_T
- typedef double stdair::NbOfYields_T
- typedef double stdair::NbOfInventoryControlRules_T
- typedef bool stdair::CensorshipFlag_T
- typedef short stdair::DTD_T
- typedef short stdair::DCP_T
- typedef std::list< DCP_T > stdair::DCPList_T
- typedef std::map< DTD_T, RealNumber_T > stdair::DTDFratMap_T
- typedef std::map< FloatDuration_T, float > stdair::DTDProbMap_T
- typedef std::vector< CensorshipFlag_T > stdair::CensorshipFlagList_T
- typedef double stdair::BookingRatio_T
- typedef double stdair::Yield_T
- typedef unsigned int stdair::YieldLevel_T

- typedef std::map< YieldLevel_T, MeanStdDevPair_T > stdair::YieldLevelDemandMap_T
- typedef std::pair< Yield_T, MeanStdDevPair_T > stdair::YieldDemandPair_T
- typedef double stdair::BidPrice_T
- typedef std::vector< BidPrice_T > stdair::BidPriceVector_T
- typedef unsigned int stdair::SeatIndex_T
- typedef std::string stdair::ControlMode_T
- typedef double stdair::OverbookingRate_T
- typedef double stdair::BookingLimit_T
- typedef double stdair::ProtectionLevel_T
- typedef std::vector< double > stdair::EmsrValueList_T
- typedef std::vector< double > stdair::BookingLimitVector_T
- typedef std::vector< double > stdair::ProtectionLevelVector_T
- typedef boost::multi_array< double, 2 > stdair::SnapshotBlock_T
- typedef SnapshotBlock_T::index_range stdair::SnapshotBlockRange_T
- typedef SnapshotBlock_T::array_view< 1 >::type stdair::SegmentCabinDTDSnapshotView_T
- typedef SnapshotBlock_T::array_view< 2 >::type stdair::SegmentCabinDTDRangeSnapshotView_T
- typedef SnapshotBlock_T::const_array_view< 1 >::type stdair::ConstSegmentCabinDTDSnapshotView_T
- typedef SnapshotBlock_T::const_array_view< 2 >::type stdair::ConstSegmentCabinDTDRangeSnapshotView_T
- typedef unsigned short stdair::SegmentDataID_T
- typedef unsigned short stdair::LegDataID_T
- typedef unsigned short stdair::ClassIndex_T

9.318 stdair/stdair_json.hpp File Reference

```
#include <string>
```

Namespaces

- namespace stdair

Classes

- class stdair::JSONString
  JSON-formatted string.

9.319 stdair/stdair_log.hpp File Reference

```
#include <string>
```

Namespaces

- namespace stdair
- namespace stdair::LOG

Enumerations

- enum stdair::LOG::EN_LogLevel {
  stdair::LOG::CRITICAL = 0, stdair::LOG::ERROR, stdair::LOG::NOTIFICATION, stdair::LOG::WARNING,
  stdair::LOG::DEBUG, stdair::LOG::VERBOSE, stdair::LOG::LAST_VALUE }

Variables

- static const std::string stdair::LOG::_logLevels [LAST_VALUE]

9.320 stdair/stdair_maths_types.hpp File Reference

```
#include <string>
#include <vector>
#include <map>
#include <boost/random/linear_congruential.hpp>
#include <boost/random/uniform_real.hpp>
#include <boost/random/normal_distribution.hpp>
#include <boost/random/exponential_distribution.hpp>
#include <boost/random/variate_generator.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef unsigned int stdair::ReplicationNumber_T
- typedef unsigned long int stdair::ExponentialSeed_T
- typedef unsigned long int stdair::UniformSeed_T
- typedef unsigned long int stdair::RandomSeed_T
- typedef boost::minstd_rand stdair::BaseGenerator_T
- typedef boost::uniform_real stdair::UniformDistribution_T
- typedef boost::variate_generator< BaseGenerator_T &, UniformDistribution_T > stdair::Uniform-Generator_T
- typedef boost::normal_distribution stdair::NormalDistribution_T
- typedef boost::variate_generator< BaseGenerator_T &, NormalDistribution_T > stdair::Normal-Generator_T
- typedef boost::exponential_distribution stdair::ExponentialDistribution_T
- typedef boost::variate_generator< BaseGenerator_T &, ExponentialDistribution_T > stdair::ExponentialGenerator_T
- typedef double stdair::MeanValue_T
- typedef double stdair::StdDevValue_T
- typedef std::pair< MeanValue_T, StdDevValue_T > stdair::MeanStdDevPair_T
- typedef std::vector< MeanStdDevPair_T > stdair::MeanStdDevPairVector_T
- typedef float stdair::Probability_T

9.321 stdair/stdair_rm_types.hpp File Reference

```
#include <string>
#include <vector>
#include <map>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_inventory_types.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef std::string stdair::ForecasterMode_T
- typedef short stdair::HistoricalDataLimit_T
- typedef std::string stdair::OptimizerMode_T
- typedef NbOfBookings_T stdair::PolicyDemand_T
- typedef std::vector< double > stdair::GeneratedDemandVector_T
- typedef std::vector< GeneratedDemandVector_T > stdair::GeneratedDemandVectorHolder_T
- typedef double stdair::SellupProbability_T
- typedef std::vector< NbOfRequests_T > stdair::UncDemVector_T
- typedef std::vector< NbOfBookings_T > stdair::BookingVector_T
- typedef double stdair::FRAT5_T
- typedef std::map< const DTD_T, FRAT5_T > stdair::FRAT5Curve_T
- typedef std::map< const DTD_T, double > stdair::FFDisutilityCurve_T
- typedef std::map< const DTD_T, double > stdair::SellUpCurve_T
- typedef std::map< const DTD_T, double > stdair::DispatchingCurve_T
- typedef std::map< BookingClass *, SellUpCurve_T > stdair::BookingClassSellUpCurveMap_T
- typedef std::map< BookingClass *, DispatchingCurve_T > stdair::BookingClassDispatchingCurveMap_T
- typedef std::map< const Yield_T, double > stdair::YieldDemandMap_T
- typedef double stdair::Revenue_T
- typedef unsigned int stdair::NbOfSamples_T

9.322 stdair/STDAIR_Service.hpp File Reference

```
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_service_types.hpp>
#include <stdair/stdair_file.hpp>
#include <stdair/basic/BasLogParams.hpp>
#include <stdair/basic/BasDBParams.hpp>
#include <stdair/basic/ServiceInitialisationType.hpp>
#include <stdair/bom/TravelSolutionTypes.hpp>
```

```
#include <stdair/bom/ConfigHolderStruct.hpp>
#include <stdair/service/STDAIR_ServiceContext.hpp>
```

Namespaces

- namespace stdair

Classes

- class stdair::STDAIR_Service

  Interface for the STDAIR Services.

9.323 stdair/stdair_service_types.hpp File Reference

```
#include <boost/shared_ptr.hpp>
```

Namespaces

- namespace stdair

Typedefs

- typedef boost::shared_ptr< STDAIR_Service > stdair::STDAIR_ServicePtr_T

9.324 stdair/stdair_types.hpp File Reference

```
#include <stdair/stdair_exceptions.hpp>
#include <stdair/stdair_log.hpp>
#include <stdair/stdair_db.hpp>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/stdair_fare_types.hpp>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_rm_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/stdair_service_types.hpp>
```

9.325 stdair/ui/cmdline/readline_autocomp.hpp File Reference

```
#include <string>
#include <iosfwd>
```

```
#include <cstdio>
#include <sys/types.h>
#include <sys/file.h>
#include <sys/stat.h>
#include <sys/errno.h>
#include <readline/readline.h>
#include <readline/history.h>
```

Classes

- struct COMMAND

Typedefs

- typedef int(∗) pt2Func (char ∗)

Functions

- char ∗ getwd ()
- char ∗ xmalloc (size_t)
- int com_list (char ∗)
- int com_view (char ∗)
- int com_rename (char ∗)
- int com_stat (char ∗)
- int com_pwd (char ∗)
- int com_delete (char ∗)
- int com_help (char ∗)
- int com_cd (char ∗)
- int com_quit (char ∗)
- char ∗ stripwhite (char ∗iString)
- COMMAND ∗ find_command (char ∗iString)
- char ∗ dupstr (char ∗iString)
- int execute_line (char ∗line)
- char ∗ command_generator (char ∗text, int state)
- char ∗∗ fileman_completion (char ∗text, int start, int end)
- void initialize_readline ()
- void too_dangerous (char ∗caller)
- int valid_argument (char ∗caller, char ∗arg)

Variables

- COMMAND commands [ ]
- int done
- static char syscom [1024]

9.325.1 Typedef Documentation

9.325.1.1 typedef int(∗) pt2Func(char ∗)

Definition at line 35 of file readline_autocomp.hpp.

9.325.2 Function Documentation

9.325.2.1 char∗ getwd ()

readline_autocomp.hpp – A tiny application which demonstrates how to use the GNU Readline library. This application interactively allows users to manipulate files and their modes.

Referenced by com_pwd().

9.325.2.2 char∗ xmalloc (size_t)

Referenced by dupstr().

9.325.2.3 void com_list (char ∗ *arg*)

List the file(s) named in arg.

Definition at line 264 of file readline_autocomp.hpp.

9.325.2.4 int com_view (char ∗)

Definition at line 274 of file readline_autocomp.hpp.

References valid_argument().

9.325.2.5 int com_rename (char ∗)

Definition at line 284 of file readline_autocomp.hpp.

References too_dangerous().

9.325.2.6 int com_stat (char ∗)

Definition at line 289 of file readline_autocomp.hpp.

References valid_argument().

9.325.2.7 int com_pwd (char ∗)

Definition at line 367 of file readline_autocomp.hpp.

References getwd().

Referenced by com_cd().

9.325.2.8 int com_delete (char ∗)

Definition at line 315 of file readline_autocomp.hpp.

References too_dangerous().

9.325.2.9 int com_help (char * *arg*)

Print out help for ARG, or for all of the commands if ARG is not present.

Definition at line 324 of file readline_autocomp.hpp.

References commands, and COMMAND::name.

9.325.2.10 int com_cd (char *)

Definition at line 356 of file readline_autocomp.hpp.

References com_pwd().

9.325.2.11 int com_quit (char *)

Definition at line 381 of file readline_autocomp.hpp.

9.325.2.12 char * stripwhite (char * *string*)

Strip whitespace from the start and end of STRING. Return a pointer into STRING.

Definition at line 152 of file readline_autocomp.hpp.

References command_generator(), and fileman_completion().

9.325.2.13 COMMAND * find_command (char * *name*)

Look up NAME as the name of a command, and return a pointer to that command. Return a NULL pointer if NAME isn't a command name.

Definition at line 136 of file readline_autocomp.hpp.

References commands, and COMMAND::name.

Referenced by execute_line().

9.325.2.14 char* dupstr (char * *iString*)

Duplicate a string

Definition at line 85 of file readline_autocomp.hpp.

References xmalloc().

Referenced by command_generator().

9.325.2.15 int execute_line (char * *line*)

Execute a command line.

Definition at line 94 of file readline_autocomp.hpp.

References find_command(), and COMMAND::func.

9.325.2.16 char * command_generator (char * *text*, int *state*)

Generator function for command completion. STATE lets us know whether to start from scratch; without any state (i.e. STATE == 0), then we start at the top of the list.

Definition at line 222 of file readline_autocomp.hpp.

References commands, and dupstr().

Referenced by stripwhite().

9.325.2.17 char ** fileman_completion (char * *text*, int *start*, int *end*)

Attempt to complete on the contents of TEXT. START and END bound the region of rl_line_buffer that contains the word to complete. TEXT is the word to complete. We can use the entire contents of rl_line_-buffer in case we want to do some simple parsing. Return the array of matches, or NULL if there aren't any.

Definition at line 200 of file readline_autocomp.hpp.

Referenced by stripwhite().

9.325.2.18 void initialize_readline ()

Tell the GNU Readline library how to complete. We want to try to complete on command names if this is the first word in the line, or on filenames if not.

Definition at line 185 of file readline_autocomp.hpp.

9.325.2.19 void too_dangerous (char * *caller*)

Definition at line 387 of file readline_autocomp.hpp.

Referenced by com_delete(), and com_rename().

9.325.2.20 int valid_argument (char * *caller*, char * *arg*)

Definition at line 395 of file readline_autocomp.hpp.

Referenced by com_stat(), and com_view().

9.325.3 Variable Documentation

9.325.3.1 COMMAND commands[ ]

Initial value:

```
{
  { "cd", (*com_cd)(), "Change to directory DIR" },
  { "delete", com_delete, "Delete FILE" },
  { "help", com_help, "Display this text" },
  { "?", com_help, "Synonym for `help'" },
  { "list", com_list, "List files in DIR" },
  { "ls", com_list, "Synonym for `list'" },
  { "pwd", com_pwd, "Print the current working directory" },
  { "quit", com_quit, "Quit using airinv" },
  { "rename", com_rename, "Rename FILE to NEWNAME" },
  { "stat", com_stat, "Print out statistics on FILE" },
  { "view", com_view, "View the contents of FILE" },
  { (char*) NULL, (pt2Func) NULL, (char*) NULL }
}
```

Definition at line 58 of file readline_autocomp.hpp.

Referenced by com_help(), command_generator(), and find_command().

9.325.3.2 int done

When non-zero, this global means the user is done using this program.

Definition at line 80 of file readline_autocomp.hpp.

9.325.3.3 char syscom[1024] [static]

String to pass to system(). This is for the LIST, VIEW and RENAME commands.

Definition at line 259 of file readline_autocomp.hpp.

9.326 stdair/ui/cmdline/SReadline.hpp File Reference

C++ wrapper around libreadline.

```
#include <stdio.h>
#include <readline/readline.h>
#include <readline/history.h>
#include <readline/keymaps.h>
#include <string>
#include <fstream>
#include <vector>
#include <stdexcept>
#include <map>
#include <boost/algorithm/string/trim.hpp>
#include <boost/tokenizer.hpp>
#include <boost/function.hpp>
```

Namespaces

- namespace swift

Classes

- class swift::SKeymap
  The readline keymap wrapper.
- class swift::SReadline
  The readline library wrapper.

Typedefs

- typedef std::vector< std::string > TokensStorage
  Tokens in a single variation of a user command.

- typedef std::vector< TokensStorage > CompletionsStorage

  Set of variations of user commands.

- typedef boost::function< int(int, int)> KeyCallback

  Pressed key callback. Must return 0 if OK, != 0 otherwise.

- typedef std::map< int, KeyCallback > KeysBind

  A set of keys binding.

Functions

- const size_t DefaultHistoryLimit (64)
- bool KeymapWasSetup (false)
- Keymap Earlykeymap (0)
- char * Generator (const char *text, int State)

  Custom completion generator.

- char ** UserCompletion (const char *text, int start, int end)

  The function is called before trying to complete a token.

- int KeyDispatcher (int Count, int Key)

  The function selects the set of bindings and makes the corresponding call.

- int StartupHook (void)

  The readline startup hook. It is required to setup the proper keymap.

- template<typename Container> bool AreTokensEqual (const Container &Pattern, const Container &Input)

  Compares all the Input tokens with starts tokens in the Pattern.

- template<typename ContainerType> void SplitTokens (const std::string &Source, ContainerType &Container)
- char ** UserCompletion (const char *text, int start, int end)
- char * Generator (const char *text, int State)
- int KeyDispatcher (int Count, int Key)
- int StartupHook (void)

Variables

- CompletionsStorage Completions
- TokensStorage Tokens
- std::map< Keymap, KeysBind > Keymaps

9.326.1 Detailed Description

C++ wrapper around libreadline.

Supported: editing, history, custom completers, keymaps. Attention: implementation is not thread safe! It is mainly because the readline library provides pure C interface and has many calls for an "atomic" completion operation

Definition in file SReadline.hpp.

9.326.2 Typedef Documentation

9.326.2.1 typedef std::vector<std::string> TokensStorage [static]

Tokens in a single variation of a user command.

Definition at line 54 of file SReadline.hpp.

9.326.2.2 typedef std::vector<TokensStorage> CompletionsStorage [static]

Set of variations of user commands.

Definition at line 59 of file SReadline.hpp.

9.326.2.3 typedef boost::function<int (int, int)> KeyCallback [static]

Pressed key callback. Must return 0 if OK, != 0 otherwise.

Definition at line 64 of file SReadline.hpp.

9.326.2.4 typedef std::map<int, KeyCallback> KeysBind [static]

A set of keys binding.

Definition at line 69 of file SReadline.hpp.

9.326.3 Function Documentation

9.326.3.1 const size_t @0::DefaultHistoryLimit (64) [static]

Default value for the history length.

9.326.3.2 bool @0::KeymapWasSetup (false) [static]

Has sense if a keymap was setup before the first readline call.

Referenced by swift::SReadline::SetKeymap(), and StartupHook().

9.326.3.3 Keymap @0::Earlykeymap (0) [static]

The keymap which was setup before the first readline call.

Referenced by swift::SReadline::SetKeymap(), and StartupHook().

9.326.3.4 char* @0::Generator (const char * *text*, int *State*) [static]

Custom completion generator.

Parameters:

text Pointer to a token to be completed

State 0 for a first call, non 0 for all consequent calls

Referenced by UserCompletion().

9.326.3.5 char** @0::UserCompletion (const char * *text*, int *start*, int *end*) [static]

The function is called before trying to complete a token.

Parameters:

text A token to be completed

start Index of the beginning of the token in the readline buffer

end Index of the end of the token in the readline buffer

Referenced by swift::SReadline::SReadline().

9.326.3.6 int @0::KeyDispatcher (int *Count*, int *Key*) [static]

The function selects the set of bindings and makes the corresponding call.

Parameters:

Count The parameter is passed by readline

Key The pressed key

Referenced by swift::SKeymap::Bind().

9.326.3.7 int @0::StartupHook (void) [static]

The readline startup hook. It is required to setup the proper keymap.

Referenced by swift::SReadline::SReadline().

9.326.3.8 template<typename Container> bool @0::AreTokensEqual (const Container & *Pattern*, const Container & *Input*) [static]

Compares all the Input tokens with starts tokens in the Pattern.

Parameters:

Pattern pattern tokens

Input user input tokens

Returns:

true if first Input.size() tokens are equal to the pattern tokens

Definition at line 146 of file SReadline.hpp.

Referenced by Generator(), and UserCompletion().

9.326.3.9 template<typename ContainerType> void @0::SplitTokens (const std::string & *Source*, ContainerType & *Container*) [static]

Definition at line 169 of file SReadline.hpp.

Referenced by swift::SReadline::GetLine(), swift::SReadline::RegisterCompletions(), and UserCompletion().

9.326.3.10 char** @0::UserCompletion (const char * *text*, int *start*, int *end*) [static]

Definition at line 189 of file SReadline.hpp.

References AreTokensEqual(), Completions, Generator(), SplitTokens(), and Tokens.

9.326.3.11 char* @0::Generator (const char * *text*, int *State*) [static]

Definition at line 228 of file SReadline.hpp.

References AreTokensEqual(), Completions, and Tokens.

9.326.3.12 int @0::KeyDispatcher (int *Count*, int *Key*) [static]

Definition at line 267 of file SReadline.hpp.

References Keymaps.

9.326.3.13 int @0::StartupHook (void) [static]

Definition at line 284 of file SReadline.hpp.

References Earlykeymap(), and KeymapWasSetup().

9.326.4 Variable Documentation

9.326.4.1 CompletionsStorage Completions [static]

Global storage of custom completions.

Definition at line 79 of file SReadline.hpp.

Referenced by Generator(), swift::SReadline::RegisterCompletions(), and UserCompletion().

9.326.4.2 TokensStorage Tokens [static]

Tokens storage for a single completion session.

Definition at line 84 of file SReadline.hpp.

Referenced by Generator(), and UserCompletion().

9.326.4.3 std::map<Keymap, KeysBind> Keymaps [static]

Global storage for keymaps.

Definition at line 89 of file SReadline.hpp.

Referenced by swift::SKeymap::Bind(), KeyDispatcher(), swift::SKeymap::SKeymap(), swift::SKeymap::Unbind(), and swift::SKeymap::~SKeymap().

9.327 test/stdair/MPBomRoot.cpp File Reference

9.328 test/stdair/MPBomRoot.hpp File Reference

9.329 test/stdair/MPInventory.cpp File Reference

9.330 test/stdair/MPInventory.hpp File Reference

9.331 test/stdair/StandardAirlineITTestSuite.cpp File Reference

9.332 test/stdair/StdairTestLib.hpp File Reference

```
#include <string>
#include <sstream>
```

Namespaces

- namespace stdair_test

Classes

- struct stdair_test::BookingClass
- struct stdair_test::Cabin

10 StdAir Page Documentation

Abstract part of the Business Object Model (BOM)

Author:

Anh Quan Nguyen <quannaus@users.sourceforge.net>

Date:

20/01/2010

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

```
 */
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <ostream>
// StdAir
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/LegDate.hpp>
```

```
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/AirportPair.hpp>
#include <stdair/bom/PosChannel.hpp>
#include <stdair/bom/DatePeriod.hpp>
#include <stdair/bom/TimePeriod.hpp>
#include <stdair/bom/FareFeatures.hpp>
#include <stdair/bom/YieldFeatures.hpp>
#include <stdair/bom/AirlineClassList.hpp>
#include <stdair/bom/Bucket.hpp>
#include <stdair/bom/TravelSolutionTypes.hpp>
#include <stdair/bom/TravelSolutionStruct.hpp>
#include <stdair/bom/BomDisplay.hpp>
#include <stdair/bom/OnDDate.hpp>

namespace stdair {

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

  }
}

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

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

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

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

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

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

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

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

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

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

    // Check if the inventory contains a list of partners

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

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

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

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

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

    // Check if the inventory contains a list of O&D dates

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

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

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

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

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

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

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

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

  const StringDemandStructMap_T& lDemandInfoMap =
    iOnDDate.getDemandInfoMap();

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

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

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

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

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

}

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

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

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

  //
  csvLegCabinDisplay (oStream, iFlightDate);

  //
  csvBucketDisplay (oStream, iFlightDate);
```

```
    //
    csvFareFamilyDisplay (oStream, iFlightDate);

    //
    csvBookingClassDisplay (oStream, iFlightDate);
  }

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

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

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

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

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

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

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

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

    oStream << "****************************************" << std::endl;
    oStream << "SegmentDates:" << std::endl
            << "----------" << std::endl;
    oStream << "Flight, Segment, Date"
            << std::endl;
```

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

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

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

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

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

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

      // Check if the current segment is operated by another segment date.
      const SegmentDate* lOperatingSD_ptr = lSD_ptr->getOperatingSegmentDate ();
      if (lOperatingSD_ptr != NULL) {

        const FlightDate* lOperatingFD_ptr = BomManager::getParentPtr<FlightDate>(*lOperatingSD_ptr);
        const Inventory* lOperatingInv_ptr = BomManager::getParentPtr<Inventory>(*lOperatingFD_ptr);
        oStream << " *** Operated by " << lOperatingInv_ptr->toString()
                << lOperatingFD_ptr->toString() << std::endl;
      }

      oStream << std::endl;
    }
  }

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

    oStream << "****************************************" << std::endl;
    oStream << "LegCabins:" << std::endl
            << "----------" << std::endl;
    oStream << "Flight, Leg, Cabin, "
            << "OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, "
```

```
          << "CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice, "
          << std::endl;

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

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

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

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

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

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

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

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

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

// //////////////////////////////////////////////////////////////////////
void BomDisplay::csvSegmentCabinDisplay (std::ostream& oStream,
```

```
                                        const FlightDate& iFlightDate) {
  // Save the formatting flags for the given STL output stream
  FlagSaver flagSaver (oStream);

}

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

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

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

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

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

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

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

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

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

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

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

```
        oStream << lBoardPoint << "-" << lOffPoint << " "
                << lSegmentDateDate << ", ";

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

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

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

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

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

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

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

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

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

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

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

      // Browse the buckets
      const BucketList_T& lBucketList = BomManager::getList<Bucket> (*lLC_ptr);
      for (BucketList_T::const_iterator itBuck = lBucketList.begin();
           itBuck != lBucketList.end(); ++itBuck) {
        const Bucket* lBucket_ptr = *itBuck;
        assert (lBucket_ptr != NULL);

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

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

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

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

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

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

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

  // Headers
  oStream << "****************************************" << std::endl;
  oStream << "Subclasses:" << std::endl
```

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

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

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

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

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

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

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

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

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

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

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

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

        // Complete the leading string to be displayed
```

```
          std::ostringstream oFFLeadingStr;
          oFFLeadingStr << oSCLeadingStr.str() << lFamilyCode << ", ";

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

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

        // Go on to the next segment-cabin
        continue;
      }
      assert (BomManager::hasList<FareFamily> (*lSC_ptr) == false);

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

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

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

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

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

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

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

// //////////////////////////////////////////////////////////////////
void BomDisplay::
csvDisplay (std::ostream& oStream,
```

```
            const DatePeriodList_T& iDatePeriodList) {

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

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

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

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

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

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

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

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

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

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

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

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

```
    const DatePeriod* lDP_ptr = *itDP;
    assert (lDP_ptr != NULL);

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

// ////////////////////////////////////////////////////////////////////
void BomDisplay::csvDateDisplay (std::ostream& oStream,
                                 const DatePeriod& iDatePeriod) {

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

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

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

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

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

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

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

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

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

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

// ////////////////////////////////////////////////////////////////////
void BomDisplay::csvTimeDisplay (std::ostream& oStream,
```

```
                                const TimePeriod& iTimePeriod) {

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

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

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

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

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

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

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

// ////////////////////////////////////////////////////////////////////
template <typename FEATURE_TYPE>
void BomDisplay::csvFeaturesDisplay (std::ostream& oStream,
                                     const FEATURE_TYPE& iFeatures) {
  // Save the formatting flags for the given STL output stream
  FlagSaver flagSaver (oStream);

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

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

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

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

```
    }
  }

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

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

}

/*!
```

Part of the Business Object Model (BOM) handling (hash-like)keys

Author:

Anh Quan Nguyen <quannaus@users.sourceforge.net>

Date:

20/01/2010

10.2 C++ Class Building Sample StdAir BOM Trees

```
 */
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// StdAir
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/BasConst_DefaultObject.hpp>
#include <stdair/basic/BasConst_Request.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomRetriever.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/AirlineFeature.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/AirportPair.hpp>
#include <stdair/bom/PosChannel.hpp>
#include <stdair/bom/DatePeriod.hpp>
#include <stdair/bom/TimePeriod.hpp>
#include <stdair/bom/FareFeatures.hpp>
#include <stdair/bom/YieldFeatures.hpp>
#include <stdair/bom/AirlineClassList.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/TravelSolutionStruct.hpp>
```

```
#include <stdair/bom/BookingRequestStruct.hpp>
#include <stdair/factory/FacBomManager.hpp>
#include <stdair/factory/FacBom.hpp>
#include <stdair/command/CmdBomManager.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/OnDDate.hpp>
#include <stdair/bom/SegmentPeriod.hpp>
#include <stdair/bom/FlightPeriod.hpp>

namespace stdair {

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

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

    // ////// Basic Bom Tree ///////
    // Build the inventory (flight-dates) and the schedule (flight period)
    // parts.
    buildSampleInventorySchedule (ioBomRoot);

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

    // ////// Partnership Bom Tree ///////
    // Build the inventory (flight-dates) and the schedule (flight period)
    // parts.
    buildPartnershipsSampleInventoryAndRM (ioBomRoot);

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

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

    // ////// Fare Families Bom Tree ///////
    // Build the inventory (flight-dates) and the schedule (flight period)
    // parts with fare families.
    buildSampleInventoryScheduleForFareFamilies (ioBomRoot);

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

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

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

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

    // Add the airline feature object to the BA inventory
    const AirlineFeatureKey lAirlineFeatureBAKey (lAirlineCodeBA);
    AirlineFeature& lAirlineFeatureBA =
      FacBom<AirlineFeature>::instance().create (lAirlineFeatureBAKey);
    FacBomManager::setAirlineFeature (lBAInv, lAirlineFeatureBA);
    FacBomManager::linkWithParent (lBAInv, lAirlineFeatureBA);
```

```
// Link the airline feature object with the top of the BOM tree
FacBomManager::addToListAndMap (ioBomRoot, lAirlineFeatureBA);

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

// Add the airline feature object to the AF inventory
const AirlineFeatureKey lAirlineFeatureAFKey (lAirlineCodeAF);
AirlineFeature& lAirlineFeatureAF =
  FacBom<AirlineFeature>::instance().create (lAirlineFeatureAFKey);
FacBomManager::setAirlineFeature (lAFInv, lAirlineFeatureAF);
FacBomManager::linkWithParent (lAFInv, lAirlineFeatureAF);
// Link the airline feature object with the top of the BOM tree
FacBomManager::addToListAndMap (ioBomRoot, lAirlineFeatureAF);

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

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

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

// Step 0.3: Segment-date level
// Create a first SegmentDate (LHR-SYD) for BA's Inventory
// See http://www.britishairways.com/travel/flightinformation/public/fr_fr?&Carrier=BA&FlightNumber=00
const AirportCode_T lLHR ("LHR");
const AirportCode_T lSYD ("SYD");
const DateOffset_T l1Day (1);
const DateOffset_T l2Days (2);
const Duration_T l2135 (21, 45, 0);
const Duration_T l0610 (6, 10, 0);
const Duration_T l2205 (22, 05, 0);
SegmentDateKey lSegmentDateKey (lLHR, lSYD);

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

// Add the routing leg keys to the LHR-SYD segment.
const std::string lBALHRRoutingLegStr = "BA;9;2011-Jun-10;LHR";
const std::string lBABKKRoutingLegStr = "BA;9;2011-Jun-10;BKK";
lLHRSYDSegment.addLegKey (lBALHRRoutingLegStr);
lLHRSYDSegment.addLegKey (lBABKKRoutingLegStr);

// Fill the SegmentDate content
lLHRSYDSegment.setBoardingDate (lDate);
lLHRSYDSegment.setOffDate (lDate + l2Days);
lLHRSYDSegment.setBoardingTime (l2135);
lLHRSYDSegment.setOffTime (l0610);
lLHRSYDSegment.setElapsedTime (l2135);

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

```
// Create a second SegmentDate (LHR-BKK) for BA's Inventory
// See http://www.britishairways.com/travel/flightinformation/public/fr_fr?&Carrier=BA&FlightNumber=00
const AirportCode_T lBKK ("BKK");
const Duration_T l1540 (15, 40, 0);
const Duration_T l1105 (11, 5, 0);
lSegmentDateKey = SegmentDateKey (lLHR, lBKK);

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

// Add the routing leg key to the LHR-BKK segment.
lLHRBKKSegment.addLegKey (lBALHRRoutingLegStr);

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

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

// Create a third SegmentDate (BKK-SYD) for BA's Inventory
// See http://www.britishairways.com/travel/flightinformation/public/fr_fr?&Carrier=BA&FlightNumber=00
const Duration_T l1705 (17, 5, 0);
const Duration_T l0905 (9, 5, 0);
lSegmentDateKey = SegmentDateKey (lBKK, lSYD);

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

// Add the routing leg key to the BKK-SYD segment.
lBKKSYDSegment.addLegKey (lBABKKRoutingLegStr);

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

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

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

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

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

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

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

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

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

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

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

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

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

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


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

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

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

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

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

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

// Create a LegCabin (Y) for the Leg BKK-SYD
LegCabin& lBKKLegYCabin =
  FacBom<LegCabin>::instance().create (lYLegCabinKey);
FacBomManager::addToListAndMap (lBKKLeg, lBKKLegYCabin);
```

```
FacBomManager::linkWithParent (lBKKLeg, lBKKLegYCabin);
// Display the leg-cabin
// STDAIR_LOG_DEBUG ("LegCabin: " << lBKKLegYCabin.toString());

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

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

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

// Create a FareFamily (1)  for the Segment BKK-SYD, cabin Y on BA's Inv
FareFamily& lBKKSYDSegmentYCabin1Family =
  FacBom<FareFamily>::instance().create (l1FareFamilyKey);
FacBomManager::addToListAndMap (lBKKSYDSegmentYCabin,
                                lBKKSYDSegmentYCabin1Family);
FacBomManager::linkWithParent (lBKKSYDSegmentYCabin,
                               lBKKSYDSegmentYCabin1Family);

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

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

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


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

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

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

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

```
// Create a BookingClass (Q) for the Segment BKK-SYD, cabin Y,
// fare family 1 on BA's Inv
BookingClass& lBKKSYDSegmentYCabin1FamilyQClass =
  FacBom<BookingClass>::instance().create (lQBookingClassKey);
FacBomManager::addToListAndMap (lBKKSYDSegmentYCabin1Family,
                                lBKKSYDSegmentYCabin1FamilyQClass);
FacBomManager::linkWithParent (lBKKSYDSegmentYCabin1Family,
                               lBKKSYDSegmentYCabin1FamilyQClass);

FacBomManager::addToListAndMap (lBKKSYDSegmentYCabin,
                                lBKKSYDSegmentYCabin1FamilyQClass);
FacBomManager::addToListAndMap (lBKKSYDSegment,
                                lBKKSYDSegmentYCabin1FamilyQClass);

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

// Create a BookingClass (Q) for the Segment LHR-SYD, cabin Y,
// fare family 1 on BA's Inv
BookingClass& lLHRSYDSegmentYCabin1FamilyQClass =
  FacBom<BookingClass>::instance().create (lQBookingClassKey);
FacBomManager::addToListAndMap (lLHRSYDSegmentYCabin1Family,
                                lLHRSYDSegmentYCabin1FamilyQClass);
FacBomManager::linkWithParent (lLHRSYDSegmentYCabin1Family,
                               lLHRSYDSegmentYCabin1FamilyQClass);

FacBomManager::addToListAndMap (lLHRSYDSegmentYCabin,
                                lLHRSYDSegmentYCabin1FamilyQClass);
FacBomManager::addToListAndMap (lLHRSYDSegment,
                                lLHRSYDSegmentYCabin1FamilyQClass);

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


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

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

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

// Step 0.3: Segment-date level
// Create a SegmentDate (CDG-SFO) for AF's Inventory
const AirportCode_T lCDG ("CDG");
const AirportCode_T lSFO ("SFO");
const Duration_T l1040 (10, 40, 0);
const Duration_T l1250 (12, 50, 0);
const Duration_T l1110 (11, 10, 0);
lSegmentDateKey = SegmentDateKey (lCDG, lSFO);

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

// Add the routing leg key to the CDG-SFO segment.
```

```
const std::string lAFCDGRoutingLegStr = "AF;84;2011-Mar-20;CDG";
lCDGSFOSegment.addLegKey (lAFCDGRoutingLegStr);

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

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

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

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

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

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

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

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

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

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

// Step 0.7: fare family level
// Create a fareFamily (1) for the Segment CDG-SFO, cabin Y on AF's Inv
FareFamily& lCDGSFOSegmentYCabin1Family =
  FacBom<FareFamily>::instance().create (l1FareFamilyKey);
FacBomManager::addToListAndMap (lCDGSFOSegmentYCabin,
                                lCDGSFOSegmentYCabin1Family);
FacBomManager::linkWithParent (lCDGSFOSegmentYCabin,
                               lCDGSFOSegmentYCabin1Family);

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

// Step 0.8: booking class level Create a BookingClass (Q) for the
```

```
// Segment CDG-SFO, cabin Y, fare family 1 on AF's Inv
BookingClass& lCDGSFOSegmentYCabin1FamilyQClass =
  FacBom<BookingClass>::instance().create (lQBookingClassKey);
FacBomManager::addToListAndMap (lCDGSFOSegmentYCabin1Family,
                                lCDGSFOSegmentYCabin1FamilyQClass);
FacBomManager::linkWithParent (lCDGSFOSegmentYCabin1Family,
                               lCDGSFOSegmentYCabin1FamilyQClass);

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

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

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

lFlightNumber = FlightNumber_T (9);

FlightPeriodKey lBA9FlightPeriodKey (lFlightNumber, lBA9PeriodStruct);

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

// Step 2: segment period level
// Create a segment period for LHR-SYD:

SegmentPeriodKey lLHRSYDSegmentPeriodKey (lLHR, lSYD);

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

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

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

lFlightNumber = FlightNumber_T (84);

FlightPeriodKey lAF84FlightPeriodKey (lFlightNumber, lAF84PeriodStruct);
```

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

  // Step 2: segment period level
  // Create a segment period for CDG-SFO:

  SegmentPeriodKey lCDGSFOSegmentPeriodKey (lCDG, lSFO);

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

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

  /*============================================================================
    ============================================================================
    ============================================================================*/
  // O&D
  // Create an O&D Date (BA;9,2010-Jun-06;LHR,SYD) for BA's Inventory
  OnDString_T  lBALHRSYDOnDStr = "BA;9,2010-Jun-06;LHR,SYD";
  OnDStringList_T lBAOnDStrList;
  lBAOnDStrList.push_back (lBALHRSYDOnDStr);

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

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

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

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

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

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

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

}

// ////////////////////////////////////////////////////////////////////
void CmdBomManager::
```

```
buildSampleInventoryScheduleForFareFamilies (BomRoot& ioBomRoot) {

  // Inventory
  // Step 0.1: Inventory level
  // Get the Inventory SQ (already built by construction)
  const InventoryKey lSQKey ("SQ");
  Inventory& lSQInv = BomManager::getObject<Inventory>(ioBomRoot,
                                                       lSQKey.toString());

  // SQ
  // Step 0.2: Flight-date level
  // Create a FlightDate (SQ747/8-FEB-2010) for SQ's Inventory
  const FlightNumber_T lFlightNumber747 = 747;
  const Date_T lDate (2010, 2, 8);
  const FlightDateKey lFlightDateKey (lFlightNumber747, lDate);

  FlightDate& lSQ747_20100208_FD =
      FacBom<FlightDate>::instance().create (lFlightDateKey);
  FacBomManager::addToListAndMap (lSQInv, lSQ747_20100208_FD);
  FacBomManager::linkWithParent (lSQInv, lSQ747_20100208_FD);

  // Display the flight-date
  // STDAIR_LOG_DEBUG ("FlightDate: " << lSQ747_20100208_FD.toString());

  // Step 0.3: Segment-date level
  // Create a SegmentDate (SIN-BKK) for SQ's Inventory
  const AirportCode_T lSIN ("SIN");
  const AirportCode_T lBKK ("BKK");
  const Duration_T l0635 (6, 35, 0);
  const Duration_T l0800 (8, 0, 0);
  const Duration_T l0225 (2, 25, 0);
  const SegmentDateKey lSegmentDateKey (lSIN, lBKK);

  SegmentDate& lSINBKKSegment =
    FacBom<SegmentDate>::instance().create (lSegmentDateKey);
  FacBomManager::addToListAndMap (lSQ747_20100208_FD, lSINBKKSegment);
  FacBomManager::linkWithParent (lSQ747_20100208_FD, lSINBKKSegment);

  // Add the routing leg key to the SIN-BKK segment.
  const std::string lSQSINRoutingLegStr = "SQ;747;2010-Feb-8;SIN";
  lSINBKKSegment.addLegKey (lSQSINRoutingLegStr);

  // Fill the SegmentDate content
  lSINBKKSegment.setBoardingDate (lDate);
  lSINBKKSegment.setOffDate (lDate);
  lSINBKKSegment.setBoardingTime (l0635);
  lSINBKKSegment.setOffTime (l0800);
  lSINBKKSegment.setElapsedTime (l0225);

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

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

  LegDate& lSINLeg = FacBom<LegDate>::instance().create (lLegDateKey);
  FacBomManager::addToListAndMap (lSQ747_20100208_FD, lSINLeg);
  FacBomManager::linkWithParent (lSQ747_20100208_FD, lSINLeg);

  // Fill the LegDate content
  lSINLeg.setOffPoint (lBKK);
  lSINLeg.setBoardingDate (lDate);
  lSINLeg.setOffDate (lDate);
  lSINLeg.setBoardingTime (l0635);
  lSINLeg.setOffTime (l0800);
  lSINLeg.setElapsedTime (l0225);
```

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

// Step 0.5: segment-cabin level
// Create a SegmentCabin (Y) for the Segment SIN-BKK of SQ's Inventory
const CabinCode_T lY ("Y");
const SegmentCabinKey lYSegmentCabinKey (lY);
SegmentCabin& lSINBKKSegmentYCabin =
  FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
FacBomManager::addToListAndMap (lSINBKKSegment, lSINBKKSegmentYCabin);
FacBomManager::linkWithParent (lSINBKKSegment, lSINBKKSegmentYCabin);
lSINBKKSegmentYCabin.activateFareFamily ();

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

// Step 0.6: leg-cabin level
// Create a LegCabin (Y) for the Leg SIN-BKK on SQ's Inventory
const LegCabinKey lYLegCabinKey (lY);
LegCabin& lSINLegYCabin =
  FacBom<LegCabin>::instance().create (lYLegCabinKey);
FacBomManager::addToListAndMap (lSINLeg, lSINLegYCabin);
FacBomManager::linkWithParent (lSINLeg, lSINLegYCabin);

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

// Step 0.7: fare family level
// Create a FareFamily (1) for the Segment SIN-BKK, cabin Y on SQ's Inv
const FamilyCode_T l1 ("1");
const FareFamilyKey l1FareFamilyKey (l1);
FareFamily& lSINBKKSegmentYCabin1Family =
  FacBom<FareFamily>::instance().create (l1FareFamilyKey);
FacBomManager::addToListAndMap (lSINBKKSegmentYCabin,
                                lSINBKKSegmentYCabin1Family);
FacBomManager::linkWithParent (lSINBKKSegmentYCabin,
                               lSINBKKSegmentYCabin1Family);

// Display the booking class
// STDAIR_LOG_DEBUG ("FareFamily: "
//                   << lSINBKKSegmentYCabin1Family.toString());

// Create a FareFamily (2) for the Segment SIN-BKK, cabin Y on SQ's Inv
const FamilyCode_T l2 ("2");
const FareFamilyKey l2FareFamilyKey (l2);
FareFamily& lSINBKKSegmentYCabin2Family =
    FacBom<FareFamily>::instance().create (l2FareFamilyKey);
FacBomManager::addToListAndMap (lSINBKKSegmentYCabin,
                                lSINBKKSegmentYCabin2Family);
FacBomManager::linkWithParent (lSINBKKSegmentYCabin,
                               lSINBKKSegmentYCabin2Family);

// Display the booking class
// STDAIR_LOG_DEBUG ("FareFamily: "
//                   << lSINBKKSegmentYCabin2Family.toString());

// Step 0.8: booking class level
// Create a BookingClass (Y) for the Segment SIN-BKK, cabin Y,
// fare family 2 on SQ's Inv
const ClassCode_T lClassY ("Y");
const BookingClassKey lYBookingClassKey (lClassY);
BookingClass& lSINBKKSegmentYCabin2FamilyYClass =
  FacBom<BookingClass>::instance().create (lYBookingClassKey);
FacBomManager::addToListAndMap (lSINBKKSegmentYCabin2Family,
                                lSINBKKSegmentYCabin2FamilyYClass);
FacBomManager::linkWithParent (lSINBKKSegmentYCabin2Family,
```

```
                                lSINBKKSegmentYCabin2FamilyYClass);

    FacBomManager::addToListAndMap (lSINBKKSegmentYCabin,
                                    lSINBKKSegmentYCabin2FamilyYClass);
    FacBomManager::addToListAndMap (lSINBKKSegment,
                                    lSINBKKSegmentYCabin2FamilyYClass);
    lSINBKKSegmentYCabin2FamilyYClass.setYield(1200);

    // Display the booking class
    // STDAIR_LOG_DEBUG ("BookingClass: "
    //                   << lSINBKKSegmentYCabin2FamilyYClass.toString());

    // Create a BookingClass (B) for the Segment SIN-BKK, cabin Y,
    // fare family 2 on SQ's Inv
    const ClassCode_T lB ("B");
    const BookingClassKey lBBookingClassKey (lB);
    BookingClass& lSINBKKSegmentYCabin2FamilyBClass =
      FacBom<BookingClass>::instance().create (lBBookingClassKey);
    FacBomManager::addToListAndMap (lSINBKKSegmentYCabin2Family,
                                    lSINBKKSegmentYCabin2FamilyBClass);
    FacBomManager::linkWithParent (lSINBKKSegmentYCabin2Family,
                                   lSINBKKSegmentYCabin2FamilyBClass);

    FacBomManager::addToListAndMap (lSINBKKSegmentYCabin,
                                    lSINBKKSegmentYCabin2FamilyBClass);
    FacBomManager::addToListAndMap (lSINBKKSegment,
                                    lSINBKKSegmentYCabin2FamilyBClass);
    lSINBKKSegmentYCabin2FamilyBClass.setYield(800);

    // Display the booking class
    // STDAIR_LOG_DEBUG ("BookingClass: "
    //                   << lSINBKKSegmentYCabin2FamilyBClass.toString());

    // Create a BookingClass (M) for the Segment SIN-BKK, cabin Y,
    // fare family 1 on SQ's Inv
    const ClassCode_T lM ("M");
    const BookingClassKey lMBookingClassKey (lM);
    BookingClass& lSINBKKSegmentYCabin1FamilyMClass =
      FacBom<BookingClass>::instance().create (lMBookingClassKey);
    FacBomManager::addToListAndMap (lSINBKKSegmentYCabin1Family,
                                    lSINBKKSegmentYCabin1FamilyMClass);
    FacBomManager::linkWithParent (lSINBKKSegmentYCabin1Family,
                                   lSINBKKSegmentYCabin1FamilyMClass);

    FacBomManager::addToListAndMap (lSINBKKSegmentYCabin,
                                    lSINBKKSegmentYCabin1FamilyMClass);
    FacBomManager::addToListAndMap (lSINBKKSegment,
                                    lSINBKKSegmentYCabin1FamilyMClass);
    lSINBKKSegmentYCabin1FamilyMClass.setYield(900);

    // Display the booking class
    // STDAIR_LOG_DEBUG ("BookingClass: "
    //                   << lSINBKKSegmentYCabin1FamilyMClass.toString());

    // Create a BookingClass (Q) for the Segment SIN-BKK, cabin Y,
    // fare family 1 on SQ's Inv
    const ClassCode_T lQ ("Q");
    const BookingClassKey lQBookingClassKey (lQ);
    BookingClass& lSINBKKSegmentYCabin1FamilyQClass =
      FacBom<BookingClass>::instance().create (lQBookingClassKey);
    FacBomManager::addToListAndMap (lSINBKKSegmentYCabin1Family,
                                    lSINBKKSegmentYCabin1FamilyQClass);
    FacBomManager::linkWithParent (lSINBKKSegmentYCabin1Family,
                                   lSINBKKSegmentYCabin1FamilyQClass);

    FacBomManager::addToListAndMap (lSINBKKSegmentYCabin,
                                    lSINBKKSegmentYCabin1FamilyQClass);
```

```
FacBomManager::addToListAndMap (lSINBKKSegment,
                                lSINBKKSegmentYCabin1FamilyQClass);
lSINBKKSegmentYCabin1FamilyQClass.setYield(600);


// Display the booking class
// STDAIR_LOG_DEBUG ("BookingClass: "
//                   << lSINBKKSegmentYCabin1FamilyQClass.toString());

/*============================================================================
  ============================================================================
  ============================================================================*/
// Schedule:
// SQ:
// Step 1: flight period level
// Create a flight period for SQ747:
const DoWStruct lDoWSrtuct ("1111111");
const Date_T lSQ747DateRangeStart (2010, boost::gregorian::Feb, 8);
const Date_T lSQ747DateRangeEnd (2010, boost::gregorian::Feb, 9);
const DatePeriod_T lSQ747DatePeriod (lSQ747DateRangeStart,
                                     lSQ747DateRangeEnd);
const PeriodStruct lSQ747PeriodStruct (lSQ747DatePeriod, lDoWSrtuct);

const FlightPeriodKey lSQ747FlightPeriodKey (lFlightNumber747,
                                       lSQ747PeriodStruct);
FlightPeriod& lSQ747FlightPeriod =
  FacBom<FlightPeriod>::instance().create (lSQ747FlightPeriodKey);
FacBomManager::addToListAndMap (lSQInv, lSQ747FlightPeriod);
FacBomManager::linkWithParent (lSQInv, lSQ747FlightPeriod);

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

const SegmentPeriodKey lSINBKKSegmentPeriodKey (lSIN, lBKK);
SegmentPeriod& lSINBKKSegmentPeriod =
  FacBom<SegmentPeriod>::instance().create (lSINBKKSegmentPeriodKey);
FacBomManager::addToListAndMap (lSQ747FlightPeriod, lSINBKKSegmentPeriod);
FacBomManager::linkWithParent (lSQ747FlightPeriod, lSINBKKSegmentPeriod);

ClassList_String_T lYBMQ  ("YBMQ");
lSINBKKSegmentPeriod.addCabinBookingClassList (lY,lYBMQ);
lSINBKKSegmentPeriod.setBoardingTime (l0635);
lSINBKKSegmentPeriod.setOffTime (l0800);
lSINBKKSegmentPeriod.setElapsedTime (l0225);

/*============================================================================
  ============================================================================
  ============================================================================*/
// O&D
// Create an O&D Date (SQ;747,2011-Feb-14;SIN,BKK) for SQ's Inventory
const OnDString_T lSQSINBKKOnDStr = "SQ;747,2011-Feb-14;SIN,BKK";
OnDStringList_T lSQOnDStrList;
lSQOnDStrList.push_back (lSQSINBKKOnDStr);

const OnDDateKey lSQOnDDateKey (lSQOnDStrList);
OnDDate& lSQ_SINBKK_OnDDate =
  FacBom<OnDDate>::instance().create (lSQOnDDateKey);
// Link to the inventory
FacBomManager::addToListAndMap (lSQInv, lSQ_SINBKK_OnDDate);
FacBomManager::linkWithParent (lSQInv, lSQ_SINBKK_OnDDate);
// Add total forecast info for cabin Y.
const MeanStdDevPair_T lMean120StdDev12 (120.0, 12.0);
const WTP_T lWTP1000 = 1000.0;
const WTPDemandPair_T lWTP1000Mean120StdDev12 (lWTP1000, lMean120StdDev12);
lSQ_SINBKK_OnDDate.setTotalForecast (lY, lWTP1000Mean120StdDev12);

// Add the segment
```

```
    FacBomManager::addToListAndMap (lSQ_SINBKK_OnDDate, lSINBKKSegment);
  }

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

    /* Build the direct accesses between the dummy segment cabins and the dummy
 leg cabins within the dummy flight dates (the dummy fare family
 flight date and the classic dummy flight date).

 As for now (May 2012), that method is called only by RMOL.
 It is a substitute for the code doing it automatically located in AirInv.
 See the AIRINV::InventoryManager::createDirectAccesses command.
     */

    // ////// Dummy Inventory Leg Segment Accesses ///////
    // Retrieve the (sample) segment-cabin.
    SegmentCabin& lDummySegmentCabin =
      BomRetriever::retrieveDummySegmentCabin (ioBomRoot);

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

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

    // ////// Fare Families Dummy Inventory Leg Segment Accesses ///////
    const bool isForFareFamilies = true;
    // Retrieve the (sample) segment-cabin for fare families.
    SegmentCabin& lFFDummySegmentCabin =
      BomRetriever::retrieveDummySegmentCabin (ioBomRoot, isForFareFamilies);

    // Retrieve the (sample) leg-cabin for fare families.
    stdair::LegCabin& lFFDummyLegCabin =
      stdair::BomRetriever::retrieveDummyLegCabin (ioBomRoot,
                                                   isForFareFamilies);

    // Links between the segment-date and the leg-date for fare families.
    FacBomManager::addToListAndMap (lFFDummyLegCabin, lFFDummySegmentCabin);
    FacBomManager::addToListAndMap (lFFDummySegmentCabin, lFFDummyLegCabin);
  }

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

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

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

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

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

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

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

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

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

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

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

}

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

  // Add the airline feature object to the dummy inventory
  const AirlineFeatureKey lAirlineFeatureKey (DEFAULT_AIRLINE_CODE);
  AirlineFeature& lAirlineFeature =
    FacBom<AirlineFeature>::instance().create (lAirlineFeatureKey);
  FacBomManager::setAirlineFeature (lInv, lAirlineFeature);
  FacBomManager::linkWithParent (lInv, lAirlineFeature);
```

```
// Link the airline feature object with the top of the BOM tree
FacBomManager::addToListAndMap (ioBomRoot, lAirlineFeature);

// Flight-date
FlightDateKey lFlightDateKey(DEFAULT_FLIGHT_NUMBER, DEFAULT_DEPARTURE_DATE);
FlightDate& lFlightDate =
  FacBom<FlightDate>::instance().create (lFlightDateKey);
FacBomManager::addToListAndMap (lInv, lFlightDate);
FacBomManager::linkWithParent (lInv, lFlightDate);

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

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

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

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

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

// Add the routing leg key to the dummy segment.
std::ostringstream oStr;
oStr << DEFAULT_AIRLINE_CODE << ";"
     << DEFAULT_FLIGHT_NUMBER << ";"
     << DEFAULT_DEPARTURE_DATE << ";"
     << DEFAULT_ORIGIN;
lSegment.addLegKey (oStr.str());

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

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

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

FareFamily& lSegmentYCabin1Family =
```

```
    FacBom<FareFamily>::instance().create (l1FareFamilyKey);
  FacBomManager::addToListAndMap (lSegmentCabin, lSegmentYCabin1Family);
  FacBomManager::linkWithParent (lSegmentCabin, lSegmentYCabin1Family);

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

  BookingClass& lSegmentYCabin1FamilyQClass =
    FacBom<BookingClass>::instance().create (lQBookingClassKey);
  FacBomManager::addToListAndMap (lSegmentYCabin1Family,
                                  lSegmentYCabin1FamilyQClass);
  FacBomManager::linkWithParent (lSegmentYCabin1Family,
                                 lSegmentYCabin1FamilyQClass);

  FacBomManager::addToListAndMap (lSegmentCabin, lSegmentYCabin1FamilyQClass);
  FacBomManager::addToListAndMap (lSegment, lSegmentYCabin1FamilyQClass);

  /*=============================================================================
    ============================================================================
    ============================================================================*/
  // Schedule:
  // XX:
  // Step 1: flight period level
  // Create a flight period for XX:
  const DoWStruct lDoWSrtuct ("1111111");
  const Date_T lXXDateRangeStart (DEFAULT_DEPARTURE_DATE);
  const Date_T lXXDateRangeEnd (DEFAULT_DEPARTURE_DATE);
  const DatePeriod_T lXXDatePeriod (lXXDateRangeStart, lXXDateRangeEnd);
  const PeriodStruct lXXPeriodStruct (lXXDatePeriod, lDoWSrtuct);

  FlightPeriodKey lXXFlightPeriodKey (DEFAULT_FLIGHT_NUMBER, lXXPeriodStruct);

  FlightPeriod& lXXFlightPeriod =
    FacBom<FlightPeriod>::instance().create (lXXFlightPeriodKey);
  FacBomManager::addToListAndMap (lInv, lXXFlightPeriod);
  FacBomManager::linkWithParent (lInv, lXXFlightPeriod);

  // Step 2: segment period level
  // Create a segment period

  SegmentPeriodKey lXXSegmentPeriodKey (DEFAULT_ORIGIN, DEFAULT_DESTINATION);

  SegmentPeriod& lXXSegmentPeriod =
    FacBom<SegmentPeriod>::instance().create (lXXSegmentPeriodKey);
  FacBomManager::addToListAndMap (lXXFlightPeriod, lXXSegmentPeriod);
  FacBomManager::linkWithParent (lXXFlightPeriod, lXXSegmentPeriod);

  lXXSegmentPeriod.setBoardingTime (Duration_T (14, 0, 0));
  lXXSegmentPeriod.setOffTime (Duration_T (16, 0, 0));
  lXXSegmentPeriod.setElapsedTime (Duration_T (8, 0, 0));
  const CabinCode_T lY ("Y");
  const ClassList_String_T lYQ ("YQ");
  lXXSegmentPeriod.addCabinBookingClassList (lY,lYQ);


}

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

  // Build a dummy inventory, containing a dummy flight-date with a
  // single segment-cabin and a single leg-cabin (for fare families
  // algorithms)

  // Get the default Inventory object (already built in by construction)
```

```
const InventoryKey lInventoryKey (DEFAULT_AIRLINE_CODE);
Inventory& lInv = BomManager::getObject<Inventory>(ioBomRoot,
                                                   lInventoryKey.toString());

// Create a dummy Flight-date
const FlightDateKey lFlightDateKey(DEFAULT_FLIGHT_NUMBER_FF,
                                   DEFAULT_DEPARTURE_DATE);
FlightDate& lFlightDate =
  FacBom<FlightDate>::instance().create (lFlightDateKey);
FacBomManager::addToListAndMap (lInv, lFlightDate);
FacBomManager::linkWithParent (lInv, lFlightDate);

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

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

// Create a dummy Leg-cabin
const LegCabinKey lLegCabinKey (DEFAULT_CABIN_CODE);
LegCabin& lLegCabin = FacBom<LegCabin>::instance().create (lLegCabinKey);
FacBomManager::addToListAndMap (lLeg, lLegCabin);
FacBomManager::linkWithParent (lLeg, lLegCabin);
const CabinCapacity_T lCapacity = DEFAULT_CABIN_CAPACITY;
lLegCabin.setCapacities (lCapacity);
lLegCabin.setAvailabilityPool (lCapacity);

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

// Add the routing leg key to the dummy segment.
std::ostringstream oStr;
oStr << DEFAULT_AIRLINE_CODE << ";"
     << DEFAULT_FLIGHT_NUMBER << ";"
     << DEFAULT_DEPARTURE_DATE << ";"
     << DEFAULT_ORIGIN;
lSegment.addLegKey (oStr.str());

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

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

// Create a dummy FareFamily (FF1)
const FamilyCode_T l1 ("FF1");
const FareFamilyKey l1FareFamilyKey (l1);
```

```
FareFamily& lSegmentYCabin1Family =
  FacBom<FareFamily>::instance().create (l1FareFamilyKey);
// Set the forecasted demand
// TODO change the size (hard code)
MeanStdDevPairVector_T lDemandVector1FareFamily;
const unsigned int size = 16;
for (unsigned int idx = 0; idx < size; ++idx) {
  double i = static_cast<double> (idx);
  MeanStdDevPair_T lMeanStdDevPair (i/4.0, i/20.0);
  lDemandVector1FareFamily.push_back(lMeanStdDevPair);
}
lSegmentYCabin1Family.setMeanStdDev(lDemandVector1FareFamily);
FacBomManager::addToListAndMap (lSegmentCabin, lSegmentYCabin1Family);
FacBomManager::linkWithParent (lSegmentCabin, lSegmentYCabin1Family);

// Create a dummy booking-class
const ClassCode_T lY ("Y");
const BookingClassKey lYBookingClassKey (lY);

BookingClass& lSegmentYCabin1FamilyYClass =
  FacBom<BookingClass>::instance().create (lYBookingClassKey);
Yield_T lYield = 1000;
lSegmentYCabin1FamilyYClass.setYield(lYield);
FacBomManager::addToListAndMap (lSegmentYCabin1Family,
                                lSegmentYCabin1FamilyYClass);
FacBomManager::linkWithParent (lSegmentYCabin1Family,
                               lSegmentYCabin1FamilyYClass);

FacBomManager::addToListAndMap (lSegmentCabin, lSegmentYCabin1FamilyYClass);
FacBomManager::addToListAndMap (lSegment, lSegmentYCabin1FamilyYClass);

// Create a second dummy booking-class
const ClassCode_T lU ("U");
const BookingClassKey lUBookingClassKey (lU);

BookingClass& lSegmentYCabin1FamilyUClass =
  FacBom<BookingClass>::instance().create (lUBookingClassKey);
lYield = 600;
lSegmentYCabin1FamilyUClass.setYield(lYield);
FacBomManager::addToListAndMap (lSegmentYCabin1Family,
                                lSegmentYCabin1FamilyUClass);
FacBomManager::linkWithParent (lSegmentYCabin1Family,
                               lSegmentYCabin1FamilyUClass);

FacBomManager::addToListAndMap (lSegmentCabin, lSegmentYCabin1FamilyUClass);
FacBomManager::addToListAndMap (lSegment, lSegmentYCabin1FamilyUClass);

// Create a second dummy FareFamily (2)
const FamilyCode_T l2 ("FF2");
const FareFamilyKey l2FareFamilyKey (l2);

FareFamily& lSegmentYCabin2Family =
  FacBom<FareFamily>::instance().create (l2FareFamilyKey);
// Set the forecasted demand
// TODO change the size (hard code)
MeanStdDevPairVector_T lDemandVector2FareFamily;
for (unsigned int idx = 0; idx < size; ++idx) {
  double i = static_cast<double> (idx);
  MeanStdDevPair_T lMeanStdDevPair (i/2.0, i/10.0);
  lDemandVector2FareFamily.push_back(lMeanStdDevPair);
}
lSegmentYCabin2Family.setMeanStdDev(lDemandVector2FareFamily);

FacBomManager::addToListAndMap (lSegmentCabin, lSegmentYCabin2Family);
FacBomManager::linkWithParent (lSegmentCabin, lSegmentYCabin2Family);
```

```
// Create a third dummy booking-class
const ClassCode_T lO ("O");
const BookingClassKey lOBookingClassKey (lO);

BookingClass& lSegmentYCabin2FamilyOClass =
  FacBom<BookingClass>::instance().create (lOBookingClassKey);
lYield = 750;
lSegmentYCabin2FamilyOClass.setYield(lYield);
FacBomManager::addToListAndMap (lSegmentYCabin2Family,
                                lSegmentYCabin2FamilyOClass);
FacBomManager::linkWithParent (lSegmentYCabin2Family,
                               lSegmentYCabin2FamilyOClass);

FacBomManager::addToListAndMap (lSegmentCabin, lSegmentYCabin2FamilyOClass);
FacBomManager::addToListAndMap (lSegment, lSegmentYCabin2FamilyOClass);

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

BookingClass& lSegmentYCabin2FamilyQClass =
  FacBom<BookingClass>::instance().create (lQBookingClassKey);
lYield = 400;
lSegmentYCabin2FamilyQClass.setYield(lYield);
FacBomManager::addToListAndMap (lSegmentYCabin2Family,
                                lSegmentYCabin2FamilyQClass);
FacBomManager::linkWithParent (lSegmentYCabin2Family,
                               lSegmentYCabin2FamilyQClass);

FacBomManager::addToListAndMap (lSegmentCabin, lSegmentYCabin2FamilyQClass);
FacBomManager::addToListAndMap (lSegment, lSegmentYCabin2FamilyQClass);


/*============================================================================
  ============================================================================
  ============================================================================*/
// Schedule:
// XX:
// Step 1: flight period level
// Create a flight period for XX:
const DoWStruct lDoWSrtuct ("1111111");
const Date_T lXXDateRangeStart (DEFAULT_DEPARTURE_DATE);
const Date_T lXXDateRangeEnd (DEFAULT_DEPARTURE_DATE);
const DatePeriod_T lXXDatePeriod (lXXDateRangeStart, lXXDateRangeEnd);
const PeriodStruct lXXPeriodStruct (lXXDatePeriod, lDoWSrtuct);

const FlightPeriodKey lXXFlightPeriodKey (DEFAULT_FLIGHT_NUMBER_FF,
                                          lXXPeriodStruct);

FlightPeriod& lXXFlightPeriod =
  FacBom<FlightPeriod>::instance().create (lXXFlightPeriodKey);
FacBomManager::addToListAndMap (lInv, lXXFlightPeriod);
FacBomManager::linkWithParent (lInv, lXXFlightPeriod);

// Step 2: segment period level
// Create a segment period
const SegmentPeriodKey lXXSegmentPeriodKey (DEFAULT_ORIGIN,
                                            DEFAULT_DESTINATION);

SegmentPeriod& lXXSegmentPeriod =
  FacBom<SegmentPeriod>::instance().create (lXXSegmentPeriodKey);
FacBomManager::addToListAndMap (lXXFlightPeriod, lXXSegmentPeriod);
FacBomManager::linkWithParent (lXXFlightPeriod, lXXSegmentPeriod);

lXXSegmentPeriod.setBoardingTime (Duration_T (14, 0, 0));
lXXSegmentPeriod.setOffTime (Duration_T (16, 0, 0));
lXXSegmentPeriod.setElapsedTime (Duration_T (8, 0, 0));
```

```
  const CabinCode_T lYCabin ("Y");
  const ClassList_String_T lYUOQ ("YUOQ");
  lXXSegmentPeriod.addCabinBookingClassList (lYCabin,lYUOQ);

}

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

    // Get the airport-pair primary key SIN-BKK
    // (already built by construction)
    const AirportPairKey lAirportPairKey ("SIN", "BKK");
    AirportPair& lAirportPair =
      BomManager::getObject<AirportPair>(ioBomRoot, lAirportPairKey.toString());

    // Set the fare date-period primary key.
    const Date_T lDateRangeStart (2010, boost::gregorian::Feb, 1);
    const Date_T lDateRangeEnd (2011, boost::gregorian::Feb, 15);
    const DatePeriod_T lDateRange (lDateRangeStart, lDateRangeEnd);
    const DatePeriodKey lDatePeriodKey (lDateRange);

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

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

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

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

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

// Pricing -- Generate the FareRule
const DayDuration_T ONE_MONTH_ADVANCE_PURCHASE = 30;
// Generate the first FareFeatures for the class Q
const FareFeaturesKey lFareFeaturesQKey (TRIP_TYPE_ONE_WAY,
                                         ONE_MONTH_ADVANCE_PURCHASE,
                                         SATURDAY_STAY,
                                         CHANGE_FEES,
                                         NON_REFUNDABLE,
                                         NO_STAY_DURATION);

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

// Generate the second FareFeatures for the class M
const FareFeaturesKey lFareFeaturesMKey (TRIP_TYPE_ONE_WAY,
                                         NO_ADVANCE_PURCHASE,
                                         SATURDAY_STAY,
                                         CHANGE_FEES,
                                         NON_REFUNDABLE,
                                         NO_STAY_DURATION);

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

// Generate the third FareFeatures for the class B
const FareFeaturesKey lFareFeaturesBKey (TRIP_TYPE_ONE_WAY,
                                          ONE_MONTH_ADVANCE_PURCHASE,
                                          SATURDAY_STAY,
                                          NO_CHANGE_FEES,
                                          NO_NON_REFUNDABLE,//Refundable
                                          NO_STAY_DURATION);

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

// Generate the fourth FareFeatures for the class Y
const FareFeaturesKey lFareFeaturesYKey (TRIP_TYPE_ONE_WAY,
                                         NO_ADVANCE_PURCHASE,
                                         SATURDAY_STAY,
                                         NO_CHANGE_FEES,
                                         NO_NON_REFUNDABLE,//Refundable
                                         NO_STAY_DURATION);

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

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

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

// Generate Segment Features and link them to their respective
// fare and yield rules.
AirlineCodeList_T lAirlineCodeList;
lAirlineCodeList.push_back ("SQ");

ClassList_StringList_T lClassYList;
lClassYList.push_back (CLASS_CODE_Y);
const AirlineClassListKey lAirlineClassYListKey (lAirlineCodeList,
                                                 lClassYList);

// Create the AirlineClassList
AirlineClassList& lAirlineClassYList =
    FacBom<AirlineClassList>::instance().create (lAirlineClassYListKey);
// Link the AirlineClassList to the FareFeatures object
FacBomManager::addToListAndMap (lFareFeaturesY, lAirlineClassYList);
FacBomManager::linkWithParent (lFareFeaturesY, lAirlineClassYList);
lAirlineClassYList.setFare (1200);
lAirlineClassYList.setYield (1200);

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

ClassList_StringList_T lClassBList;
lClassBList.push_back ("B");
const AirlineClassListKey lAirlineClassBListKey (lAirlineCodeList,
                                                 lClassBList);
// Create the AirlineClassList
AirlineClassList& lAirlineClassBList =
  FacBom<AirlineClassList>::instance().create (lAirlineClassBListKey);
// Link the AirlineClassList to the FareFeatures object
FacBomManager::addToListAndMap (lFareFeaturesB, lAirlineClassBList);
FacBomManager::linkWithParent (lFareFeaturesB, lAirlineClassBList);
lAirlineClassBList.setFare (800);
lAirlineClassBList.setYield (800);

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

ClassList_StringList_T lClassMList;
lClassMList.push_back ("M");
const AirlineClassListKey lAirlineClassMListKey (lAirlineCodeList,
                                                 lClassMList);

// Create the AirlineClassList
AirlineClassList& lAirlineClassMList =
  FacBom<AirlineClassList>::instance().create (lAirlineClassMListKey);
// Link the AirlineClassList to the FareFeatures object
FacBomManager::addToListAndMap (lFareFeaturesM, lAirlineClassMList);
FacBomManager::linkWithParent (lFareFeaturesM, lAirlineClassMList);
lAirlineClassMList.setFare (900);
lAirlineClassMList.setYield (900);

// Link the AirlineClassList to the YieldFeatures object
FacBomManager::addToListAndMap (lYieldFeatures, lAirlineClassMList);
```

```
  // \todo (gsabatier): the following calls overrides the parent for
  //       lAirlineClassList. Check that it is what is actually wanted.
  FacBomManager::linkWithParent (lYieldFeatures, lAirlineClassMList);

  ClassList_StringList_T lClassQList;
  lClassQList.push_back ("Q");
  const AirlineClassListKey lAirlineClassQListKey (lAirlineCodeList,
                                                   lClassQList);

  // Create the AirlineClassList
  AirlineClassList& lAirlineClassQList =
    FacBom<AirlineClassList>::instance().create (lAirlineClassQListKey);
  // Link the AirlineClassList to the FareFeatures object
  FacBomManager::addToListAndMap (lFareFeaturesQ, lAirlineClassQList);
  FacBomManager::linkWithParent (lFareFeaturesQ, lAirlineClassQList);
  lAirlineClassQList.setFare (600);
  lAirlineClassQList.setYield (600);

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

}

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

  // Clean the list
  ioTravelSolutionList.clear();

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

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

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

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

  // Clean the list
  ioTravelSolutionList.clear();

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

  // Add the segment date key to the travel solution
  TravelSolutionStruct lTS1;
  lTS1.addSegment (lBA9_SegmentDateKey);

  // Fare option number 1
  const ClassCode_T lClassPathQ (CLASS_CODE_Q);
  const Fare_T lFare900 (900);
  const ChangeFees_T lChangeFee (CHANGE_FEES);
  const NonRefundable_T isNonRefundable (NON_REFUNDABLE);
  const SaturdayStay_T lSaturdayStay (SATURDAY_STAY);
  const FareOptionStruct lFareOption1 (lClassPathQ, lFare900, lChangeFee,
                                       isNonRefundable, lSaturdayStay);
```

```
  // Add (a copy of) the fare option
  lTS1.addFareOption (lFareOption1);
  //

  // Map of class availabilities: set the availability for the Q
  // booking class (the one corresponding to the fare option) to 8.
  ClassAvailabilityMap_T lClassAvailabilityMap1;
  const Availability_T lAvl1 (8);
  bool hasInsertOfQBeenSuccessful = lClassAvailabilityMap1.
    insert (ClassAvailabilityMap_T::value_type (lClassPathQ, lAvl1)).second;
  assert (hasInsertOfQBeenSuccessful == true);
  // Add the map to the dedicated list held by the travel solution
  lTS1.addClassAvailabilityMap (lClassAvailabilityMap1);

  // Add the travel solution to the list
  ioTravelSolutionList.push_back (lTS1);

  //
  const std::string lQF12_SegmentDateKey ("QF, 12, 2011-06-10, LHR, SYD, 20:45");

  // Add the segment date key to the travel solution
  TravelSolutionStruct lTS2;
  lTS2.addSegment (lQF12_SegmentDateKey);

  // Fare option number 2
  const ClassCode_T lClassPathY (CLASS_CODE_Y);
  const Fare_T lFare1000 (1000);
  const ChangeFees_T lNoChangeFee (NO_CHANGE_FEES);
  const NonRefundable_T isRefundable (NO_NON_REFUNDABLE);
  const FareOptionStruct lFareOption2 (lClassPathY, lFare1000, lNoChangeFee,
                                       isRefundable, lSaturdayStay);

  // Map of class availabilities: set the availability for the Y
  // booking class (the one corresponding to the fare option) to 9.
  ClassAvailabilityMap_T lClassAvailabilityMap2;
  const Availability_T lAvl2 (9);
  const bool hasInsertOfYBeenSuccessful = lClassAvailabilityMap2.
    insert (ClassAvailabilityMap_T::value_type (lClassPathY, lAvl2)).second;
  assert (hasInsertOfYBeenSuccessful == true);
  // Add the map to the dedicated list held by the travel solution
  lTS2.addClassAvailabilityMap (lClassAvailabilityMap2);

  // Add (a copy of) the fare option
  lTS2.addFareOption (lFareOption2);

  // Fare option number 3
  const Fare_T lFare920 (920);
  const FareOptionStruct lFareOption3 (lClassPathQ, lFare920, lNoChangeFee,
                                       isNonRefundable, lSaturdayStay);

  // Map of class availabilities: set the availability for the Q
  // booking class (the one corresponding to the fare option) to 9.
  hasInsertOfQBeenSuccessful = lClassAvailabilityMap2.
    insert (ClassAvailabilityMap_T::value_type (lClassPathQ, lAvl2)).second;
  assert (hasInsertOfYBeenSuccessful == true);
  // Add the map to the dedicated list held by the travel solution
  lTS2.addClassAvailabilityMap (lClassAvailabilityMap2);

  // Add (a copy of) the fare option
  lTS2.addFareOption (lFareOption3);

  // Add the travel solution to the list
  ioTravelSolutionList.push_back (lTS2);

}

// ////////////////////////////////////////////////////////////////////
```

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

  // Destination
  const AirportCode_T lDestination (AIRPORT_SYD);

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

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

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

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

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

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

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

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

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

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

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

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

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

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

  // Restrictions
  const ChangeFees_T lChangeFees = false;
  const Disutility_T lChangeFeeDisutility = 30;
  const NonRefundable_T lNonRefundable = false;
  const Disutility_T lNonRefundableDisutility = 50;

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

```
                                        lNonRefundable,
                                        lNonRefundableDisutility);

  return oBookingRequest;
}

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

  // Destination
  const AirportCode_T lDestination (AIRPORT_BKK);

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

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

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

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

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

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

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

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

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

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

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

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

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

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

  // Restrictions
  const ChangeFees_T lChangeFees = true;
  const Disutility_T lChangeFeeDisutility = 50;
  const NonRefundable_T lNonRefundable = true;
  const Disutility_T lNonRefundableDisutility = 50;

  // Creation of the booking request structure
  BookingRequestStruct oBookingRequest (lOrigin,
                                        lDestination,
                                        lPOS,
```

```
                                        lPreferredDepartureDate,
                                        lRequestDateTime,
                                        lPreferredCabin,
                                        lPartySize, lChannel,
                                        lTripType, lStayDuration,
                                        lFrequentFlyerType,
                                        lPreferredDepartureTime,
                                        lWTP, lValueOfTime,
                                        lChangeFees, lChangeFeeDisutility,
                                        lNonRefundable,
                                        lNonRefundableDisutility);

  return oBookingRequest;
}

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

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

  // Add the airline feature object to the SQ inventory
  const AirlineFeatureKey lAirlineFeatureSQKey (lAirlineCodeSQ);
  AirlineFeature& lAirlineFeatureSQ =
    FacBom<AirlineFeature>::instance().create (lAirlineFeatureSQKey);
  FacBomManager::setAirlineFeature (lSQInv, lAirlineFeatureSQ);
  FacBomManager::linkWithParent (lSQInv, lAirlineFeatureSQ);
  // Link the airline feature object with the top of the BOM tree
  FacBomManager::addToListAndMap (ioBomRoot, lAirlineFeatureSQ);

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

  // Add the airline feature object to the CX inventory
  const AirlineFeatureKey lAirlineFeatureCXKey (lAirlineCodeCX);
  AirlineFeature& lAirlineFeatureCX =
    FacBom<AirlineFeature>::instance().create (lAirlineFeatureCXKey);
  FacBomManager::setAirlineFeature (lCXInv, lAirlineFeatureCX);
  FacBomManager::linkWithParent (lCXInv, lAirlineFeatureCX);
  // Link the airline feature object with the top of the BOM tree
  FacBomManager::addToListAndMap (ioBomRoot, lAirlineFeatureCX);

  // ////// SQ ///////
  // Step 0.2: Flight-date level
  // Create a FlightDate (SQ11/08-MAR-2010) for SQ's Inventory
  FlightNumber_T lFlightNumber = 11;
  Date_T lDate (2010, 3, 8);
  FlightDateKey lFlightDateKey (lFlightNumber, lDate);

  FlightDate& lSQ11_20100308_FD =
    FacBom<FlightDate>::instance().create (lFlightDateKey);
  FacBomManager::addToListAndMap (lSQInv, lSQ11_20100308_FD);
  FacBomManager::linkWithParent (lSQInv, lSQ11_20100308_FD);

  // Create a (mkt) FlightDate (SQ1200/08-MAR-2010) for SQ's Inventory
  FlightNumber_T lMktFlightNumber = 1200;
  //lDate = Date_T (2010, 3, 8);
```

```
FlightDateKey lMktFlightDateKey (lMktFlightNumber, lDate);

FlightDate& lSQ1200_20100308_FD =
  FacBom<FlightDate>::instance().create (lMktFlightDateKey);
FacBomManager::addToListAndMap (lSQInv, lSQ1200_20100308_FD);
FacBomManager::linkWithParent (lSQInv, lSQ1200_20100308_FD);

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

// Step 0.3: Segment-date level
// Create a first SegmentDate (SIN-BKK) for SQ's Inventory
const AirportCode_T lSIN ("SIN");
const AirportCode_T lBKK ("BKK");
const DateOffset_T l1Day (1);
const DateOffset_T l2Days (2);
const Duration_T l0820 (8, 20, 0);
const Duration_T l1100 (11, 0, 0);
const Duration_T l0340 (3, 40, 0);
SegmentDateKey lSegmentDateKey (lSIN, lBKK);

SegmentDate& lSINBKKSegment =
  FacBom<SegmentDate>::instance().create (lSegmentDateKey);
FacBomManager::addToListAndMap (lSQ11_20100308_FD, lSINBKKSegment);
FacBomManager::linkWithParent (lSQ11_20100308_FD, lSINBKKSegment);

// Add the routing leg key to the SIN-BKK segment.
const std::string lSQSINRoutingLegStr = "SQ;11;2010-Mar-8;SIN";
lSINBKKSegment.addLegKey (lSQSINRoutingLegStr);

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

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

SegmentDate& lMktBKKHKGSegment =
  FacBom<SegmentDate>::instance().create (lMktSegmentDateKey);
FacBomManager::addToListAndMap (lSQ1200_20100308_FD, lMktBKKHKGSegment);
FacBomManager::linkWithParent (lSQ1200_20100308_FD, lMktBKKHKGSegment);

// Add the routing leg key CX;12;2010-Mar-8;BKK to the marketing
// SQ;1200;2010-Mar-8;BKK-HKG segment.
const std::string lCXBKKRoutingLegStr = "CX;12;2010-Mar-8;BKK";
lMktBKKHKGSegment.addLegKey (lCXBKKRoutingLegStr);

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

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

LegDate& lSINLeg = FacBom<LegDate>::instance().create (lLegDateKey);
FacBomManager::addToListAndMap (lSQ11_20100308_FD, lSINLeg);
```

```
FacBomManager::linkWithParent (lSQ11_20100308_FD, lSINLeg);

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

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

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

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


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

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

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


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

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

// Create a FareFamily (1) for the (mkt) Segment BKK-HKG, cabin Y on SQ's Inv
FareFamily& lMktBKKHKGSegmentYCabin1Family =
  FacBom<FareFamily>::instance().create (l1FareFamilyKey);
FacBomManager::addToListAndMap (lMktBKKHKGSegmentYCabin,
                                lMktBKKHKGSegmentYCabin1Family);
FacBomManager::linkWithParent (lMktBKKHKGSegmentYCabin,
                                lMktBKKHKGSegmentYCabin1Family);

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

BookingClass& lSINBKKSegmentYCabin1FamilyYClass =
```

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

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

lSINBKKSegmentYCabin1FamilyYClass.setYield(700);

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

FacBomManager::addToListAndMap (lMktBKKHKGSegmentYCabin,
                                lMktBKKHKGSegmentYCabin1FamilyYClass);
FacBomManager::addToListAndMap (lMktBKKHKGSegment,
                                lMktBKKHKGSegmentYCabin1FamilyYClass);

lMktBKKHKGSegmentYCabin1FamilyYClass.setYield(700);


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

BookingClass& lSINBKKSegmentYCabin1FamilyMClass =
  FacBom<BookingClass>::instance().create (lMBookingClassKey);
FacBomManager::addToListAndMap (lSINBKKSegmentYCabin1Family,
                                lSINBKKSegmentYCabin1FamilyMClass);
FacBomManager::linkWithParent (lSINBKKSegmentYCabin1Family,
                               lSINBKKSegmentYCabin1FamilyMClass);

FacBomManager::addToListAndMap (lSINBKKSegmentYCabin,
                                lSINBKKSegmentYCabin1FamilyMClass);
FacBomManager::addToListAndMap (lSINBKKSegment,
                                lSINBKKSegmentYCabin1FamilyMClass);

lSINBKKSegmentYCabin1FamilyMClass.setYield(500);

// Create a BookingClass (M) for the (mkt) Segment BKK-HKG, cabin Y,
// fare family 1 on SQ's Inv
BookingClass& lMktBKKHKGSegmentYCabin1FamilyMClass =
  FacBom<BookingClass>::instance().create (lMBookingClassKey);
FacBomManager::addToListAndMap (lMktBKKHKGSegmentYCabin1Family,
                                lMktBKKHKGSegmentYCabin1FamilyMClass);
FacBomManager::linkWithParent (lMktBKKHKGSegmentYCabin1Family,
                               lMktBKKHKGSegmentYCabin1FamilyMClass);

FacBomManager::addToListAndMap (lMktBKKHKGSegmentYCabin,
                                lMktBKKHKGSegmentYCabin1FamilyMClass);
FacBomManager::addToListAndMap (lMktBKKHKGSegment,
                                lMktBKKHKGSegmentYCabin1FamilyMClass);

lMktBKKHKGSegmentYCabin1FamilyMClass.setYield(500);

/* ================================================================================ */
```

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

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

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

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

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

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

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

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

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

// Add total forecast info for cabin Y.
```

```
const WTP_T lWTP400 = 400.0;
const WTPDemandPair_T lWTP400Mean60StdDev6 (lWTP400, lMean60StdDev6);
lSQ_SINBKK_OnDDate.setTotalForecast (lY, lWTP400Mean60StdDev6);

// Add demand info (optional).
lCabinClassPairList.clear();
lCabinClassPairList.push_back(lCC_YM1);
const MeanStdDevPair_T lMean20StdDev1 (20.0, 1.0);
const Yield_T lYield500 = 500.0;
const YieldDemandPair_T lYield500Mean20StdDev1 (lYield500, lMean20StdDev1);
lSQ_SINBKK_OnDDate.setDemandInformation (lCabinClassPairList,
                                         lYield500Mean20StdDev1);

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

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

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

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

// Demand info is not added for purely marketed O&Ds
// Add demand info
// lCabinClassPairList.clear();
// lCabinClassPairList.push_back(lCC_YM2);
// lMktSQ_BKKHKG_OnDDate.setDemandInformation (lCabinClassPairList, 500.0, 20.0, 1.0);
*****************************************************************************/


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

FlightDate& lCX12_20100308_FD =
  FacBom<FlightDate>::instance().create (lFlightDateKey);
FacBomManager::addToListAndMap (lCXInv, lCX12_20100308_FD);
FacBomManager::linkWithParent (lCXInv, lCX12_20100308_FD);

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

FlightDate& lCX1100_20100308_FD =
  FacBom<FlightDate>::instance().create (lMktFlightDateKey);
FacBomManager::addToListAndMap (lCXInv, lCX1100_20100308_FD);
FacBomManager::linkWithParent (lCXInv, lCX1100_20100308_FD);

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

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

lSegmentDateKey = SegmentDateKey (lBKK, lHKG);

SegmentDate& lBKKHKGSegment =
  FacBom<SegmentDate>::instance().create (lSegmentDateKey);
FacBomManager::addToListAndMap (lCX12_20100308_FD, lBKKHKGSegment);
FacBomManager::linkWithParent (lCX12_20100308_FD, lBKKHKGSegment);

// Add the routing leg key to the marketing BKK-HKG segment.
lBKKHKGSegment.addLegKey (lCXBKKRoutingLegStr);

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

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

SegmentDate& lMktSINBKKSegment =
  FacBom<SegmentDate>::instance().create (lMktSegmentDateKey);
FacBomManager::addToListAndMap (lCX1100_20100308_FD, lMktSINBKKSegment);
FacBomManager::linkWithParent (lCX1100_20100308_FD, lMktSINBKKSegment);

// Add the routing leg key SQ;11;2010-Mar-8;SIN to the marketing
// CX;1100;2010-Mar-8;SIN-BKK segment.
lMktSINBKKSegment.addLegKey (lSQSINRoutingLegStr);

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

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

LegDate& lBKKLeg = FacBom<LegDate>::instance().create (lLegDateKey);
FacBomManager::addToListAndMap (lCX12_20100308_FD, lBKKLeg);
FacBomManager::linkWithParent (lCX12_20100308_FD, lBKKLeg);

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

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

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

// Create a SegmentCabin (Y) for the (mkt) Segment SIN-BKK of CX's Inventory
```

```
SegmentCabin& lMktSINBKKSegmentYCabin =
  FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
FacBomManager::addToListAndMap (lMktSINBKKSegment, lMktSINBKKSegmentYCabin);
FacBomManager::linkWithParent (lMktSINBKKSegment, lMktSINBKKSegmentYCabin);

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

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

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

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


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

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

lBKKHKGSegmentYCabin1FamilyYClass.setYield(700);

// Create a BookingClass (Y) for the (mkt) Segment SIN-BKK, cabin Y,
// fare family 1 on CX's Inv
BookingClass& lMktSINBKKSegmentYCabin1FamilyYClass =
  FacBom<BookingClass>::instance().create (lYBookingClassKey);
FacBomManager::addToListAndMap (lMktSINBKKSegmentYCabin1Family,
                                lMktSINBKKSegmentYCabin1FamilyYClass);
FacBomManager::linkWithParent (lMktSINBKKSegmentYCabin1Family,
                               lMktSINBKKSegmentYCabin1FamilyYClass);

FacBomManager::addToListAndMap (lMktSINBKKSegmentYCabin,
                                lMktSINBKKSegmentYCabin1FamilyYClass);
FacBomManager::addToListAndMap (lMktSINBKKSegment,
                                lMktSINBKKSegmentYCabin1FamilyYClass);

lMktSINBKKSegmentYCabin1FamilyYClass.setYield(700);
```

```
//Create a BookingClass (M) for the
// Segment BKK-HKG, cabin Y, fare family 1 on CX's Inv
BookingClass& lBKKHKGSegmentYCabin1FamilyMClass =
  FacBom<BookingClass>::instance().create (lMBookingClassKey);
FacBomManager::addToListAndMap (lBKKHKGSegmentYCabin1Family,
                                lBKKHKGSegmentYCabin1FamilyMClass);
FacBomManager::linkWithParent (lBKKHKGSegmentYCabin1Family,
                               lBKKHKGSegmentYCabin1FamilyMClass);

FacBomManager::addToListAndMap (lBKKHKGSegmentYCabin,
                                lBKKHKGSegmentYCabin1FamilyMClass);
FacBomManager::addToListAndMap (lBKKHKGSegment,
                                lBKKHKGSegmentYCabin1FamilyMClass);

lBKKHKGSegmentYCabin1FamilyMClass.setYield(500);

// Create a BookingClass (M) for the (mkt) Segment SIN-BKK, cabin Y,
// fare family 1 on CX's Inv
BookingClass& lMktSINBKKSegmentYCabin1FamilyMClass =
  FacBom<BookingClass>::instance().create (lMBookingClassKey);
FacBomManager::addToListAndMap (lMktSINBKKSegmentYCabin1Family,
                                lMktSINBKKSegmentYCabin1FamilyMClass);
FacBomManager::linkWithParent (lMktSINBKKSegmentYCabin1Family,
                               lMktSINBKKSegmentYCabin1FamilyMClass);

FacBomManager::addToListAndMap (lMktSINBKKSegmentYCabin,
                                lMktSINBKKSegmentYCabin1FamilyMClass);
FacBomManager::addToListAndMap (lMktSINBKKSegment,
                                lMktSINBKKSegmentYCabin1FamilyMClass);

lMktSINBKKSegmentYCabin1FamilyMClass.setYield(500);

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

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

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

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

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

// Add demand info
lCabinClassPairList.clear();
lCabinClassPairList.push_back(lCC_YM1);
lCabinClassPairList.push_back(lCC_YM2);
lCX_SINHKG_OnDDate.setDemandInformation (lCabinClassPairList,
                                         lYield850Mean20StdDev2);

lCabinClassPairList.clear();
lCabinClassPairList.push_back(lCC_YY1);
lCabinClassPairList.push_back(lCC_YY2);
lCX_SINHKG_OnDDate.setDemandInformation (lCabinClassPairList,
```

```
                                        lYield1200Mean10StdDev1);

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

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

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

  // Demand info is not added for purely marketed O&Ds
  // Add demand info
  // lCabinClassPairList.clear();
  // lCabinClassPairList.push_back(lCC_YM1);
  // lMktCX_SINBKK_OnDDate.setDemandInformation (lCabinClassPairList, 500.0, 20.0, 1.0);
  *************************************************************************/

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

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

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

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

  // Add demand info
  lCabinClassPairList.clear();
  lCabinClassPairList.push_back(lCC_YM2);
  lCX_BKKHKG_OnDDate.setDemandInformation (lCabinClassPairList,
                                           lYield500Mean20StdDev1);

  lCabinClassPairList.clear();
  lCabinClassPairList.push_back(lCC_YY2);
  const YieldDemandPair_T lYield700Mean10StdDev1 (lYield700, lMean10StdDev1 );
  lCX_BKKHKG_OnDDate.setDemandInformation (lCabinClassPairList,
                                           lYield700Mean10StdDev1);

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

  lFlightNumber = FlightNumber_T (11);
```

```
  FlightPeriodKey lFlightPeriodKey (lFlightNumber, lPeriodStruct);

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

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

  SegmentPeriodKey lSegmentPeriodKey (lSIN, lBKK);

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

  lSINBKKSegmentPeriod.setBoardingTime (l0820);
  lSINBKKSegmentPeriod.setOffTime (l1100);
  lSINBKKSegmentPeriod.setElapsedTime (l0340);
  ClassList_String_T lYM ("YM");
  lSINBKKSegmentPeriod.addCabinBookingClassList (lY,lYM);

  // CX:
  // Step 1: flight period level
  // Create a flight period for CX12:
  lFlightNumber = FlightNumber_T (12);

  lFlightPeriodKey = FlightPeriodKey(lFlightNumber, lPeriodStruct);

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

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

  lSegmentPeriodKey = SegmentPeriodKey (lBKK, lHKG);

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

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

}

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


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

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

```
// Set the fare date-period primary key.
const Date_T lDateRangeStart (2010, boost::gregorian::Mar, 01);
const Date_T lDateRangeEnd (2010, boost::gregorian::Mar, 31);
const DatePeriod_T lDateRange (lDateRangeStart, lDateRangeEnd);
const DatePeriodKey lDatePeriodKey (lDateRange);

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

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

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

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

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

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

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

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

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

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

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

```
  FacBomManager::linkWithParent (lSINBKKFareFeatures, lSQAirlineYClassList);

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

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

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

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

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

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

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

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

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

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

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

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

  const AirlineClassListKey lCXAirlineYClassListKey (lCXAirlineCodeList,
                                                     lYClassCodeList);

  const AirlineClassListKey lCXAirlineMClassListKey (lCXAirlineCodeList,
                                                     lMClassCodeList);

  // Create the AirlineClassListKey and link it to the FareFeatures object.
  AirlineClassList& lCXAirlineYClassList =
```

```
  FacBom<AirlineClassList>::instance().create (lCXAirlineYClassListKey);
lCXAirlineYClassList.setFare(700);
FacBomManager::addToListAndMap (lBKKHKGFareFeatures, lCXAirlineYClassList);
FacBomManager::linkWithParent (lBKKHKGFareFeatures, lCXAirlineYClassList);

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

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

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

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

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

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

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

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

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

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

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

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

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

```
                                                          lY_YClassCodeList);

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

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

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



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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

AirlineClassList& lRAC_CXAirlineMClassList =
  FacBom<AirlineClassList>::instance().create (lCXAirlineMClassListKey);
lRAC_CXAirlineMClassList.setYield(500);
```

```
    FacBomManager::addToListAndMap (lBKKHKGYieldFeatures,
                                    lRAC_CXAirlineMClassList);
    FacBomManager::linkWithParent (lBKKHKGYieldFeatures,
                                   lRAC_CXAirlineMClassList);

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

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

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

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

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

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

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

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

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

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

    AirlineClassList& lRAC_SQ_CXAirlineMClassList =
      FacBom<AirlineClassList>::instance().create (lSQ_CXAirlineMClassListKey);
    lRAC_SQ_CXAirlineMClassList.setYield(850);
    FacBomManager::addToListAndMap (lSINHKGYieldFeatures,
                                    lRAC_SQ_CXAirlineMClassList);
    FacBomManager::linkWithParent (lSINHKGYieldFeatures,
                                   lRAC_SQ_CXAirlineMClassList);

  }

}

/*!


 */
// ////////////////////////////////////////////////////////////////////
// Import section
```

```
// //////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// StdAir
#include <stdair/factory/FacBomManager.hpp>
#include <stdair/factory/FacCloneBom.hpp>
#include <stdair/command/CmdCloneBomManager.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/BomRetriever.hpp>

namespace stdair {

  // //////////////////////////////////////////////////////////////////
  void CmdCloneBomManager::cloneBomRoot (const BomRoot& iBomRoot,
                                         BomRoot& ioCloneBomRoot) {

    // Check whether there are Inventory objects
    const bool hasInventoryList = BomManager::hasList<Inventory> (iBomRoot);
    if (hasInventoryList == true) {

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

        // Clone the current inventory
        Inventory& lCloneInventory = cloneInventory (*lInv_ptr, ioCloneBomRoot);
        FacBomManager::addToListAndMap (ioCloneBomRoot, lCloneInventory);
        FacBomManager::linkWithParent (ioCloneBomRoot, lCloneInventory);
      }
    }

    // Check whether there are Airport Pair objects
    const bool hastAirportPairList =
      BomManager::hasList<AirportPair> (iBomRoot);
    if (hastAirportPairList == true) {

      // Browse the airport pairs
      const AirportPairList_T& lAirportPairList =
        BomManager::getList<AirportPair> (iBomRoot);
      for (AirportPairList_T::const_iterator itAirportPair =
             lAirportPairList.begin();
           itAirportPair != lAirportPairList.end(); ++itAirportPair) {
        const AirportPair* lAirportPair_ptr = *itAirportPair;
        assert (lAirportPair_ptr != NULL);

        // Clone the current airport pair
        AirportPair& lCloneAirportPair = cloneAirportPair (*lAirportPair_ptr);
        FacBomManager::addToListAndMap (ioCloneBomRoot, lCloneAirportPair);
        FacBomManager::linkWithParent (ioCloneBomRoot, lCloneAirportPair);
      }
    }
  }

  // //////////////////////////////////////////////////////////////////
  Inventory& CmdCloneBomManager::cloneInventory (const Inventory& iInventory,
                                                 BomRoot& ioCloneBomRoot) {

    Inventory& lCloneInventory =
      FacCloneBom<Inventory>::instance().clone (iInventory);

    // Check whether there are FlightDate objects
    const bool hasFlighDateList = BomManager::hasList<FlightDate> (iInventory);
```

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

    // Clone the current flight-date
    FlightDate& lCloneFD = cloneFlightDate (*lFD_ptr);
    FacBomManager::addToListAndMap (lCloneInventory, lCloneFD);
    FacBomManager::linkWithParent (lCloneInventory, lCloneFD);
  }
}

// Check if the inventory contains a list of partners
const bool hasPartnerList = BomManager::hasList<Inventory> (iInventory);
if (hasPartnerList == true) {

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

  for (InventoryList_T::const_iterator itInv =
         lPartnerInventoryList.begin();
       itInv != lPartnerInventoryList.end(); ++itInv) {
    const Inventory* lInv_ptr = *itInv;
    assert (lInv_ptr != NULL);

    // Clone the current partnership inventory
    Inventory& lClonePartnerInventory = cloneInventory (*lInv_ptr,
                                                        ioCloneBomRoot);
    FacBomManager::addToListAndMap (lCloneInventory,
                                    lClonePartnerInventory);
    FacBomManager::linkWithParent (lCloneInventory,
                                   lClonePartnerInventory);
  }
}

// Check whether there are O&D date objects
const bool hasOnDList = BomManager::hasList<OnDDate> (iInventory);
if (hasOnDList == true){

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

  for (OnDDateList_T::const_iterator itOnD = lOnDDateList.begin();
       itOnD != lOnDDateList.end(); ++itOnD) {
    const OnDDate* lOnDDate_ptr = *itOnD;
    assert (lOnDDate_ptr != NULL);

    // Clone the current O&D date
    OnDDate& lCloneOnDDate = cloneOnDDate (*lOnDDate_ptr);
    FacBomManager::addToListAndMap (lCloneInventory, lCloneOnDDate);
    FacBomManager::linkWithParent (lCloneInventory, lCloneOnDDate);
  }
}

// Check whether there are Flight Period objects
const bool hasFlightPeriodList =
  BomManager::hasList<FlightPeriod> (iInventory);
if (hasFlightPeriodList == true) {

  // Browse the flight-periods
  const FlightPeriodList_T& lFlightPeriodList =
```

```
      BomManager::getList<FlightPeriod> (iInventory);
    for (FlightPeriodList_T::const_iterator itFlightPeriod =
           lFlightPeriodList.begin();
         itFlightPeriod != lFlightPeriodList.end(); ++itFlightPeriod) {
      const FlightPeriod* lFlightPeriod_ptr = *itFlightPeriod;
      assert (lFlightPeriod_ptr != NULL);

      // Clone the current flight period
      FlightPeriod& lCloneFlightPeriod = cloneFlightPeriod (*lFlightPeriod_ptr);
      FacBomManager::addToListAndMap (lCloneInventory, lCloneFlightPeriod);
      FacBomManager::linkWithParent (lCloneInventory, lCloneFlightPeriod);
    }
  }

  // Check whether there is an airline feature object
  const AirlineFeature* lAirlineFeature_ptr =
    BomManager::getObjectPtr<AirlineFeature,Inventory> (iInventory,
                                                        iInventory.getAirlineCode());
  if (lAirlineFeature_ptr != NULL) {
    // Clone the current airline feature object
    AirlineFeature& lCloneAirlineFeature =
      cloneAirlineFeature (*lAirlineFeature_ptr);
    FacBomManager::setAirlineFeature (lCloneInventory, lCloneAirlineFeature);
    FacBomManager::linkWithParent (lCloneInventory, lCloneAirlineFeature);
    // Link the airline feature object with the top of the BOM tree
    FacBomManager::addToListAndMap (ioCloneBomRoot, lCloneAirlineFeature);
  }

  return lCloneInventory;
}

// ////////////////////////////////////////////////////////////////////
AirlineFeature& CmdCloneBomManager::
cloneAirlineFeature (const AirlineFeature& iAirlineFeature) {

  AirlineFeature& lCloneAirlineFeature =
    FacCloneBom<AirlineFeature>::instance().clone (iAirlineFeature);

  return lCloneAirlineFeature;
}


// ////////////////////////////////////////////////////////////////////
OnDDate& CmdCloneBomManager::cloneOnDDate (const OnDDate& iOnDDate) {

  OnDDate& lCloneOnDDate =
    FacCloneBom<OnDDate>::instance().clone (iOnDDate);

  return lCloneOnDDate;
}

// ////////////////////////////////////////////////////////////////////
FlightDate& CmdCloneBomManager::
cloneFlightDate (const FlightDate& iFlightDate) {

  FlightDate& lCloneFlightDate =
    FacCloneBom<FlightDate>::instance().clone (iFlightDate);

  // Check whether there are LegDate objects
  const bool hasLegDateList = BomManager::hasList<LegDate> (iFlightDate);
  if (hasLegDateList == true) {

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

```
      const LegDate* lLD_ptr = *itLD;
      assert (lLD_ptr != NULL);

      // Clone the current leg-date
      LegDate& lCloneLegDate = cloneLegDate (*lLD_ptr);
      FacBomManager::addToListAndMap (lCloneFlightDate, lCloneLegDate);
      FacBomManager::linkWithParent (lCloneFlightDate, lCloneLegDate);
    }
  }

  // Check whether there are SegmentDate objects
  const bool hasSegmentDateList =
    BomManager::hasList<SegmentDate> (iFlightDate);
  if (hasSegmentDateList == true) {

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

      // Clone the current segment-date
      SegmentDate& lCloneSegmentDate = cloneSegmentDate (*lSD_ptr);
      FacBomManager::addToListAndMap (lCloneFlightDate, lCloneSegmentDate);
      FacBomManager::linkWithParent (lCloneFlightDate, lCloneSegmentDate);

    }
  }

  return lCloneFlightDate;
}

// ////////////////////////////////////////////////////////////////////
LegDate& CmdCloneBomManager::cloneLegDate (const LegDate& iLegDate) {

  LegDate& lCloneLegDate =
    FacCloneBom<LegDate>::instance().clone (iLegDate);

  // Check whether there are LegCabin objects
  const bool hasLegCabinList = BomManager::hasList<LegCabin> (iLegDate);
  if (hasLegCabinList == true) {
    // Browse the leg-cabins
    const LegCabinList_T& lLegCabinList =
      BomManager::getList<LegCabin> (iLegDate);
    for (LegCabinList_T::const_iterator itLC = lLegCabinList.begin();
         itLC != lLegCabinList.end(); ++itLC) {
      const LegCabin* lLC_ptr = *itLC;
      assert (lLC_ptr != NULL);

      // Clone the current leg-cabin
      LegCabin& lCloneLegCabin = cloneLegCabin (*lLC_ptr);
      FacBomManager::addToListAndMap (lCloneLegDate, lCloneLegCabin);
      FacBomManager::linkWithParent (lCloneLegDate, lCloneLegCabin);
    }
  }

  return lCloneLegDate;
}

// ////////////////////////////////////////////////////////////////////
LegCabin& CmdCloneBomManager::cloneLegCabin (const LegCabin& iLegCabin) {

  LegCabin& lCloneLegCabin =
    FacCloneBom<LegCabin>::instance().clone (iLegCabin);
```

```
  // Check whether there are Bucket objects
  const bool hasBucketList = BomManager::hasList<Bucket> (iLegCabin);
  if (hasBucketList == true) {
    // Browse the buckets
    const BucketList_T& lBucketList =
      BomManager::getList<Bucket> (iLegCabin);
    for (BucketList_T::const_iterator itBucket = lBucketList.begin();
         itBucket != lBucketList.end(); ++itBucket) {
      const Bucket* lBucket_ptr = *itBucket;
      assert (lBucket_ptr != NULL);

      // Clone the current bucket
      Bucket& lCloneBucket = cloneBucket (*lBucket_ptr);
      FacBomManager::addToListAndMap (lCloneLegCabin, lCloneBucket);
      FacBomManager::linkWithParent (lCloneLegCabin, lCloneBucket);
    }
  }

  return lCloneLegCabin;
}

// ////////////////////////////////////////////////////////////////////
Bucket& CmdCloneBomManager::cloneBucket (const Bucket& iBucket) {

  Bucket& lCloneBucket =
    FacCloneBom<Bucket>::instance().clone (iBucket);

  return lCloneBucket;
}

// ////////////////////////////////////////////////////////////////////
SegmentDate& CmdCloneBomManager::
cloneSegmentDate (const SegmentDate& iSegmentDate) {

  SegmentDate& lCloneSegmentDate =
    FacCloneBom<SegmentDate>::instance().clone (iSegmentDate);

  // Check whether there are SegmentCabin objects
  const bool hasSegmentCabinList =
    BomManager::hasList<SegmentCabin> (iSegmentDate);
  if (hasSegmentCabinList == true) {
    // Browse the segment-cabins
    const SegmentCabinList_T& lSegmentCabinList =
      BomManager::getList<SegmentCabin> (iSegmentDate);
    for (SegmentCabinList_T::const_iterator itSC = lSegmentCabinList.begin();
         itSC != lSegmentCabinList.end(); ++itSC) {
      const SegmentCabin* lSC_ptr = *itSC;
      assert (lSC_ptr != NULL);

      // Clone the current segment-cabin
      SegmentCabin& lCloneSegmentCabin = cloneSegmentCabin (*lSC_ptr);
      FacBomManager::addToListAndMap (lCloneSegmentDate, lCloneSegmentCabin);
      FacBomManager::linkWithParent (lCloneSegmentDate, lCloneSegmentCabin);

      linkBookingClassesWithSegment (lCloneSegmentDate,
                                     lCloneSegmentCabin);

    }
  }
  return lCloneSegmentDate;
}

// ////////////////////////////////////////////////////////////////////
void CmdCloneBomManager::
linkBookingClassesWithSegment (SegmentDate& iCloneSegmentDate,
                               SegmentCabin& iCloneSegmentCabin) {
```

```
    // Browse the fare families to link the booking-classes to the
    // segment-cabin and to the segment-date
    const bool hasFareFamilyList =
      BomManager::hasList<FareFamily> (iCloneSegmentCabin);
    if (hasFareFamilyList == true) {
      const FareFamilyList_T& lCloneFFList =
        BomManager::getList<FareFamily> (iCloneSegmentCabin);
      for (FareFamilyList_T::const_iterator itCloneFF = lCloneFFList.begin();
           itCloneFF != lCloneFFList.end(); ++itCloneFF) {
        const FareFamily* lCloneFF_ptr = *itCloneFF;
        assert (lCloneFF_ptr != NULL);

        // Browse the list of booking classes
        const bool hasBookingClasslist =
          BomManager::hasList<BookingClass> (*lCloneFF_ptr);
        if (hasBookingClasslist == true) {
          const BookingClassList_T& lCloneBCList =
            BomManager::getList<BookingClass> (*lCloneFF_ptr);
          for (BookingClassList_T::const_iterator itCloneBC =
                 lCloneBCList.begin();
               itCloneBC != lCloneBCList.end(); ++itCloneBC) {
            const BookingClass* lCloneBC_ptr = *itCloneBC;
            assert (lCloneBC_ptr != NULL);

            // Link the booking-class to the segment-cabin
            stdair::FacBomManager::addToListAndMap (iCloneSegmentCabin,
                                                    *lCloneBC_ptr);

            // Link the booking-class to the segment-date
            stdair::FacBomManager::addToListAndMap (iCloneSegmentDate,
                                                    *lCloneBC_ptr);
          }
        }
      }
    }
  }

  // ////////////////////////////////////////////////////////////////////
  SegmentCabin& CmdCloneBomManager::
  cloneSegmentCabin (const SegmentCabin& iSegmentCabin) {

    SegmentCabin& lCloneSegmentCabin =
      FacCloneBom<SegmentCabin>::instance().clone (iSegmentCabin);

    // Check whether there are fare family objects
    const bool hasFareFamilyList =
      BomManager::hasList<FareFamily> (iSegmentCabin);
    if (hasFareFamilyList == true) {
      // Browse the fare families
      const FareFamilyList_T& lFareFamilyList =
        BomManager::getList<FareFamily> (iSegmentCabin);
      for (FareFamilyList_T::const_iterator itFF = lFareFamilyList.begin();
           itFF != lFareFamilyList.end(); ++itFF) {
        const FareFamily* lFF_ptr = *itFF;
        assert (lFF_ptr != NULL);

        // Clone the current fare-family
        FareFamily& lCloneFareFamily = cloneFareFamily (*lFF_ptr);
        FacBomManager::addToListAndMap (lCloneSegmentCabin, lCloneFareFamily);
        FacBomManager::linkWithParent (lCloneSegmentCabin, lCloneFareFamily);
      }
    }

    return lCloneSegmentCabin;
  }

  // ////////////////////////////////////////////////////////////////////
```

```
FareFamily& CmdCloneBomManager::
cloneFareFamily (const FareFamily& iFareFamily) {
  FareFamily& lCloneFareFamily =
    FacCloneBom<FareFamily>::instance().clone (iFareFamily);

  // Check whether there are booking classes objects
  const bool hasBookingClassList =
    BomManager::hasList<BookingClass> (iFareFamily);
  if (hasBookingClassList == true) {
    // Browse the list of booking classes
    const BookingClassList_T& lBookingClassList =
      BomManager::getList<BookingClass> (iFareFamily);
    for (BookingClassList_T::const_iterator itBookingClass =
           lBookingClassList.begin();
         itBookingClass != lBookingClassList.end(); ++itBookingClass) {
      const BookingClass* lBC_ptr = *itBookingClass;
      assert (lBC_ptr != NULL);

      // Clone the current booking class
      BookingClass& lCloneBookingClass = cloneBookingClass (*lBC_ptr);
      FacBomManager::addToListAndMap (lCloneFareFamily, lCloneBookingClass);
      FacBomManager::linkWithParent (lCloneFareFamily, lCloneBookingClass);
    }
  }

  return lCloneFareFamily;
}

// ////////////////////////////////////////////////////////////////////
BookingClass& CmdCloneBomManager::
cloneBookingClass (const BookingClass& iBookingClass) {

  BookingClass& lCloneBookingClass =
    FacCloneBom<BookingClass>::instance().clone (iBookingClass);

  return lCloneBookingClass;
}

// ////////////////////////////////////////////////////////////////////
AirportPair& CmdCloneBomManager::
cloneAirportPair (const AirportPair& iAirportPair) {

  AirportPair& lCloneAirportPair =
    FacCloneBom<AirportPair>::instance().clone (iAirportPair);

  // Check whether there are date-period objects
  const bool hasDatePeriodList =
    BomManager::hasList<DatePeriod> (iAirportPair);
  if (hasDatePeriodList == true) {
    // Browse the date-periods
    const DatePeriodList_T& lDatePeriodList =
      BomManager::getList<DatePeriod> (iAirportPair);
    for (DatePeriodList_T::const_iterator itDatePeriod =
           lDatePeriodList.begin();
         itDatePeriod != lDatePeriodList.end(); ++itDatePeriod) {
      const DatePeriod* lDatePeriod_ptr = *itDatePeriod;
      assert (lDatePeriod_ptr != NULL);

      // Clone the current date-period
      DatePeriod& lCloneDatePeriod = cloneDatePeriod (*lDatePeriod_ptr);
      FacBomManager::addToListAndMap (lCloneAirportPair, lCloneDatePeriod);
      FacBomManager::linkWithParent (lCloneAirportPair, lCloneDatePeriod);
    }
  }

  return lCloneAirportPair;
}
```

```
// //////////////////////////////////////////////////////////////////
DatePeriod& CmdCloneBomManager::
cloneDatePeriod (const DatePeriod& iDatePeriod) {

  DatePeriod& lCloneDatePeriod =
    FacCloneBom<DatePeriod>::instance().clone (iDatePeriod);

  // Check whether there are pos-channel objects
  const bool hasPosChannelList =
    BomManager::hasList<PosChannel> (iDatePeriod);
  if (hasPosChannelList == true) {
    // Browse the pos-channels
    const PosChannelList_T& lPosChannelList =
      BomManager::getList<PosChannel> (iDatePeriod);
    for (PosChannelList_T::const_iterator itPosChannel =
           lPosChannelList.begin();
         itPosChannel != lPosChannelList.end(); ++itPosChannel) {
      const PosChannel* lPosChannel_ptr = *itPosChannel;
      assert (lPosChannel_ptr != NULL);

      // Clone the current pos-channel
      PosChannel& lClonePosChannel = clonePosChannel (*lPosChannel_ptr);
      FacBomManager::addToListAndMap (lCloneDatePeriod, lClonePosChannel);
      FacBomManager::linkWithParent (lCloneDatePeriod, lClonePosChannel);
    }
  }

  return lCloneDatePeriod;
}


// //////////////////////////////////////////////////////////////////
PosChannel& CmdCloneBomManager::
clonePosChannel (const PosChannel& iPosChannel) {

  PosChannel& lClonePosChannel =
    FacCloneBom<PosChannel>::instance().clone (iPosChannel);

  // Check whether there are time-period objects
  const bool hasTimePeriodList =
    BomManager::hasList<TimePeriod> (iPosChannel);
  if (hasTimePeriodList == true) {
    // Browse the time-periods
    const TimePeriodList_T& lTimePeriodList =
      BomManager::getList<TimePeriod> (iPosChannel);
    for (TimePeriodList_T::const_iterator itTimePeriod =
           lTimePeriodList.begin();
         itTimePeriod != lTimePeriodList.end(); ++itTimePeriod) {
      const TimePeriod* lTimePeriod_ptr = *itTimePeriod;
      assert (lTimePeriod_ptr != NULL);

      // Clone the current time-period
      TimePeriod& lCloneTimePeriod = cloneTimePeriod (*lTimePeriod_ptr);
      FacBomManager::addToListAndMap (lClonePosChannel, lCloneTimePeriod);
      FacBomManager::linkWithParent (lClonePosChannel, lCloneTimePeriod);
    }
  }

  return lClonePosChannel;
}

// //////////////////////////////////////////////////////////////////
TimePeriod& CmdCloneBomManager::
cloneTimePeriod (const TimePeriod& iTimePeriod) {

  TimePeriod& lCloneTimePeriod =
```

```
    FacCloneBom<TimePeriod>::instance().clone (iTimePeriod);

  // Check whether there are fare-feature objects
  const bool hasFareFeaturesList =
    BomManager::hasList<FareFeatures> (iTimePeriod);
  if (hasFareFeaturesList == true) {
    // Browse the fare-features
    const FareFeaturesList_T& lFareFeaturesList =
      BomManager::getList<FareFeatures> (iTimePeriod);
    for (FareFeaturesList_T::const_iterator itFF = lFareFeaturesList.begin();
         itFF != lFareFeaturesList.end(); ++itFF) {
      const FareFeatures* lFF_ptr = *itFF;
      assert (lFF_ptr != NULL);

      // Clone the current fare-feature
      FareFeatures& lCloneFareFeatures =
        cloneFeatures<FareFeatures> (*lFF_ptr);
      FacBomManager::addToListAndMap (lCloneTimePeriod, lCloneFareFeatures);
      FacBomManager::linkWithParent (lCloneTimePeriod, lCloneFareFeatures);
    }
  }

  // Check whether there are yield-feature objects
  const bool hasYieldFeaturesList =
    BomManager::hasList<YieldFeatures> (iTimePeriod);
  if (hasYieldFeaturesList == true) {
    // Browse the yield-features
    const YieldFeaturesList_T& lYieldFeaturesList =
      BomManager::getList<YieldFeatures> (iTimePeriod);
    for (YieldFeaturesList_T::const_iterator itYF =
           lYieldFeaturesList.begin();
         itYF != lYieldFeaturesList.end(); ++itYF) {
      const YieldFeatures* lYF_ptr = *itYF;
      assert (lYF_ptr != NULL);

      // Clone the current yield-feature
      YieldFeatures& lCloneYieldFeatures =
        cloneFeatures<YieldFeatures> (*lYF_ptr);
      FacBomManager::addToListAndMap (lCloneTimePeriod, lCloneYieldFeatures);
      FacBomManager::linkWithParent (lCloneTimePeriod, lCloneYieldFeatures);
    }
  }

  return lCloneTimePeriod;
}

// ////////////////////////////////////////////////////////////////////
template <typename FEATURE_TYPE>
FEATURE_TYPE& CmdCloneBomManager::
cloneFeatures (const FEATURE_TYPE& iFeatures) {

  FEATURE_TYPE& lCloneFeatures =
    FacCloneBom<FEATURE_TYPE>::instance().clone (iFeatures);

  // Check whether there are airline-class list objects
  const bool hasAirlineClassListList =
    BomManager::hasList<AirlineClassList> (iFeatures);
  if (hasAirlineClassListList == true) {
    // Browse the airline-class lists
    const AirlineClassListList_T& lAirlineClassList =
      BomManager::getList<AirlineClassList> (iFeatures);
    for (AirlineClassListList_T::const_iterator itACList =
           lAirlineClassList.begin();
         itACList != lAirlineClassList.end(); ++itACList) {
      const AirlineClassList* lACList_ptr = *itACList;
      assert (lACList_ptr != NULL);
```

```
        // Clone the current airline-class list
        AirlineClassList& lCloneAirlineClassList =
          cloneAirlineClassList (*lACList_ptr);
        FacBomManager::addToListAndMap (lCloneFeatures,
                                        lCloneAirlineClassList);
        FacBomManager::linkWithParent (lCloneFeatures,
                                       lCloneAirlineClassList);
      }
    }

    return lCloneFeatures;
  }

  // ////////////////////////////////////////////////////////////////////
  AirlineClassList& CmdCloneBomManager::
  cloneAirlineClassList (const AirlineClassList& iAirlineClassList) {

    AirlineClassList& lCloneAirlineClassList =
      FacCloneBom<AirlineClassList>::instance().clone (iAirlineClassList);

    return lCloneAirlineClassList;
  }

  // ////////////////////////////////////////////////////////////////////
  FlightPeriod& CmdCloneBomManager::
  cloneFlightPeriod (const FlightPeriod& iFlightPeriod) {

    FlightPeriod& lCloneFlightPeriod =
      FacCloneBom<FlightPeriod>::instance().clone (iFlightPeriod);

    // Check whether there are airline-class list objects
    const bool hasSegmentPeriodList =
      BomManager::hasList<SegmentPeriod> (iFlightPeriod);
    if (hasSegmentPeriodList == true) {
      // Browse the airline-class lists
      const SegmentPeriodList_T& lSegmentPeriodList =
        BomManager::getList<SegmentPeriod> (iFlightPeriod);
      for (SegmentPeriodList_T::const_iterator itSegmentPeriod =
             lSegmentPeriodList.begin();
           itSegmentPeriod != lSegmentPeriodList.end(); ++itSegmentPeriod) {
        const SegmentPeriod* lSegmentPeriod_ptr = *itSegmentPeriod;
        assert (lSegmentPeriod_ptr != NULL);

        // Clone the current airline-class list
        SegmentPeriod& lCloneSegmentPeriod =
          cloneSegmentPeriod (*lSegmentPeriod_ptr);
        FacBomManager::addToListAndMap (lCloneFlightPeriod,
                                        lCloneSegmentPeriod);
        FacBomManager::linkWithParent (lCloneFlightPeriod,
                                       lCloneSegmentPeriod);
      }
    }

    return lCloneFlightPeriod;
  }

  // ////////////////////////////////////////////////////////////////////
  SegmentPeriod& CmdCloneBomManager::
  cloneSegmentPeriod (const SegmentPeriod& iSegmentPeriod) {

    SegmentPeriod& lCloneSegmentPeriod =
      FacCloneBom<SegmentPeriod>::instance().clone (iSegmentPeriod);

    return lCloneSegmentPeriod;
  }

}
```

```
/*!
```

10.3 C++ Class Storing the StdAir Context

```
 */
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// Boost
#if BOOST_VERSION >= 103900
#include <boost/make_shared.hpp>
#else  // BOOST_VERSION >= 103900
#include <boost/shared_ptr.hpp>
#endif // BOOST_VERSION >= 103900
// StdAir
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/factory/FacBom.hpp>
#include <stdair/factory/FacCloneBom.hpp>
#include <stdair/service/STDAIR_ServiceContext.hpp>

namespace stdair {

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

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

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

  // //////////////////////////////////////////////////////////////////////
  void STDAIR_ServiceContext::init() {
    //
    initBomRoot();
    initConfigHolder();
  }

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

  // //////////////////////////////////////////////////////////////////////
  void STDAIR_ServiceContext::initCloneBomRoot() {
    _cloneBomRoot =
      &FacCloneBom<BomRoot>::instance().clone(*_persistentBomRoot);
```

```
  }

  // ////////////////////////////////////////////////////////////////////
  void STDAIR_ServiceContext::initConfigHolder() {
    _configHolderPtr = boost::make_shared<ConfigHolderStruct> ();
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string STDAIR_ServiceContext::shortDisplay() const {
    std::ostringstream oStr;
    oStr << "STDAIR_ServiceContext -- " << _initType
         << " -- DB: " << _dbParams;
    return oStr.str();
  }

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

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

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

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

  // ////////////////////////////////////////////////////////////////////
  ConfigHolderStruct& STDAIR_ServiceContext::getConfigHolder() const {
    assert (_configHolderPtr != NULL);
    return *_configHolderPtr;
  }
}

/*!
```

10.4 People

10.4.1 Project Admins (and Developers)

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

10.4.2 Retired Developers

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

10.4.3 Contributors

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

10.4.4 Distribution Maintainers

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

Note:

(N) - Amadeus employees.

10.5 Coding Rules

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

10.5.1 Default Naming Rules for Variables

Variables names follow Java naming conventions. Examples:

- `lNumberOfPassengers`
- `lSeatAvailability`

10.5.2 Default Naming Rules for Functions

Function names follow Java naming conventions. Example:

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

10.5.3 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`

10.5.4 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`

10.5.5 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.

10.6 Copyright and License

10.6.1 GNU LESSER GENERAL PUBLIC LICENSE

10.6.1.1 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.]
```

10.6.2 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.

10.6.3 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.

10.6.3.1 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.

10.6.3.2 END OF TERMS AND CONDITIONS

10.6.4 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

10.7 Documentation Rules

10.7.1 General Rules

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

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

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

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

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

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

private:

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

10.7.2 File Header

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

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

10.7.3 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);
```

10.8 Main features

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

10.8.1 Standard Airline IT Business Object Model (BOM)

- (Airline) Network-related classes:
  - Network, ReachableUniverse
- (Air) Travel-related classes:
  - TravelSolution, OriginDestination,
- (Airline) Inventory-related classes:
  - Inventory, FlightDate, SegmentDate, SegmentCabin, BookingClass, LegDate, LegCabin, Bucket
- (Airline) Schedule-related classes:
  - FlightPeriod, SegmentPeriod, LegPeriod
- (Simulated) Passenger-related demand classes:
  - DemandStream, BookingRequest
- (Air) Price-related classes:
  - YieldStore

10.8.2 Architecture of the StdAir library

- Separate structure and content classes
- Boost.Fusion

10.9 Make a Difference

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

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

- Start using StdAir
- Tell your friends about StdAir and help them to get started using it
- If you find a bug, report it to us (on the dedicated Sourceforge's Trac Web site). Without your help we can never hope to produce a bug free code.
- Help us to improve the documentation by providing information about documentation bugs
- Answer support requests in the StdAir discussion forums on SourceForge. If you know the answer to a question, help others to overcome their StdAir problems.
- Help us to improve our algorithms. If you know of a better way (e.g., that is faster or requires less memory) to implement some of our algorithms, then let us know.
- Help to port StdAir to new platforms. If you manage to compile StdAir on a new platform, then tell how you did it.
- Send your code. If you have a good StdAir compatible code, which you can release under the LGPL, and you think it should be included in StdAir, then send it to the communauty.
- Become an StdAir developer. Send us (see the People page) an e-mail and tell what you can do for StdAir.

10.10 Make a new release

10.10.1 Introduction

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

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

10.10.2 Initialisation

Clone locally the full Git project:

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

10.10.3 Branch creation

Create the branch, on your local clone, corresponding to the new release (say, 0.5.0):

```
cd ~/dev/sim/stdairgit
git checkout trunk
git checkout -b 0.5.0
```

Update the version in the various build system files, replacing 99.99.99 by the correct version number:

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

Update the version and add a change-log in the ChangeLog and in the RPM specification files:

```
vi ChangeLog
vi stdair.spec
```

10.10.4 Commit and publish the release branch

Commit the new release:

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

10.10.5 Update the change-log in the trunk as well

Update the change-log in the ChangeLog and RPM specification files:

```
cd ~/dev/sim/stdairgit
git checkout trunk
vi ChangeLog
vi stdair.spec
```

Commit the change-logs and publish the trunk (main development branch):

```
git commit -m "[Doc] Integrated the change-log of the release 0.5.0."
git push
```

10.10.6 Create distribution packages

Create the distribution packages using the following command:

```
cd ~/dev/sim/stdairgit
git checkout 0.5.0
rm -rf build && mkdir -p build
cd build
cmake -DCMAKE_INSTALL_PREFIX=/home/user/dev/deliveries/stdair-0.5.0 \
 -DCMAKE_BUILD_TYPE:STRING=Debug -DINSTALL_DOC:BOOL=ON ..
make check && make dist
```

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

10.10.7 Generation the RPM packages

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

```
cd ~/dev/sim/stdairgit
git checkout 0.5.0
rm -rf build && mkdir -p build
cd build
cmake -DCMAKE_INSTALL_PREFIX=/home/user/dev/deliveries/stdair-99.99.99 \
 -DCMAKE_BUILD_TYPE:STRING=Debug -DINSTALL_DOC:BOOL=ON ..
make dist
```

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

```
cp stdair.spec ~/dev/packages/SPECS \
  && cp stdair-0.5.0.tar.bz2 ~/dev/packages/SOURCES
cd ~/dev/packages/SPECS
rpmbuild -ba stdair.spec
rpmlint -i ../SPECS/stdair.spec ../SRPMS/stdair-0.5.0-1.fc15.src.rpm \
  ../RPMS/noarch/stdair-* ../RPMS/i686/stdair-*
```

10.10.8 Update distributed change log

Update the NEWS and ChangeLog files with appropriate information, including what has changed since the previous release. Then commit and push the changes into the StdAir's Git repository.

10.10.9 Create the binary package, including the documentation

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

```
make package
```

The output binary package will be named, for instance, stdair-0.5.0-Linux.tar.bz2. That package contains both the HTML and PDF documentation. The binary package contains also the executables and shared libraries, as well as C++ header files, but all of those do not interest us for now.

10.10.10 Upload the files to SourceForge

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

10.10.11 Upload the documentation to SourceForge

In order to update the Web site files, either:

- synchronise them with rsync and SSH:

  ```
  cd ~/dev/sim/stdairgit
  git checkout 0.5.0
  rsync -aiv doc/html/ doc/latex/refman.pdf joe,stdair@web.sourceforge.net:htdocs/
  ```

  where -aiv options mean:

  - `-a`: archive/mirror mode; equals `-rlptgoD` (no `-H`, `-A`, `-X`)
  - `-v`: increase verbosity
  - `-i`: output a change-summary for all updates
  - Note the trailing slashes (/) at the end of both the source and target directories. It means that the content of the source directory (`doc/html`), rather than the directory itself, has to be copied into the content of the target directory.

- or use the `SourceForge Shell service`.

10.10.12 Make a new post

- submit a new entry in the `SourceForge project-related news feed`
- make a new post on the `SourceForge hosted WordPress blog`
- and update, if necessary, `Trac tickets`.

10.10.13 Send an email on the announcement mailing-list

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

10.11 Installation

10.11.1 Table of Contents

10.11.2 Fedora/RedHat Linux distributions

Note that on Fedora/RedHat Linux distributions, RPM packages are available and can be installed with your usual package manager. For instance:

```
yum -y install stdair-devel stdair-doc
```

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

10.11.3 StdAir Requirements

StdAir should compile without errors or warnings on most GNU/Linux systems, on UNIX systems like Solaris SunOS, and on POSIX based environments for Microsoft Windows like Cygwin or MinGW with MSYS. It can be also built on Microsoft Windows NT/2000/XP/Vista/7 using Microsoft's Visual C++ .NET, but our support for this compiler is limited. For GNU/Linux, SunOS, Cygwin and MinGW we assume that you have at least the following GNU software installed on your computer:

- GNU Autotools:
  - autoconf,
  - automake,
  - libtool,
  - make, version 3.72.1 or later (check version with 'make –version')
- GCC - GNU C++ Compiler (g++), version 4.3.x or later (check version with 'gcc –version')
- Boost - C++ STL extensions, version 1.35 or later (check version with 'grep "define BOOST_LIB_VERSION" /usr/include/boost/version.hpp')
- MySQL - Database client libraries, version 5.0 or later (check version with 'mysql –version')
- 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 StdAir.

10.11.4 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.

10.11.5 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.

10.11.6 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.

10.11.7 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.

10.11.8 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’.

10.11.9 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
```

10.11.10 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'.

10.11.11 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.

10.11.12 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
```

10.11.13 '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/stdair-0.50.0 \
 -DLIB_SUFFIX=64 -DCMAKE_BUILD_TYPE:STRING=Debug -DINSTALL_DOC:BOOL=ON ..
-- The C compiler identification is GNU
-- The CXX compiler identification is GNU
-- Check for working C compiler: /usr/lib64/ccache/gcc
-- Check for working C compiler: /usr/lib64/ccache/gcc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: /usr/lib64/ccache/c++
-- Check for working CXX compiler: /usr/lib64/ccache/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Requires Git without specifying any version
-- Current Git revision name: e8beb4d11ff9b1af6b3f3e9ff1e92250aee0291a trunk
-- Requires PythonLibs-2.7
-- Found PythonLibs: /usr/lib64/libpython2.7.so (Required is at least version "2.7")
-- Found PythonLibs 2.7
-- Requires Boost-1.41
-- Boost version: 1.46.0
-- Found the following Boost libraries:
--   program_options
--   date_time
--   iostreams
--   serialization
--   filesystem
--   unit_test_framework
--   python
-- Found Boost version: 1.46.0
-- Found BoostWrapper: /usr/include (Required is at least version "1.41")
-- Requires ZeroMQ-2.0
-- Found ZeroMQ: /usr/lib64/libzmq.so (Required is at least version "2.0")
-- Found ZeroMQ version: 2.1
-- Requires MySQL-5.1
-- Using mysql-config: /usr/bin/mysql_config
-- Found MySQL: /usr/lib64/mysql/libmysqlclient.so (Required is at least version "5.1")
-- Found MySQL version: 5.5.14
-- Requires SOCI-3.0
-- Using soci-config: /usr/bin/soci-config
-- SOCI headers are buried
-- Found SOCI: /usr/lib64/libsoci_core.so (Required is at least version "3.0")
-- Found SOCIMySQL: /usr/lib64/libsoci_mysql.so (Required is at least version "3.0")
-- Found SOCI with MySQL back-end support version: 3.0.0
-- Requires Doxygen-1.7
-- Found Doxygen: /usr/bin/doxygen
-- Found DoxygenWrapper: /usr/bin/doxygen (Required is at least version "1.7")
-- Found Doxygen version: 1.7.4
-- Had to set the linker language for 'stdairlib' to CXX
-- Had to set the linker language for 'stdairuiclib' to CXX
```

```
-- Test 'StdAirTest' to be built with 'MPBomRoot.cpp;MPInventory.cpp;StandardAirlineITTestSuite.cpp'
--
-- ============================================================
-- ----------------------------------
-- ---     Project Information    ---
-- ----------------------------------
-- PROJECT_NAME ................... : stdair
-- PACKAGE_PRETTY_NAME ............ : StdAir
-- PACKAGE ........................ : stdair
-- PACKAGE_NAME ................... : STDAIR
-- PACKAGE_VERSION ................ : 0.50.0
-- GENERIC_LIB_VERSION ............ : 0.50.0
-- GENERIC_LIB_SOVERSION .......... : 99.99
--
-- ----------------------------------
-- ---     Build Configuration    ---
-- ----------------------------------
-- Modules to build ............... : stdair
-- Libraries to build/install ..... : stdairlib;stdairuicllib
-- Binaries to build/install ...... : stdair
-- Modules to test ................ : stdair
-- Binaries to test ............... : StdAirTesttst
--
-- * Module ....................... : stdair
--   + Layers to build ............ : .;basic;bom;factory;dbadaptor;command;service
--   + Dependencies on other layers :
--   + Libraries to build/install . : stdairlib;stdairuicllib
--   + Executables to build/install : stdair
--   + Tests to perform ........... : StdAirTesttst
--
-- BUILD_SHARED_LIBS .............. : ON
-- CMAKE_BUILD_TYPE ............... : Debug
--  * CMAKE_C_FLAGS ............... :
--  * CMAKE_CXX_FLAGS ............. : -O2 -g -pipe -Wall -Wp,-D_FORTIFY_SOURCE=2 -fexceptions -fstack-prot
--  * BUILD_FLAGS ................. :
--  * COMPILE_FLAGS ............... :
-- CMAKE_MODULE_PATH .............. : /home/user/dev/sim/stdair/stdairgithub/config/
-- CMAKE_INSTALL_PREFIX ........... : /home/user/dev/deliveries/stdair-0.50.0
--
-- * Doxygen:
--   - DOXYGEN_VERSION ............. : 1.7.4
--   - DOXYGEN_EXECUTABLE .......... : /usr/bin/doxygen
--   - DOXYGEN_DOT_EXECUTABLE ...... : /usr/bin/dot
--   - DOXYGEN_DOT_PATH ............ : /usr/bin
--
-- ----------------------------------
-- --- Installation Configuration ---
-- ----------------------------------
-- INSTALL_LIB_DIR ................ : /home/user/dev/deliveries/stdair-0.50.0/lib64
-- INSTALL_BIN_DIR ................ : /home/user/dev/deliveries/stdair-0.50.0/bin
-- INSTALL_INCLUDE_DIR ............ : /home/user/dev/deliveries/stdair-0.50.0/include
-- INSTALL_DATA_DIR ............... : /home/user/dev/deliveries/stdair-0.50.0/share
-- INSTALL_SAMPLE_DIR ............. : /home/user/dev/deliveries/stdair-0.50.0/share/stdair/samples
-- INSTALL_DOC .................... : ON
--
-- ----------------------------------
-- ---   Packaging Configuration  ---
-- ----------------------------------
-- CPACK_PACKAGE_CONTACT .......... : Denis Arnaud <denis_arnaud - at - users dot sourceforge dot net>
-- CPACK_PACKAGE_VENDOR ........... : Denis Arnaud
-- CPACK_PACKAGE_VERSION .......... : 0.50.0
-- CPACK_PACKAGE_DESCRIPTION_FILE . : /home/user/dev/sim/stdair/stdairgithub/README
-- CPACK_RESOURCE_FILE_LICENSE .... : /home/user/dev/sim/stdair/stdairgithub/COPYING
-- CPACK_GENERATOR ................ : TBZ2
-- CPACK_DEBIAN_PACKAGE_DEPENDS ... :
-- CPACK_SOURCE_GENERATOR ......... : TBZ2;TGZ
-- CPACK_SOURCE_PACKAGE_FILE_NAME . : stdair-0.50.0
```

```
--
-- --------------------------------
-- ---     External libraries    ---
-- --------------------------------
--
-- * Python:
--   - PYTHONLIBS_VERSION ......... : 2.7
--   - PYTHON_LIBRARIES ........... : /usr/lib64/libpython2.7.so
--   - PYTHON_INCLUDE_PATH ........ : /usr/include/python2.7
--   - PYTHON_INCLUDE_DIRS ........ : /usr/include/python2.7
--   - PYTHON_DEBUG_LIBRARIES ..... :
--   - Python_ADDITIONAL_VERSIONS . :
--
-- * ZeroMQ:
--   - ZeroMQ_VERSION ............. : 2.1
--   - ZeroMQ_LIBRARIES ........... : /usr/lib64/libzmq.so
--   - ZeroMQ_INCLUDE_DIR ......... : /usr/include
--
-- * Boost:
--   - Boost_VERSION .............. : 104600
--   - Boost_LIB_VERSION .......... : 1_46
--   - Boost_HUMAN_VERSION ........ : 1.46.0
--   - Boost_INCLUDE_DIRS ......... : /usr/include
--   - Boost required components .. : program_options;date_time;iostreams;serialization;filesystem;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
--
-- Change a value with: cmake -D<Variable>=<Value>
-- ================================================================
--
-- Configuring done
-- Generating done
-- Build files have been written to: /home/user/dev/sim/stdair/stdairgithub/build
```

It is recommended that you check if your library has been compiled and linked properly and works as expected. To do so, you should execute the testing process 'make check'. As a result, you should obtain a similar report:

```
[  0%] Built target hdr_cfg_stdair
[ 97%] Built target stdairlib
[100%] Built target StdAirTesttst
Scanning dependencies of target check_stdairtst
Test project /home/user/dev/sim/stdair/stdairgithub/build/test/stdair
    Start 1: StdAirTesttst
1/1 Test #1: StdAirTesttst ....................   Passed    0.02 sec

100% tests passed, 0 tests failed out of 1

Total Test time (real) =   0.27 sec
[100%] Built target check_stdairtst
Scanning dependencies of target check
[100%] Built target check
```

Check if all the executed tests PASSED. If not, please contact us by filling a bug-report.

Finally, you should install the compiled and linked library, include files and (optionally) HTML and PDF documentation by typing:

```
make install
```

Depending on the PREFIX settings during configuration, you might need the root (administrator) access to perform this step.

Eventually, you might invoke the following command

```
make clean
```

to remove all files created during compilation process, or even

```
cd ~/dev/sim/stdairgit
rm -rf build && mkdir build
cd build
```

to remove everything.

10.12 Linking with StdAir

10.12.1 Table of Contents

10.12.2 Introduction

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

10.12.3 Using the pkg-config command

'pkg-config' is a helper tool used when compiling applications and libraries. It helps you insert the correct compiler and linker options. The syntax of the 'pkg-config' is as follows:

```
pkg-config <options> <library_name>
```

For instance, assuming that you need to compile an StdAir based program 'my_prog.cpp', you should use the following command:

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

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

10.12.4 Using the stdair-config script

StdAir provides a shell script called `'stdair-config'`, which is installed by default in `'$prefix/bin'` (`'/usr/local/bin'`) directory. It can be used to simplify compilation and linking of StdAir based programs. The usage of this script is quite similar to the usage of the `'pkg-config'` command.

Assuming that you need to compile the program `'my_prog`.cpp' you can now do that with the following command:

```
g++ `stdair-config --cflags` -o my_prog my_prog.cpp `stdair-config --libs`
```

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

```
stdair-config --help
```

If the `'stdair-config'` command is not found by your shell, you should add its location `'$prefix/bin'` to the `PATH` environment variable, e.g.:

```
export PATH=/usr/local/bin:$PATH
```

10.12.5 M4 macro for the GNU Autotools

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

- `'STDAIR_VERSION'` (e.g., defined to 0.2.0)
- `'STDAIR_CFLAGS'` (e.g., defined to `'-I${prefix}/include'`)
- `'STDAIR_LIBS'` (e.g., defined to `'-L${prefix}/lib -lstdair'`)

10.12.6 Using StdAir with dynamic linking

When using static linking some of the library routines in StdAir are copied into your executable program. This can lead to unnecessary large executables. To avoid having too large executable files you may use dynamic linking instead. Dynamic linking means that the actual linking is performed when the program is executed. This requires that the system is able to locate the shared StdAir library file during your program execution. If you install the StdAir library using a non-standard prefix, the `'LD_LIBRARY_PATH'` environment variable might be used to inform the linker of the dynamic library location, e.g.:

```
export LD_LIBRARY_PATH=<StdAir installation prefix>/lib:$LD_LIBRARY_PATH
```

10.13 Test Rules

This section describes how the functionality of the StdAir library should be verified. In the `'test/stdair'` subdirectory, test source files are provided. All functionality should be tested using these test source files.

10.13.1 The Test Source Files

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

The test source file should print relevant data to a standard output that can be used to verify the functionality. All relevant parameter settings should be tested.

The test source file should be placed in the `'test/stdair'` subdirectory and should have a name ending with `'TestSuite.cpp'`.

10.13.2 The Reference File

Consider a test source file named `'YieldTestSuite.cpp'`. A reference file named `'YieldTestSuite.ref'` should accompany the test source file. The reference file contains a reference printout of the standard output generated when running the test program. The reference file should be maintained using version control and updated according to the test source file.

10.13.3 Testing StdAir Library

One can compile and execute all test programs from the `'test/stdair'` sub-directory by typing:

```
% make check
```

after successful compilation of the StdAir library.

10.14 Users Guide

10.14.1 Table of Contents

10.14.2 Introduction

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

10.14.3 Get Started

10.14.3.1 Get the StdAir library

10.14.3.2 Build the StdAir project To run the configuration script the first time, go to the top directory (where the StdAir package has been un-packed), and issue either of the following two commands, depending on whether the StdAir project has been checked out from the Subversion repository or downloaded as a tar-ball package from the Sourceforge Web site:

- ./autogen.sh
- ./configure

10.14.3.3 Build and Run the Tests

10.14.3.4 Install the StdAir Project (Binaries, Documentation)

10.14.4 Exploring the Predefined BOM Tree

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

10.14.4.1 Airline Distribution BOM Tree

- stdair::TravelSolutionStruct

10.14.4.2 Airline Network BOM Tree

- stdair::FlightPeriod

10.14.4.3 Airline Inventory BOM Tree

- stdair::Inventory
- stdair::FlightDate

Airline Inventory Marketing BOM Tree

- stdair::SegmentDate
- stdair::SegmentCabin
- stdair::FareFamily
- stdair::BookingClass

Airline Inventory Operating BOM Tree

- `stdair::LegDate`
- `stdair::LegCabin`
- `stdair::Bucket`

10.14.5 Extending the BOM Tree

10.15 Supported Systems

10.15.1 Table of Contents

10.15.2 Introduction

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

10.15.3 StdAir 3.10.x

10.15.3.1 Linux Systems

Fedora Core 4 with ATLAS

- **Platform**: Intel Pentium 4
- **Operating System**: Fedora Core 4 (x86)
- **Compiler**: g++ (GCC) 4.0.2 20051125
- **StdAir release**: 3.10.0
- **External Libraries**: From FC4 distribution:
  - `fftw3.i386-3.0.1-3`
  - `fftw3-devel.i386-3.0.1-3`
  - `atlas-sse2.i386-3.6.0-8.fc4`
  - `atlas-sse2-devel.i386-3.6.0-8.fc4`
  - `blas.i386-3.0-35.fc4`
  - `lapack.i386-3.0-35.fc4`
- **Tests Status**: All tests PASSED
- **Comments**: StdAir configured with:

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

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

Gentoo Linux with ACML

- **Platform**: AMD Sempron 3000+
- **Operating System**: Gentoo Linux 2006.0 (x86 arch)
- **Compiler(s)**: g++ (GCC) 3.4.5
- **StdAir release**: 3.10.1
- **External Libraries**: Compiled and installed from portage tree:
  - `sci-libs/acml-3.0.0`
- **Tests Status**: All tests PASSED
- **Comments**: BLAS and LAPACK libs set by using the following system commands:

```
% eselect blas set ACML
% eselect lapack set ACML
```

StdAir configured with:

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

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

Gentoo Linux with ATLAS

- **Platform**: Intel Pentium M Centrino
- **Operating System**: Gentoo Linux 2006.0 (x86)
- **Compiler**: g++ (GCC) 3.4.5
- **StdAir release**: 3.10.1
- **External Libraries**: Compiled and installed from portage tree:
  - `sci-libs/fftw-3.1`
  - `sci-libs/blas-atlas-3.6.0-r1`
  - `sci-libs/lapack-atlas-3.6.0`
- **Tests Status**: All tests PASSED
- **Comments**: BLAS and LAPACK libs set by using the following system commands:

```
% eselect blas set ATLAS
% eselect lapack set ATLAS
```

  StdAir configured with:

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

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

Gentoo Linux with MKL

- **Platform**: Intel Pentium M Centrino
- **Operating System**: Gentoo Linux 2006.0 (x86 arch)
- **Compiler**: g++ (GCC) 3.4.5
- **StdAir release**: 3.10.0
- **External Libraries**: Intel Math Kernel Library (MKL) 8.0.1 installed manually in the following directory: `/opt/intel/mkl/8.0.1`
- **Tests Status**: All tests PASSED
- **Comments**: StdAir configured using the following commands:

```
% export LDFLAGS="-L/opt/intel/mkl/8.0.1/lib/32"
% export CPPFLAGS="-I/opt/intel/mkl/8.0.1/include"
% ./configure
```

- **Date**: February 28, 2006
- **Tester**: Adam Piatyszek (ediap)

Gentoo Linux with NetLib's BLAS and LAPACK

- **Platform**: Intel Pentium M Centrino
- **Operating System**: Gentoo Linux 2006.0 (x86)
- **Compiler**: g++ (GCC) 3.4.5
- **StdAir release**: 3.10.1
- **External Libraries**: Compiled and installed from portage tree:
  - `sci-libs/fftw-3.1`
  - `sci-libs/blas-reference-19940131-r2`
  - `sci-libs/cblas-reference-20030223`
  - `sci-libs/lapack-reference-3.0-r2`
- **Tests Status**: All tests PASSED
- **Comments**: BLAS and LAPACK libs set by using the following system commands:

  ```
  % blas-config reference
  % lapack-config reference
  ```

  StdAir configured with:

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

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

Red Hat Enterprise Linux with StdAir External

- **Platform**: Intel Pentium 4
- **Operating System**: Red Hat Enterprise Linux AS release 4 (Nahant Update 2)
- **Compiler**: g++ (GCC) 3.4.4 20050721 (Red Hat 3.4.4-2)
- **StdAir release**: 3.10.0
- **External Libraries**: BLAS, CBLAS, LAPACK and FFTW libraries from StdAir External 2.1.1 package
- **Tests Status**: All tests PASSED
- **Date**: March 7, 2006
- **Tester**: Erik G. Larsson

SUSE Linux 10.0 with NetLib's BLAS and LAPACK

- **Platform**: Intel Pentium 4 CPU 3.20GHz (64-bit)
- **Operating System**: SUSE Linux 10.0 (x86_64)
- **Compiler(s)**: g++ (GCC) 4.0.2
- **StdAir release**: 3.10.0
- **External Libraries**: BLAS, LAPACK and FFTW libraries installed from OpenSuse 10.0 RPM repository:
  - `blas-3.0-926`
  - `lapack-3.0-926`
  - `fftw3-3.0.1-114`
  - `fftw3-threads-3.0.1-114`
  - `fftw3-devel-3.0.1-114`
- **Tests Status**: All tests PASSED
- **Comments**: StdAir configured with:

```
% export CXXFLAGS="-m64 -march=nocona -O3 -pipe"
% ./configure --with-lapack="/usr/lib64/liblapack.so.3"
```

- **Date**: March 1, 2006
- **Tester**: Adam Piatyszek (ediap)

SUSE Linux 10.0 with MKL

- **Platform**: Intel Pentium 4 CPU 3.20GHz (64-bit)
- **Operating System**: SUSE Linux 10.0 (x86_64)
- **Compiler(s)**: g++ (GCC) 4.0.2
- **StdAir release**: 3.10.0
- **External Libraries**: Intel Math Kernel Library (MKL) 8.0.1 installed manually in the following directory: `/opt/intel/mkl/8.0.1`
- **Tests Status**: All tests PASSED
- **Comments**: StdAir configured with:

```
% export CXXFLAGS="-m64 -march=nocona -O3 -pipe"
% export LDFLAGS="-L/opt/intel/mkl/8.0.1/lib/em64t"
% export CPPFLAGS="-I/opt/intel/mkl/8.0.1/include"
% ./configure
```

- **Date**: March 1, 2006
- **Tester**: Adam Piatyszek (ediap)

10.15.3.2 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)
- **StdAir release**: 3.10.1
- **External Libraries**: Installed from Cygwin's repository:
  - `fftw-3.0.1-2`
  - `fftw-dev-3.0.1-1`
  - `lapack-3.0-4`
- **Tests Status**: All tests PASSED
- **Comments**: Only static library can be built. StdAir configured with:

```
% ./configure
```

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

Microsoft Windows XP with Cygwin and ATLAS

- **Platform**: AMD Sempron 3000+
- **Operating System**: Microsoft Windows XP SP2, Cygwin 1.5.19-4
- **Compiler(s)**: g++ (GCC) 3.4.4 (cygming special)
- **StdAir release**: 3.10.1
- **External Libraries**: Installed from Cygwin's repository:
  - `fftw-3.0.1-2`
  - `fftw-dev-3.0.1-1`

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

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

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

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

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

Microsoft Windows XP with Cygwin and ACML

- **Platform**: AMD Sempron 3000+
- **Operating System**: Microsoft Windows XP SP2, Cygwin 1.5.19-4
- **Compiler(s)**: g++ (GCC) 3.4.4 (cygming special)
- **StdAir release**: 3.10.2
- **External Libraries**: ACML version 3.1.0 (`acml3.1.0-32-win32-g77.exe`) installed into a default directory, i.e. `"c:\Program Files\AMD\acml3.1.0"`
- **Tests Status**: All tests PASSED
- **Comments**: Only static library can be built. StdAir configured with:

```
% export LDFLAGS="-L/cygdrive/c/Progra~1/AMD/acml3.1.0/gnu32/lib"
% export CPPFLAGS="-I/cygdrive/c/Progra~1/AMD/acml3.1.0/gnu32/include"
% ./configure --enable-debug
```

- **Date**: May 15, 2006
- **Tester**: Adam Piatyszek (ediap)

Microsoft Windows XP with MinGW, MSYS and ACML

- **Platform**: AMD Sempron 3000+
- **Operating System**: Microsoft Windows XP SP2, MinGW 5.0.2, MSYS 1.0.10
- **Compiler(s)**: g++ (GCC) 3.4.4 (mingw special)
- **StdAir release**: 3.10.2
- **External Libraries**: ACML version 3.1.0 (`acml3.1.0-32-win32-g77.exe`) installed into a default directory, i.e. `"c:\Program Files\AMD\acml3.1.0"`
- **Tests Status**: All tests PASSED
- **Comments**: Only static library can be built. StdAir configured with:

```
% export LDFLAGS="-L/c/Progra~1/AMD/acml3.1.0/gnu32/lib"
% export CPPFLAGS="-I/c/Progra~1/AMD/acml3.1.0/gnu32/include"
% ./configure --enable-debug
```

- **Date**: May 15, 2006
- **Tester**: Adam Piatyszek (ediap)

Microsoft Windows XP with MinGW, MSYS and StdAir External

- **Platform**: AMD Sempron 3000+
- **Operating System**: Microsoft Windows XP SP2, MinGW 5.0.2, MSYS 1.0.10
- **Compiler(s)**: g++ (GCC) 3.4.4 (mingw special)
- **StdAir release**: 3.10.5
- **External Libraries**: BLAS, CBLAS, LAPACK and FFTW libraries from StdAir External 2.2.0 package
- **Tests Status**: All tests PASSED
- **Comments**: Only static library can be built. StdAir configured with:

```
% export LDFLAGS="-L/usr/local/lib"
% export CPPFLAGS="-I/usr/local/include"
% export CXXFLAGS="-Wall -O3 -march=athlon-tbird -pipe"
% ./configure --disable-html-doc
```

- **Date**: August 11, 2006
- **Tester**: Adam Piatyszek (ediap)

Microsoft Windows XP with MS Visual C++ and Intel MKL

- **Platform**: AMD Sempron 3000+
- **Operating System**: Microsoft Windows XP SP2
- **Compiler(s)**: Microsoft Visual C++ 2005 .NET
- **StdAir release**: 3.10.5
- **External Libraries**: Intel Math Kernel Library (MKL) 8.1 installed manually in the following directory: `"C:\Program Files\Intel\MKL\8.1"`
- **Tests Status**: Not fully tested. Some StdAir based programs compiled and run with success.
- **Comments**: Only static library can be built. StdAir built by opening the `"win32\stdair.vcproj"` project file in MSVC++ and executing `"Build → Build Solution"` command from menu.
- **Date**: August 11, 2006
- **Tester**: Adam Piatyszek (ediap)

10.15.3.3 Unix Systems

SunOS 5.9 with StdAir External

- **Platform**: SUNW, Sun-Blade-100 (SPARC)
- **Operating System**: SunOS 5.9 Generic_112233-10
- **Compiler(s)**: g++ (GCC) 3.4.5
- **StdAir release**: 3.10.2
- **External Libraries**: BLAS, CBLAS, LAPACK and FFTW libraries from StdAir External 2.1.1 package. The following configuration command has been used:

```
% export CFLAGS="-mcpu=ultrasparc -O2 -pipe -funroll-all-loops"
% ./configure
```

- **Tests Status**: All tests PASSED
- **Comments**: StdAir configured with:

```
% export LDFLAGS="-L/usr/local/lib"
% export CPPFLAGS="-I/usr/local/include"
% export CXXFLAGS="-mcpu=ultrasparc -O2 -pipe"
% ./configure --enable-debug
```

- **Date**: May 15, 2006
- **Tester**: Adam Piatyszek (ediap)

10.16 StdAir Supported Systems (Previous Releases)

10.16.1 StdAir 3.9.1

10.16.2 StdAir 3.9.0

10.16.3 StdAir 3.8.1

10.17 Tutorials

10.17.1 Table of Contents

10.17.2 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).

10.17.2.1 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.

10.17.3 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

10.17.3.1 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:

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

10.17.3.2 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.

```
  const InventoryKey lBAKey (lAirlineCodeBA);
```

Thanks to that key, an airline inventory object, i.e. of type (class) `stdair::Inventory`, can be instantiated. Let us name that airline inventory object `lBAInv`.

```
  Inventory& lBAInv = FacBom<Inventory>::instance().create (lBAKey);
```

10.17.3.3 Link the Inventory Object with the BOM Root Then, both objects have to be linked: the airline inventory object (`stdair::Inventory`) has to be linked with the root of the BOM tree (`stdair::BomRoot`). That operation is as simple as using the `stdair::FacBomManager::addToListAndMap()` method:

```
  FacBomManager::addToListAndMap (ioBomRoot, lBAInv);
  FacBomManager::linkWithParent (ioBomRoot, lBAInv);
```

10.17.3.4 Build Another Airline Inventory Another airline inventory object, corresponding to the Air France (Air France) company, is instantiated the same way:

```
const InventoryKey lAFKey (lAirlineCodeAF);
Inventory& lAFInv = FacBom<Inventory>::instance().create (lAFKey);
FacBomManager::addToListAndMap (ioBomRoot, lAFInv);
FacBomManager::linkWithParent (ioBomRoot, lAFInv);
```

See the corresponding full program (C++ Class Building Sample StdAir BOM Trees) for more details.

10.17.3.5 Dump The BOM Tree Content From the `BomRoot` (of type `stdair::BomRoot`) object instance, the list of airline inventories (of type `stdair::Inventory`) can then be retrieved...

```
const InventoryList_T& lInventoryList =
  BomManager::getList<Inventory> (iBomRoot);
```

... and browsed:

```
  for (InventoryList_T::const_iterator itInv = lInventoryList.begin();
       itInv != lInventoryList.end(); ++itInv, ++invIdx) {
    const Inventory* lInv_ptr = *itInv;
    assert (lInv_ptr != NULL);

    // Retrieve the inventory key (airline code)
    const AirlineCode_T& lAirlineCode = lInv_ptr->getAirlineCode();

    // Display only the requested inventories
    if (iAirlineCode == "all" || iAirlineCode == lAirlineCode) {
      // Get the list of flight-dates for that inventory
      list (oStream, *lInv_ptr, invIdx, iFlightNumber);
    }
  }
}

// ////////////////////////////////////////////////////////////////////
void BomDisplay::list (std::ostream& oStream, const Inventory& iInventory,
                       const unsigned short iInventoryIndex,
                       const FlightNumber_T& iFlightNumber) {
  // Save the formatting flags for the given STL output stream
  FlagSaver flagSaver (oStream);

  // Check whether there are FlightDate objects
  if (BomManager::hasMap<FlightDate> (iInventory) == false) {
    return;
  }

  //
  const AirlineCode_T& lAirlineCode = iInventory.getAirlineCode();
  oStream << iInventoryIndex << ". " << lAirlineCode << std::endl;

  // Browse the flight-dates
  unsigned short lCurrentFlightNumber = 0;
  unsigned short flightNumberIdx = 0;
  unsigned short departureDateIdx = 1;
  const FlightDateMap_T& lFlightDateList =
    BomManager::getMap<FlightDate> (iInventory);
  for (FlightDateMap_T::const_iterator itFD = lFlightDateList.begin();
       itFD != lFlightDateList.end(); ++itFD, ++departureDateIdx) {
    const FlightDate* lFD_ptr = itFD->second;
    assert (lFD_ptr != NULL);

    // Retrieve the key of the flight-date
```

```
    const FlightNumber_T& lFlightNumber = lFD_ptr->getFlightNumber();
    const Date_T& lFlightDateDate = lFD_ptr->getDepartureDate();

    // Display only the requested flight number
    if (iFlightNumber == 0 || iFlightNumber == lFlightNumber) {
      //
      if (lCurrentFlightNumber != lFlightNumber) {
        lCurrentFlightNumber = lFlightNumber;
        ++flightNumberIdx; departureDateIdx = 1;
        oStream << "  " << iInventoryIndex << "." << flightNumberIdx << ". "
                << lAirlineCode << lFlightNumber << std::endl;
      }

      oStream << "    " << iInventoryIndex << "." << flightNumberIdx
              << "." << departureDateIdx << ". "
              << lAirlineCode << lFlightNumber << " / " << lFlightDateDate
              << std::endl;
    }
  }
}

// ////////////////////////////////////////////////////////////////////
void BomDisplay::listAirportPairDateRange (std::ostream& oStream,
                                           const BomRoot& iBomRoot) {
  // Save the formatting flags for the given STL output stream
  FlagSaver flagSaver (oStream);

  // Check whether there are AirportPair objects
  if (BomManager::hasList<AirportPair> (iBomRoot) == false) {
    return;
  }

  const AirportPairList_T& lAirportPairList =
    BomManager::getList<AirportPair> (iBomRoot);
  for (AirportPairList_T::const_iterator itAir = lAirportPairList.begin();
       itAir != lAirportPairList.end(); ++itAir ) {
    const AirportPair* lAir_ptr = *itAir;
    assert (lAir_ptr != NULL);

    // Check whether there are date-period objects
    assert (BomManager::hasList<DatePeriod> (*lAir_ptr) == true);

    // Browse the date-period objects
    const DatePeriodList_T& lDatePeriodList =
      BomManager::getList<DatePeriod> (*lAir_ptr);

    for (DatePeriodList_T::const_iterator itDP = lDatePeriodList.begin();
         itDP != lDatePeriodList.end(); ++itDP) {
      const DatePeriod* lDP_ptr = *itDP;
      assert (lDP_ptr != NULL);

      // Display the date-period object
      oStream << lAir_ptr->describeKey()
              <<" / " << lDP_ptr->describeKey() << std::endl;
    }

  }
}

// ////////////////////////////////////////////////////////////////////
void BomDisplay::csvDisplay (std::ostream& oStream,
```

See the corresponding full program (C++ Utility Class Browsing and Dumping the StdAir BOM Tree) for more details.

10.17.3.6 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 (Command-Line Utility to Demonstrate Typical StdAir Usage) for more details.

10.17.4 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.

10.17.4.1 Extend an Airline Inventory Object For instance, let us assume that some (IT) provider (e.g., you) would like to have a specific implementation of the `Inventory` object. The corresponding class has just to extend the `stdair::Inventory` class:

```
namespace myprovider {
  class Inventory : public stdair::Inventory {
```

The STL containers have to be defined accordingly too:

```
  typedef std::list<Inventory*> InventoryList_T;
```

See the full class definition (Specific Implementation of an Airline Inventory) and implementation (Specific Implementation of an Airline Inventory) for more details.

10.17.4.2 Build the Specific BOM Objects The BOM root object (`stdair::BomRoot`) is instantiated the classical way:

```
  const std::string& lBomRootKeyStr = lPersistentBomRoot.describeKey();
```

Then, the specific implementation of the airline inventory object (`myprovider::Inventory`) can be instantiated the same way as a standard Inventory (`stdair::Inventory`) would be:

```
  const stdair::InventoryKey lBAKey (lBAAirlineCode);
  myprovider::Inventory& lBAInv =
    stdair::FacBom<myprovider::Inventory>::instance().create (lBAKey);
```

Then, the specific implementation of the airline inventory object (`myprovider::Inventory`) is linked to the root of the BOM tree (`stdair::BomRoot`) the same way as the standard Inventory (`stdair::Inventory`) would be:

```
  stdair::FacBomManager::addToList (lBomRoot, lBAInv);
```

Another specific airline inventory object is instantiated the same way:

```
  const stdair::InventoryKey lAFKey (lAFAirlineCode);
  myprovider::Inventory& lAFInv =
    stdair::FacBom<myprovider::Inventory>::instance().create (lAFKey);
  stdair::FacBomManager::addToList (lBomRoot, lAFInv);
```

From the `BomRoot` (of type `stdair::BomRoot`) object instance, the list of specific airline inventories (of type `stdair::Inventory`) can then be retrieved...

```
  const myprovider::InventoryList_T& lInventoryList =
      stdair::BomManager::getList<myprovider::Inventory> (lBomRoot);
```

... and browsed:

```
  for (myprovider::InventoryList_T::const_iterator itInv =
         lInventoryList.begin(); itInv != lInventoryList.end();
       ++itInv, ++idx) {
    const myprovider::Inventory* lInv_ptr = *itInv;
    BOOST_REQUIRE (lInv_ptr != NULL);

    BOOST_CHECK_EQUAL (lInventoryKeyArray[idx], lInv_ptr->describeKey());
    BOOST_CHECK_MESSAGE (lInventoryKeyArray[idx] == lInv_ptr->describeKey(),
                         "They inventory key, '" << lInventoryKeyArray[idx]
                         << "', does not match that of the Inventory object: '"
                         << lInv_ptr->describeKey() << "'");
  }
```

10.17.4.3 Result of the Tutorial Program

When this program is run, the output should look like:

```
Inventory: BA
Inventory: AF
```

See the corresponding full program (Command-Line Test to Demonstrate How To Extend StdAir BOM) for more details.

10.18 Command-Line Utility to Demonstrate Typical StdAir Usage

```
 */
// STL
#include <cassert>
#include <iostream>
#include <sstream>
#include <fstream>
#include <string>
// Boost (Extended STL)
#include <boost/date_time/posix_time/posix_time.hpp>
#include <boost/date_time/gregorian/gregorian.hpp>
#include <boost/program_options.hpp>
#include <boost/tokenizer.hpp>
#include <boost/lexical_cast.hpp>
// StdAir
#include <stdair/stdair_types.hpp>
#include <stdair/bom/BomArchive.hpp>
#include <stdair/bom/BookingRequestStruct.hpp>
#include <stdair/bom/TravelSolutionStruct.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/STDAIR_Service.hpp>
#include <stdair/config/stdair-paths.hpp>
```

```
// //////// Constants //////
const std::string K_STDAIR_DEFAULT_LOG_FILENAME ("stdair.log");

const std::string K_STDAIR_DEFAULT_INPUT_FILENAME (STDAIR_SAMPLE_DIR
                                                   "/schedule01.csv");

const bool K_STDAIR_DEFAULT_BUILT_IN_INPUT = false;

const bool K_STDAIR_DEFAULT_BUILT_FOR_RMOL = false;

const bool K_STDAIR_DEFAULT_BUILT_FOR_CRS = false;

const int K_STDAIR_EARLY_RETURN_STATUS = 99;

// ///////// Parsing of Options & Configuration /////////
// A helper function to simplify the main part.
template<class T> std::ostream& operator<< (std::ostream& os,
                                            const std::vector<T>& v) {
  std::copy (v.begin(), v.end(), std::ostream_iterator<T> (std::cout, " "));
  return os;
}

int readConfiguration (int argc, char* argv[], bool& ioIsBuiltin,
                       bool& ioIsForRMOL, bool& ioIsForCRS,
                       stdair::Filename_T& ioInputFilename,
                       std::string& ioLogFilename) {
  // Default for the built-in input
  ioIsBuiltin = K_STDAIR_DEFAULT_BUILT_IN_INPUT;

  // Default for the RMOL input
  ioIsForRMOL = K_STDAIR_DEFAULT_BUILT_FOR_RMOL;

  // Default for the CRS input
  ioIsForCRS = K_STDAIR_DEFAULT_BUILT_FOR_CRS;

  // Declare a group of options that will be allowed only on command line
  boost::program_options::options_description generic ("Generic options");
  generic.add_options()
    ("prefix", "print installation prefix")
    ("version,v", "print version string")
    ("help,h", "produce help message");

  // Declare a group of options that will be allowed both on command
  // line and in config file

  boost::program_options::options_description config ("Configuration");
  config.add_options()
    ("builtin,b",
     "The sample BOM tree can be either built-in or parsed from an input file. That latter must then be gi
    ("rmol,r",
     "Build a sample BOM tree for RMOL (i.e., a dummy flight-date with a single leg-cabin)")
    ("crs,c",
     "Build a sample BOM tree for CRS")
    ("input,i",
     boost::program_options::value< std::string >(&ioInputFilename)->default_value(K_STDAIR_DEFAULT_INPUT_
     "(CVS) input file for the demand distributions")
    ("log,l",
     boost::program_options::value< std::string >(&ioLogFilename)->default_value(K_STDAIR_DEFAULT_LOG_FILE
     "Filename for the logs")
    ;

  // Hidden options, will be allowed both on command line and
  // in config file, but will not be shown to the user.
  boost::program_options::options_description hidden ("Hidden options");
  hidden.add_options()
    ("copyright",
```

```
    boost::program_options::value< std::vector<std::string> >(),
    "Show the copyright (license)");

  boost::program_options::options_description cmdline_options;
  cmdline_options.add(generic).add(config).add(hidden);

  boost::program_options::options_description config_file_options;
  config_file_options.add(config).add(hidden);
  boost::program_options::options_description visible ("Allowed options");
  visible.add(generic).add(config);

  boost::program_options::positional_options_description p;
  p.add ("copyright", -1);

  boost::program_options::variables_map vm;
  boost::program_options::
    store (boost::program_options::command_line_parser (argc, argv).
           options (cmdline_options).positional(p).run(), vm);

  std::ifstream ifs ("stdair.cfg");
  boost::program_options::store (parse_config_file (ifs, config_file_options),
                                 vm);
  boost::program_options::notify (vm);

  if (vm.count ("help")) {
    std::cout << visible << std::endl;
    return K_STDAIR_EARLY_RETURN_STATUS;
  }

  if (vm.count ("version")) {
    std::cout << PACKAGE_NAME << ", version " << PACKAGE_VERSION << std::endl;
    return K_STDAIR_EARLY_RETURN_STATUS;
  }

  if (vm.count ("prefix")) {
    std::cout << "Installation prefix: " << PREFIXDIR << std::endl;
    return K_STDAIR_EARLY_RETURN_STATUS;
  }

  if (vm.count ("builtin")) {
    ioIsBuiltin = true;
  }

  if (vm.count ("rmol")) {
    ioIsForRMOL = true;

    // The RMOL sample tree takes precedence over the default built-in BOM tree
    ioIsBuiltin = false;
  }

  if (vm.count ("crs")) {
    ioIsForCRS = true;

    // The RMOL sample tree takes precedence over the default built-in BOM tree
    ioIsBuiltin = false;
  }

  const std::string isBuiltinStr = (ioIsBuiltin == true)?"yes":"no";
  std::cout << "The BOM should be built-in? " << isBuiltinStr << std::endl;

  const std::string isForRMOLStr = (ioIsForRMOL == true)?"yes":"no";
  std::cout << "The BOM should be built-in for RMOL? " << isForRMOLStr
            << std::endl;

  const std::string isForCRSStr = (ioIsForCRS == true)?"yes":"no";
  std::cout << "The BOM should be built-in for CRS? " << isForCRSStr
            << std::endl;
```

```
  if (ioIsBuiltin == false && ioIsForRMOL == false && ioIsForCRS == false) {
    if (vm.count ("input")) {
      ioInputFilename = vm["input"].as< std::string >();
      std::cout << "Input filename is: " << ioInputFilename << std::endl;

    } else {
      std::cerr << "Either one among the -b/--builtin, -r/--rmol, -c/--crs "
                << "or -i/--input options must be specified" << std::endl;
    }
  }

  if (vm.count ("log")) {
    ioLogFilename = vm["log"].as< std::string >();
    std::cout << "Log filename is: " << ioLogFilename << std::endl;
  }

  return 0;
}


// ////////////////// M A I N ////////////////////
int main (int argc, char* argv[]) {

  // State whether the BOM tree should be built-in or parsed from an
  // input file
  bool isBuiltin;

  // State whether a sample BOM tree should be built for RMOL.
  bool isForRMOL;

  // State whether a sample BOM tree should be built for the CRS.
  bool isForCRS;

  // Input file name
  stdair::Filename_T lInputFilename;

  // Output log File
  std::string lLogFilename;

  // Call the command-line option parser
  const int lOptionParserStatus =
    readConfiguration (argc, argv, isBuiltin, isForRMOL, isForCRS,
                       lInputFilename, lLogFilename);

  if (lOptionParserStatus == K_STDAIR_EARLY_RETURN_STATUS) {
    return 0;
  }

  // Set the log parameters
  std::ofstream logOutputFile;
  // Open and clean the log outputfile
  logOutputFile.open (lLogFilename.c_str());
  logOutputFile.clear();

  const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
  stdair::STDAIR_Service stdairService (lLogParams);

  // DEBUG
  STDAIR_LOG_DEBUG ("Welcome to stdair");

  // Check wether or not a (CSV) input file should be read
  if (isBuiltin == true || isForRMOL == true || isForCRS == true) {

    if (isForRMOL == true) {
      // Build the sample BOM tree for RMOL
      stdairService.buildDummyInventory (300);
```

```
    } else if (isForCRS == true) {
      //
      stdair::TravelSolutionList_T lTravelSolutionList;
      stdairService.buildSampleTravelSolutions (lTravelSolutionList);

      // Build the sample BOM tree for CRS
      const stdair::BookingRequestStruct& lBookingRequest =
        stdairService.buildSampleBookingRequest();

      // DEBUG: Display the travel solution and booking request
      STDAIR_LOG_DEBUG ("Booking request: " << lBookingRequest.display());

      const std::string& lCSVDump =
        stdairService.csvDisplay (lTravelSolutionList);
      STDAIR_LOG_DEBUG (lCSVDump);

    } else {
      assert (isBuiltin == true);

      // Build a sample BOM tree
      stdairService.buildSampleBom();
    }

  } else {
    // Read the input file
    //stdairService.readFromInputFile (lInputFilename);

    // DEBUG
    STDAIR_LOG_DEBUG ("StdAir will parse " << lInputFilename
                      << " and build the corresponding BOM tree.");
  }

  // DEBUG: Display the whole persistent BOM tree
  const std::string& lCSVDump = stdairService.csvDisplay ();
  STDAIR_LOG_DEBUG (lCSVDump);

  // Close the Log outputFile
  logOutputFile.close();

  /*
    Note: as that program is not intended to be run on a server in
    production, it is better not to catch the exceptions. When it
    happens (that an exception is throwned), that way we get the
    call stack.
  */

  return 0;
}

/*!
```

10.19 Specific Implementation of a BOM Root

```
 */
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
// StdAir Test
#include <test/stdair/MPBomRoot.hpp>

namespace myprovider {

  // ////////////////////////////////////////////////////////////////////
```

```
  BomRoot::BomRoot (const Key_T& iKey) : stdair::BomRoot (iKey) {
  }

  // ////////////////////////////////////////////////////////////////////
  BomRoot::~BomRoot () {
  }

}
/*!
```

10.20 Specific Implementation of a BOM Root

```
 */

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <string>
// StdAir
#include <stdair/bom/BomRoot.hpp>

namespace myprovider {

  class BomRoot : public stdair::BomRoot {
  public:
    // ////////// Display support methods /////////
    std::string toString() const { return describeKey(); }

    const std::string describeKey() const { return std::string (""); }

  public:
    BomRoot (const Key_T&);
    ~BomRoot();
    BomRoot ();
    BomRoot (const BomRoot&);
  };

}
/*!
```

10.21 Specific Implementation of an Airline Inventory

```
 */
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
// StdAir
#include <stdair/stdair_inventory_types.hpp>
// StdAir Test
#include <test/stdair/MPInventory.hpp>

namespace myprovider {

  // ////////////////////////////////////////////////////////////////////
  Inventory::Inventory (const Key_T& iKey) : stdair::Inventory (iKey) {
  }

  // ////////////////////////////////////////////////////////////////////
  Inventory::~Inventory () {
  }
```

```
  // ////////////////////////////////////////////////////////////////////
  std::string Inventory::toString() const {
    std::ostringstream oStr;
    oStr << _key.toString();
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string Inventory::describeKey() const {
    return _key.toString();
  }

}
/*!
```

10.22 Specific Implementation of an Airline Inventory

```
 */

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <list>
// StdAir
#include <stdair/bom/Inventory.hpp>

namespace myprovider {

  class Inventory : public stdair::Inventory {
  public:
    // ////////// Display support methods /////////
    std::string toString() const;

    const std::string describeKey() const;

  public:
    Inventory (const Key_T&);
    ~Inventory();
    Inventory ();
    Inventory (const Inventory&);
  };

  // ////////// Type definitions /////////
  typedef std::list<Inventory*> InventoryList_T;

}
/*!
```

10.23 Command-Line Test to Demonstrate How To Extend StdAir BOM

```
 */
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <sstream>
#include <fstream>
#include <string>
// Boost MPL
#include <boost/mpl/push_back.hpp>
#include <boost/mpl/vector.hpp>
#include <boost/mpl/at.hpp>
#include <boost/mpl/assert.hpp>
```

```
#include <boost/type_traits/is_same.hpp>
// Boost Unit Test Framework (UTF)
#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MAIN
#define BOOST_TEST_MODULE StdAirTest
#if BOOST_VERSION >= 103900
#include <boost/test/unit_test.hpp>
#else  // BOOST_VERSION >= 103900
#include <boost/test/test_tools.hpp>
#include <boost/test/results_reporter.hpp>
#include <boost/test/unit_test_suite.hpp>
#include <boost/test/output_test_stream.hpp>
#include <boost/test/unit_test_log.hpp>
#include <boost/test/framework.hpp>
#include <boost/test/detail/unit_test_parameters.hpp>
#endif // BOOST_VERSION >= 103900
// Boost Serialisation
#include <boost/archive/text_oarchive.hpp>
#include <boost/archive/text_iarchive.hpp>
// StdAir
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/STDAIR_Service.hpp>
#include <stdair/basic/float_utils.hpp>
#include <stdair/bom/BomDisplay.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/factory/FacBom.hpp>
#include <stdair/factory/FacBomManager.hpp>
// StdAir Test Suite
#include <test/stdair/StdairTestLib.hpp>
#include <test/stdair/MPInventory.hpp>

namespace boost_utf = boost::unit_test;

#if BOOST_VERSION >= 103900

// (Boost) Unit Test XML Report
std::ofstream utfReportStream ("StandardAirlineITTestSuite_utfresults.xml");

struct UnitTestConfig {
  UnitTestConfig() {
    boost_utf::unit_test_log.set_stream (utfReportStream);
    boost_utf::unit_test_log.set_format (boost_utf::XML);
    boost_utf::unit_test_log.set_threshold_level (boost_utf::log_test_units);
    // boost_utf::unit_test_log.set_threshold_level (boost_utf::log_successful_tests);
  }

  ~UnitTestConfig() {
  }
};


// /////////////// Main: Unit Test Suite //////////////

// Set the UTF configuration (re-direct the output to a specific file)
BOOST_GLOBAL_FIXTURE (UnitTestConfig);

// Start the test suite
BOOST_AUTO_TEST_SUITE (master_test_suite)


BOOST_AUTO_TEST_CASE (float_comparison_test) {
  float a = 0.2f;
  a = 5*a;
  const float b = 1.0f;
```

```
  // Test the Boost way
  BOOST_CHECK_MESSAGE (a == b, "The two floats (" << a << " and " << b
                       << ") should be equal, but are not");
  BOOST_CHECK_CLOSE (a, b, 0.0001);

  // Test the Google way
  const FloatingPoint<float> lhs (a), rhs (b);
  BOOST_CHECK_MESSAGE (lhs.AlmostEquals (rhs),
                       "The two floats (" << a << " and " << b
                       << ") should be equal, but are not");
}

BOOST_AUTO_TEST_CASE (mpl_structure_test) {
  const stdair::ClassCode_T lBookingClassCodeA ("A");
  const stdair_test::BookingClass lA (lBookingClassCodeA);
  const stdair_test::Cabin lCabin (lA);

  BOOST_CHECK_EQUAL (lCabin.toString(), lBookingClassCodeA);
  BOOST_CHECK_MESSAGE (lCabin.toString() == lBookingClassCodeA,
                       "The cabin key, '" << lCabin.toString()
                       << "' is not equal to '" << lBookingClassCodeA << "'");

  // MPL
  typedef boost::mpl::vector<stdair_test::BookingClass> MPL_BookingClass;
  typedef boost::mpl::push_back<MPL_BookingClass,
                                stdair_test::Cabin>::type types;

  if (boost::is_same<stdair_test::BookingClass,
                     stdair_test::Cabin::child>::value == false) {
    BOOST_ERROR ("The two types mut be equal, but are not");
  }

  if (boost::is_same<boost::mpl::at_c<types, 1>::type,
                     stdair_test::Cabin>::value == false) {
    BOOST_ERROR ("The type must be stdair_test::Cabin, but is not");
  }
}

BOOST_AUTO_TEST_CASE (stdair_service_initialisation_test) {
  // Output log File
  const std::string lLogFilename ("StandardAirlineITTestSuite_init.log");

  // Set the log parameters
  std::ofstream logOutputFile;

  // Open and clean the log outputfile
  logOutputFile.open (lLogFilename.c_str());
  logOutputFile.clear();

  // Initialise the stdair BOM
  const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
  stdair::STDAIR_Service stdairService (lLogParams);

  // Retrieve (a reference on) the top of the persistent BOM tree
  stdair::BomRoot& lPersistentBomRoot = stdairService.getPersistentBomRoot();

  // Retrieve the BomRoot key, and compare it to the expected one
  const std::string& lBomRootKeyStr = lPersistentBomRoot.describeKey();
  const std::string lBomRootString (" -- ROOT -- ");

  // DEBUG
  STDAIR_LOG_DEBUG ("The BOM root key is '" << lBomRootKeyStr
                    << "'. It should be equal to '" << lBomRootString << "'");

  BOOST_CHECK_EQUAL (lBomRootKeyStr, lBomRootString);
  BOOST_CHECK_MESSAGE (lBomRootKeyStr == lBomRootString,
                       "The BOM root key, '" << lBomRootKeyStr
```

```
                      << "', should be equal to '" << lBomRootString
                      << "', but is not.");

  // Build a sample BOM tree
  stdairService.buildSampleBom();

  // DEBUG: Display the whole BOM tree
  const std::string& lCSVDump = stdairService.csvDisplay ();
  STDAIR_LOG_DEBUG (lCSVDump);

  // Close the Log outputFile
  logOutputFile.close();
}

BOOST_AUTO_TEST_CASE (bom_structure_instantiation_test) {
  // Step 0.0: initialisation
  // Create the root of a Bom tree (i.e., a BomRoot object)
  stdair::BomRoot& lBomRoot =
    stdair::FacBom<stdair::BomRoot>::instance().create();

  // Step 0.1: Inventory level
  // Create an Inventory (BA)
  const stdair::AirlineCode_T lBAAirlineCode ("BA");
  const stdair::InventoryKey lBAKey (lBAAirlineCode);
  myprovider::Inventory& lBAInv =
    stdair::FacBom<myprovider::Inventory>::instance().create (lBAKey);
  stdair::FacBomManager::addToList (lBomRoot, lBAInv);

  BOOST_CHECK_EQUAL (lBAInv.describeKey(), lBAAirlineCode);
  BOOST_CHECK_MESSAGE (lBAInv.describeKey() == lBAAirlineCode,
                       "The inventory key, '" << lBAInv.describeKey()
                       << "', should be equal to '" << lBAAirlineCode
                       << "', but is not");

  // Create an Inventory for AF
  const stdair::AirlineCode_T lAFAirlineCode ("AF");
  const stdair::InventoryKey lAFKey (lAFAirlineCode);
  myprovider::Inventory& lAFInv =
    stdair::FacBom<myprovider::Inventory>::instance().create (lAFKey);
  stdair::FacBomManager::addToList (lBomRoot, lAFInv);

  BOOST_CHECK_EQUAL (lAFInv.describeKey(), lAFAirlineCode);
  BOOST_CHECK_MESSAGE (lAFInv.describeKey() == lAFAirlineCode,
                       "The inventory key, '" << lAFInv.describeKey()
                       << "', should be equal to '" << lAFAirlineCode
                       << "', but is not");

  // Browse the inventories
  const myprovider::InventoryList_T& lInventoryList =
      stdair::BomManager::getList<myprovider::Inventory> (lBomRoot);
  const std::string lInventoryKeyArray[2] = {lBAAirlineCode, lAFAirlineCode};
  short idx = 0;
  for (myprovider::InventoryList_T::const_iterator itInv =
         lInventoryList.begin(); itInv != lInventoryList.end();
       ++itInv, ++idx) {
    const myprovider::Inventory* lInv_ptr = *itInv;
    BOOST_REQUIRE (lInv_ptr != NULL);

    BOOST_CHECK_EQUAL (lInventoryKeyArray[idx], lInv_ptr->describeKey());
    BOOST_CHECK_MESSAGE (lInventoryKeyArray[idx] == lInv_ptr->describeKey(),
                         "They inventory key, '" << lInventoryKeyArray[idx]
                         << "', does not match that of the Inventory object: '"
                         << lInv_ptr->describeKey() << "'");
  }
}

BOOST_AUTO_TEST_CASE (bom_structure_serialisation_test) {
```

```
// Backup (thanks to Boost.Serialisation) file
const std::string lBackupFilename = "StandardAirlineITTestSuite_serial.txt";

// Output log File
const std::string lLogFilename ("StandardAirlineITTestSuite_serial.log");

// Set the log parameters
std::ofstream logOutputFile;

// Open and clean the log outputfile
logOutputFile.open (lLogFilename.c_str());
logOutputFile.clear();

// Initialise the stdair BOM
const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
stdair::STDAIR_Service stdairService (lLogParams);

// Build a sample BOM tree
stdairService.buildSampleBom();

// Retrieve (a reference on) the top of the persistent BOM tree
stdair::BomRoot& lPersistentBomRoot = stdairService.getPersistentBomRoot();

// DEBUG: Display the whole BOM tree
const std::string& lCSVDump = stdairService.csvDisplay ();
STDAIR_LOG_DEBUG (lCSVDump);

// Clone the persistent BOM
stdairService.clonePersistentBom ();

// Retrieve the BomRoot key, and compare it to the expected one
const std::string lBAInvKeyStr ("BA");
stdair::Inventory* lBAInv_ptr =
  lPersistentBomRoot.getInventory (lBAInvKeyStr);

// DEBUG
STDAIR_LOG_DEBUG ("There should be an Inventory object corresponding to the '"
                  << lBAInvKeyStr << "' key.");

BOOST_REQUIRE_MESSAGE (lBAInv_ptr != NULL,
                       "An Inventory object should exist with the key, '"
                       << lBAInvKeyStr << "'.");

// create and open a character archive for output
std::ofstream ofs (lBackupFilename.c_str());

// save data to archive
{
  boost::archive::text_oarchive oa (ofs);
  // write class instance to archive
  oa << lPersistentBomRoot;
  // archive and stream closed when destructors are called
}

// ... some time later restore the class instance to its orginal state
stdair::BomRoot& lRestoredBomRoot =
  stdair::FacBom<stdair::BomRoot>::instance().create();
{
  // create and open an archive for input
  std::ifstream ifs (lBackupFilename.c_str());
  boost::archive::text_iarchive ia(ifs);
  // read class state from archive
  ia >> lRestoredBomRoot;
  // archive and stream closed when destructors are called
}
```

```
  // DEBUG: Display the whole restored BOM tree
  const std::string& lRestoredCSVDump =
    stdairService.csvDisplay(lRestoredBomRoot);
  STDAIR_LOG_DEBUG (lRestoredCSVDump);

  // Retrieve the BomRoot key, and compare it to the expected one
  const std::string& lBomRootKeyStr = lRestoredBomRoot.describeKey();
  const std::string lBomRootString (" -- ROOT -- ");

  // DEBUG
  STDAIR_LOG_DEBUG ("The BOM root key is '" << lBomRootKeyStr
                    << "'. It should be equal to '" << lBomRootString << "'");

  BOOST_CHECK_EQUAL (lBomRootKeyStr, lBomRootString);
  BOOST_CHECK_MESSAGE (lBomRootKeyStr == lBomRootString,
                       "The BOM root key, '" << lBomRootKeyStr
                       << "', should be equal to '" << lBomRootString
                       << "', but is not.");

  // Retrieve the Inventory
  stdair::Inventory* lRestoredBAInv_ptr =
    lRestoredBomRoot.getInventory (lBAInvKeyStr);

  // DEBUG
  STDAIR_LOG_DEBUG ("There should be an Inventory object corresponding to the '"
                    << lBAInvKeyStr << "' key in the restored BOM root.");

  BOOST_CHECK_MESSAGE (lRestoredBAInv_ptr != NULL,
                       "An Inventory object should exist with the key, '"
                       << lBAInvKeyStr << "' in the restored BOM root.");

  // Close the Log outputFile
  logOutputFile.close();
}

BOOST_AUTO_TEST_CASE (bom_structure_clone_test) {

  // Output log File
  const std::string lLogFilename ("StandardAirlineITTestSuite_clone.log");

  // Set the log parameters
  std::ofstream logOutputFile;

  // Open and clean the log outputfile
  logOutputFile.open (lLogFilename.c_str());
  logOutputFile.clear();

  // Initialise the stdair BOM
  const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
  stdair::STDAIR_Service stdairService (lLogParams);

  // Build a sample BOM tree
  stdairService.buildSampleBom();

  // Retrieve (a constant reference on) the top of the persistent BOM tree
  const stdair::BomRoot& lPersistentBomRoot =
    stdairService.getPersistentBomRoot();

  // DEBUG: Display the whole persistent BOM tree
  const std::string& lCSVDump = stdairService.csvDisplay ();
  STDAIR_LOG_DEBUG ("Display the persistent BOM tree.");
  STDAIR_LOG_DEBUG (lCSVDump);

  // Clone the persistent BOM
  stdairService.clonePersistentBom ();

  // Retrieve (a reference on) the top of the clone BOM tree
```

```
  stdair::BomRoot& lCloneBomRoot = stdairService.getBomRoot();

  // DEBUG: Display the clone BOM tree after the clone process.
  const std::string& lAfterCloneCSVDump =
    stdairService.csvDisplay(lCloneBomRoot);
  STDAIR_LOG_DEBUG ("Display the clone BOM tree after the clone process.");
  STDAIR_LOG_DEBUG (lAfterCloneCSVDump);

  // Retrieve the clone BomRoot key, and compare it to the persistent BomRoot
  // key.
  const std::string& lCloneBomRootKeyStr = lCloneBomRoot.describeKey();
  const std::string& lPersistentBomRootKeyStr =
    lPersistentBomRoot.describeKey();

  // DEBUG
  STDAIR_LOG_DEBUG ("The clone BOM root key is '" << lCloneBomRootKeyStr
                    << "'. It should be equal to '"
                    << lPersistentBomRootKeyStr << "'");

  BOOST_CHECK_EQUAL (lCloneBomRootKeyStr, lPersistentBomRootKeyStr);
  BOOST_CHECK_MESSAGE (lCloneBomRootKeyStr == lPersistentBomRootKeyStr,
                       "The clone BOM root key, '" << lCloneBomRootKeyStr
                       << "', should be equal to '" << lPersistentBomRootKeyStr
                       << "', but is not.");

  // Retrieve the BA inventory in the clone BOM root
  const std::string lBAInvKeyStr ("BA");
  stdair::Inventory* lCloneBAInv_ptr =
    lCloneBomRoot.getInventory (lBAInvKeyStr);

  // DEBUG
  STDAIR_LOG_DEBUG ("There should be an Inventory object corresponding to the '"
                    << lBAInvKeyStr << "' key in the clone BOM root.");

  BOOST_CHECK_MESSAGE (lCloneBAInv_ptr != NULL,
                       "An Inventory object should exist with the key, '"
                       << lBAInvKeyStr << "' in the clone BOM root.");

  // Close the Log outputFile
  logOutputFile.close();
}

// End the test suite
BOOST_AUTO_TEST_SUITE_END()

#else // BOOST_VERSION >= 103900
boost_utf::test_suite* init_unit_test_suite (int, char* []) {
  boost_utf::test_suite* test = BOOST_TEST_SUITE ("Unit test example 1");
  return test;
}
#endif // BOOST_VERSION >= 103900

/*!
```