

StdAir

1.00.3

Generated by Doxygen 1.8.9.1

Sun Jun 28 2015 18:08:36

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

#### 1.3 StdAir Development

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

#### 1.4 External Libraries

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

## 1.5 Support StdAir

## 1.6 About StdAir

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

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

The StdAir library originates from the department of Operational Research and Innovation at [Amadeus](#), Sophia Antipolis, France. StdAir is released under the terms of the [GNU Lesser General Public License](#) (L↔GPLv2.1) 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 BomAbstract

Abstract part of the Business Object Model (BOM)

### Author

Anh Quan Nguyen [quannaus@users.sourceforge.net](mailto:quannaus@users.sourceforge.net)

### Date

20/01/2010

## 3 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;
}
}

```

## 4 KeyAbstract

Part of the Business Object Model (BOM) handling (hash-like )keys

Author

Anh Quan Nguyen [quannaus@users.sourceforge.net](mailto:quannaus@users.sourceforge.net)

Date

20/01/2010

## 5 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=0009&from=LHR&to=SYD&depD
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=0009&from=LHR&to=BKK&depD
const AirportCode_T lBKK ("BKK");
const Duration_T l11540 (11, 54, 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 (l11540);
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=0009&from=BKK&to=SYD&depD

```

```

const Duration_T l1705 (17, 5, 0);
const Duration_T l0905 (9, 5, 0);
lSegmentDateKey = SegmentDateKey (lBKK, lSYD);

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

// Add the routing leg key to the BKK-SYD segment.
lBKKSYSYDSegment.addLegKey (lBABKKRoutingLegStr);

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

// Create a SegmentCabin (Y) of the Segment LHR-SYD;
SegmentCabin& lLHRSYSYDSegmentYCabin =
    FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);

```

```

FacBomManager::addToListAndMap (1LHRSYDSegment, 1LHRSYDSegmentYCabin);
FacBomManager::linkWithParent (1LHRSYDSegment, 1LHRSYDSegmentYCabin);

// Display the segment-cabin
// STDAIR_LOG_DEBUG ("SegmentCabin: " << 1LHRSYDSegmentYCabin.toString());

// Step 0.6: leg-cabin level
// Create a LegCabin (Y) for the Leg LHR-BKK on BA's Inventory
LegCabinKey 1YLegCabinKey (1Y);

LegCabin& 1LHRLegYCabin =
    FacBom<LegCabin>::instance().create (1YLegCabinKey);
FacBomManager::addToListAndMap (1LHRLeg, 1LHRLegYCabin);
FacBomManager::linkWithParent (1LHRLeg, 1LHRLegYCabin);

// Display the leg-cabin
// STDAIR_LOG_DEBUG ("LegCabin: " << 1LHRLegYCabin.toString());

// Create a LegCabin (Y) for the Leg BKK-SYD
LegCabin& 1BKKLegYCabin =
    FacBom<LegCabin>::instance().create (1YLegCabinKey);
FacBomManager::addToListAndMap (1BKKLeg, 1BKKLegYCabin);
FacBomManager::linkWithParent (1BKKLeg, 1BKKLegYCabin);
// Display the leg-cabin
// STDAIR_LOG_DEBUG ("LegCabin: " << 1BKKLegYCabin.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 1l ("EcoSaver");
FareFamilyKey 1lFareFamilyKey (1l);

FareFamily& 1LHRBKKSegmentYCabin1Family =
    FacBom<FareFamily>::instance().create (1lFareFamilyKey);
FacBomManager::addToListAndMap (1LHRBKKSegmentYCabin,
                                1LHRBKKSegmentYCabin1Family);
FacBomManager::linkWithParent (1LHRBKKSegmentYCabin,
                                1LHRBKKSegmentYCabin1Family);

// Display the booking class
// STDAIR_LOG_DEBUG ("FareFamily: "
//                    << 1LHRBKKSegmentYCabin1Family.toString());

// Create a FareFamily (1) for the Segment BKK-SYD, cabin Y on BA's Inv
FareFamily& 1BKKSYSYDSegmentYCabin1Family =
    FacBom<FareFamily>::instance().create (1lFareFamilyKey);
FacBomManager::addToListAndMap (1BKKSYSYDSegmentYCabin,
                                1BKKSYSYDSegmentYCabin1Family);
FacBomManager::linkWithParent (1BKKSYSYDSegmentYCabin,
                                1BKKSYSYDSegmentYCabin1Family);

// Display the booking class
// STDAIR_LOG_DEBUG ("FareFamily: "
//                    << 1LHRBKKSegmentYCabin1Family.toString());

// Create a FareFamily (1) for the Segment LHR-SYD, cabin Y on BA's Inv
FareFamily& 1LHRSYDSegmentYCabin1Family =
    FacBom<FareFamily>::instance().create (1lFareFamilyKey);
FacBomManager::addToListAndMap (1LHRSYDSegmentYCabin,
                                1LHRSYDSegmentYCabin1Family);
FacBomManager::linkWithParent (1LHRSYDSegmentYCabin,
                                1LHRSYDSegmentYCabin1Family);

// Display the booking class
// STDAIR_LOG_DEBUG ("FareFamily: "
//                    << 1LHRBKKSegmentYCabin1Family.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 1Q ("Q");
BookingClassKey 1QBookingClassKey (1Q);

BookingClass& 1LHRBKKSegmentYCabin1FamilyQClass =
    FacBom<BookingClass>::instance().create (1QBookingClassKey);
FacBomManager::addToListAndMap (1LHRBKKSegmentYCabin1Family,
                                1LHRBKKSegmentYCabin1FamilyQClass);
FacBomManager::linkWithParent (1LHRBKKSegmentYCabin1Family,
                                1LHRBKKSegmentYCabin1FamilyQClass);

FacBomManager::addToListAndMap (1LHRBKKSegmentYCabin,
                                1LHRBKKSegmentYCabin1FamilyQClass);
FacBomManager::addToListAndMap (1LHRBKKSegment,
                                1LHRBKKSegmentYCabin1FamilyQClass);

// Display the booking class
// STDAIR_LOG_DEBUG ("BookingClass: "

```

```

//                                     << 1LHRBKKSegmentYCabin1FamilyQClass.toString());

// Create a BookingClass (Q) for the Segment BKK-SYD, cabin Y,
// fare family 1 on BA's Inv
BookingClass& 1BKKSYSYDSegmentYCabin1FamilyQClass =
    FacBom<BookingClass>::instance().create (1QBookingClassKey);
FacBomManager::addToListAndMap (1BKKSYSYDSegmentYCabin1Family,
                                1BKKSYSYDSegmentYCabin1FamilyQClass);
FacBomManager::linkWithParent (1BKKSYSYDSegmentYCabin1Family,
                                1BKKSYSYDSegmentYCabin1FamilyQClass);

FacBomManager::addToListAndMap (1BKKSYSYDSegmentYCabin,
                                1BKKSYSYDSegmentYCabin1FamilyQClass);
FacBomManager::addToListAndMap (1BKKSYSYDSegment,
                                1BKKSYSYDSegmentYCabin1FamilyQClass);

// Display the booking class
// STDAIR_LOG_DEBUG ("BookingClass: "
//                                     << 1LHRBKKSegmentYCabin1FamilyQClass.toString());

// Create a BookingClass (Q) for the Segment LHR-SYD, cabin Y,
// fare family 1 on BA's Inv
BookingClass& 1LHRSYSYDSegmentYCabin1FamilyQClass =
    FacBom<BookingClass>::instance().create (1QBookingClassKey);
FacBomManager::addToListAndMap (1LHRSYSYDSegmentYCabin1Family,
                                1LHRSYSYDSegmentYCabin1FamilyQClass);
FacBomManager::linkWithParent (1LHRSYSYDSegmentYCabin1Family,
                                1LHRSYSYDSegmentYCabin1FamilyQClass);

FacBomManager::addToListAndMap (1LHRSYSYDSegmentYCabin,
                                1LHRSYSYDSegmentYCabin1FamilyQClass);
FacBomManager::addToListAndMap (1LHRSYSYDSegment,
                                1LHRSYSYDSegmentYCabin1FamilyQClass);

// Display the booking class
// STDAIR_LOG_DEBUG ("BookingClass: "
//                                     << 1LHRBKKSegmentYCabin1FamilyQClass.toString());

// ===== AF =====
// Step 0.2: Flight-date level
// Create a FlightDate (AF084/20-MAR-2011) for AF's Inventory
1FlightNumber = 84;
1Date = Date_T (2011, 3, 20);
1FlightDateKey = FlightDateKey (1FlightNumber, 1Date);

FlightDate& 1AF084_20110320_FD =
    FacBom<FlightDate>::instance().create (1FlightDateKey);
FacBomManager::addToListAndMap (1AFInv, 1AF084_20110320_FD);
FacBomManager::linkWithParent (1AFInv, 1AF084_20110320_FD);

// Display the flight-date
// STDAIR_LOG_DEBUG ("FlightDate: " << 1AF084_20110320_FD.toString());

// Step 0.3: Segment-date level
// Create a SegmentDate (CDG-SFO) for AF's Inventory
const AirportCode_T 1CDG ("CDG");
const AirportCode_T 1SFO ("SFO");
const Duration_T 11040 (10, 40, 0);
const Duration_T 11250 (12, 50, 0);
const Duration_T 11110 (11, 10, 0);
1SegmentDateKey = SegmentDateKey (1CDG, 1SFO);

SegmentDate& 1CDGSFOSegment =
    FacBom<SegmentDate>::instance().create (1SegmentDateKey);
FacBomManager::addToListAndMap (1AF084_20110320_FD, 1CDGSFOSegment);
FacBomManager::linkWithParent (1AF084_20110320_FD, 1CDGSFOSegment);

// Add the routing leg key to the CDG-SFO segment.
const std::string 1AFCDGRoutingLegStr = "AF;84;2011-Mar-20;CDG";
1CDGSFOSegment.addLegKey (1AFCDGRoutingLegStr);

// Display the segment-date
// STDAIR_LOG_DEBUG ("SegmentDate: " << 1CDGSFOSegment.toString());

// Fill the SegmentDate content
1CDGSFOSegment.setBoardingDate (1Date);
1CDGSFOSegment.setOffDate (1Date);
1CDGSFOSegment.setBoardingTime (11040);
1CDGSFOSegment.setOffTime (11250);
1CDGSFOSegment.setElapsedTime (11110);

// Step 0.4: Leg-date level
// Create a LegDate (CDG) for AF's Inventory
1LegDateKey = LegDateKey (1CDG);

LegDate& 1CDGLeg = 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& lCDGSFOSegmentYCabinlFamily =
    FacBom<FareFamily>::instance().create (l1FareFamilyKey);
FacBomManager::addToListAndMap (lCDGSFOSegmentYCabin,
                                lCDGSFOSegmentYCabinlFamily);
FacBomManager::linkWithParent (lCDGSFOSegmentYCabin,
                                lCDGSFOSegmentYCabinlFamily);

// Display the fare family
// STDAIR_LOG_DEBUG ("fareFamily: "
//
//
// << lCDGSFOSegmentYCabinlFamily.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& lCDGSFOSegmentYCabinlFamilyQClass =
    FacBom<BookingClass>::instance().create (lQBookingClassKey);
FacBomManager::addToListAndMap (lCDGSFOSegmentYCabinlFamily,
                                lCDGSFOSegmentYCabinlFamilyQClass);
FacBomManager::linkWithParent (lCDGSFOSegmentYCabinlFamily,
                                lCDGSFOSegmentYCabinlFamilyQClass);

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

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

/*=====
=====
=====*/
// Schedule:
// BA:
// Step 1: flight period level
// Create a flight period for BA9:
const DoWStruct lDoWSrtuct ("111111");
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);

```

```

1SINBKKSegmentYCabin2FamilyBClass.setYield(800);

// Display the booking class
// STDAIR_LOG_DEBUG ("BookingClass: "
//                  << 1SINBKKSegmentYCabin2FamilyBClass.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& 1SINBKKSegmentYCabin1FamilyMClass =
    FacBom<BookingClass>::instance().create (lMBookingClassKey);
FacBomManager::addToListAndMap (1SINBKKSegmentYCabin1Family,
                                1SINBKKSegmentYCabin1FamilyMClass);
FacBomManager::linkWithParent (1SINBKKSegmentYCabin1Family,
                                1SINBKKSegmentYCabin1FamilyMClass);

FacBomManager::addToListAndMap (1SINBKKSegmentYCabin,
                                1SINBKKSegmentYCabin1FamilyMClass);
FacBomManager::addToListAndMap (1SINBKKSegment,
                                1SINBKKSegmentYCabin1FamilyMClass);
1SINBKKSegmentYCabin1FamilyMClass.setYield(900);

// Display the booking class
// STDAIR_LOG_DEBUG ("BookingClass: "
//                  << 1SINBKKSegmentYCabin1FamilyMClass.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& 1SINBKKSegmentYCabin1FamilyQClass =
    FacBom<BookingClass>::instance().create (lQBookingClassKey);
FacBomManager::addToListAndMap (1SINBKKSegmentYCabin1Family,
                                1SINBKKSegmentYCabin1FamilyQClass);
FacBomManager::linkWithParent (1SINBKKSegmentYCabin1Family,
                                1SINBKKSegmentYCabin1FamilyQClass);

FacBomManager::addToListAndMap (1SINBKKSegmentYCabin,
                                1SINBKKSegmentYCabin1FamilyQClass);
FacBomManager::addToListAndMap (1SINBKKSegment,
                                1SINBKKSegmentYCabin1FamilyQClass);
1SINBKKSegmentYCabin1FamilyQClass.setYield(600);

// Display the booking class
// STDAIR_LOG_DEBUG ("BookingClass: "
//                  << 1SINBKKSegmentYCabin1FamilyQClass.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");
1SINBKKSegmentPeriod.addCabinBookingClassList (lY, lYBMQ);
1SINBKKSegmentPeriod.setBoardingTime (10635);
1SINBKKSegmentPeriod.setOffTime (10800);
1SINBKKSegmentPeriod.setElapsedTime (10225);

```

```

/*=====
=====*/
// 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::ostream oStr;
    oStr << DEFAULT_AIRLINE_CODE << ",";
    oStr << DEFAULT_FLIGHT_NUMBER << ",";
    oStr << DEFAULT_DEPARTURE_DATE << ",";
    oStr << 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 lAv11 (8);
    bool hasInsertOfQBeenSuccessful = lClassAvailabilityMap1.
        insert (ClassAvailabilityMap_T::value_type (lClassPathQ, lAv11)).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 10820 (8, 20, 0);
const Duration_T 11100 (11, 0, 0);
const Duration_T 10340 (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 (10820);
lSINBKKSegment.setOffTime (11100);
lSINBKKSegment.setElapsedTime (10340);

// Create a second (mkt) SegmentDate (BKK-HKG) for SQ's Inventory
const AirportCode_T lHKG ("HKG");
const Duration_T 11200 (12, 0, 0);
const Duration_T 11540 (15, 40, 0);
const Duration_T 10240 (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 (11200);
lMktBKKHKGSegment.setOffTime (11540);
lMktBKKHKGSegment.setElapsedTime (10240);

// 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 (10820);
lSINLeg.setOffTime (11100);
lSINLeg.setElapsedTime (10340);

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

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

// 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& lSINBKKSegmentYCabinlFamilyYClass =
    FacBom<BookingClass>::instance().create (lYBookingClassKey);
FacBomManager::addToListAndMap (lSINBKKSegmentYCabinlFamily,
                                lSINBKKSegmentYCabinlFamilyYClass);
FacBomManager::linkWithParent (lSINBKKSegmentYCabinlFamily,
                                lSINBKKSegmentYCabinlFamilyYClass);

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

lSINBKKSegmentYCabinlFamilyYClass.setYield(700);

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

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

lMktBKKHKGSegmentYCabinlFamilyYClass.setYield(700);

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

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

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

lSINBKKSegmentYCabinlFamilyMClass.setYield(500);

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

```

```

FacBomManager::addToListAndMap (lMktBKKHKGSegmentYCabinlFamily,
                                lMktBKKHKGSegmentYCabinlFamilyMClass);
FacBomManager::linkWithParent (lMktBKKHKGSegmentYCabinlFamily,
                                lMktBKKHKGSegmentYCabinlFamilyMClass);

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

lMktBKKHKGSegmentYCabinlFamilyMClass.setYield(500);

/* ===== */

// Step 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.
// First leg: cabin Y, class M.
CabinClassPair_T lCC_YM1 (lY,lM);
// Second leg: cabin Y, class M too.
CabinClassPair_T lCC_YM2 (lY,lM);
CabinClassPairList_T lCabinClassPairList;
lCabinClassPairList.push_back (lCC_YM1);
lCabinClassPairList.push_back (lCC_YM2);
const MeanStdDevPair_T lMean20StdDev2 (20.0, 2.0);
const Yield_T lYield850 = 850.0;
const YieldDemandPair_T lYield850Mean20StdDev2 (lYield850, lMean20StdDev2);
lSQ_SINHKG_OnDDate.setDemandInformation (lCabinClassPairList, 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& lBKKHKGSegmentYCabinlFamily =
    FacBom<FareFamily>::instance().create (l1FareFamilyKey);
FacBomManager::addToListAndMap (lBKKHKGSegmentYCabin,
    lBKKHKGSegmentYCabinlFamily);
FacBomManager::linkWithParent (lBKKHKGSegmentYCabin,
    lBKKHKGSegmentYCabinlFamily);

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

// 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& lBKKHKGSegmentYCabinlFamilyYClass =
    FacBom<BookingClass>::instance().create (lYBookingClassKey);
FacBomManager::addToListAndMap (lBKKHKGSegmentYCabinlFamily,
    lBKKHKGSegmentYCabinlFamilyYClass);
FacBomManager::linkWithParent (lBKKHKGSegmentYCabinlFamily,
    lBKKHKGSegmentYCabinlFamilyYClass);

```

```

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

lBKKHKGSegmentYCabinlFamilyYClass.setYield(700);

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

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

lMktSINBKKSegmentYCabinlFamilyYClass.setYield(700);

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

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

lBKKHKGSegmentYCabinlFamilyMClass.setYield(500);

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

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

lMktSINBKKSegmentYCabinlFamilyMClass.setYield(500);

/* ===== */

// Step 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_SINBKK_OnDDate.setDemandInformation (lCabinClassPairList,
                                         lYield1200Mean10StdDev1);

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

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

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

// Demand info is not added for purely marketed O&Ds
// Add demand info
// lCabinClassPairList.clear();
// lCabinClassPairList.push_back(lCC_YM1);
// lMktCX_SINBKK_OnDDate.setDemandInformation (lCabinClassPairList, 500.0, 20.0, 1.0);
*/

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

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

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

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

// Add demand info
lCabinClassPairList.clear();
lCabinClassPairList.push_back(lCC_YM2);
lCX_BKKHKG_OnDDate.setDemandInformation (lCabinClassPairList,
                                         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 DoWSrtuct lDoWSrtuct ("111111");
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 (lSql1FlightPeriod, 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& lBKKHKGTimePeriod =
    FacBom<TimePeriod>::instance().create (lFareTimePeriodKey);
FacBomManager::addToListAndMap (lBKKPosChannel, lBKKHKGTimePeriod);
FacBomManager::linkWithParent (lBKKPosChannel, lBKKHKGTimePeriod);

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

// Create the FareFeaturesKey and link it to the TimePeriod object.

```



```

FareFeatures& 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 hasAirportPairList =
        BomManager::hasList<AirportPair> (iBomRoot);
    if (hasAirportPairList == 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 hasFlightDateList = BomManager::hasList<FlightDate> (iInventory);
    if (hasFlightDateList == 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 =
                cloneFareFeatures<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;
}
}

```

## 6 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;
}
}

```

## 7 People

### 7.1 Project Admins (and Developers)

- Denis Arnaud [denis\\_arnaud@users.sourceforge.net](mailto:denis_arnaud@users.sourceforge.net) (N)
- Anh Quan Nguyen [quannaus@users.sourceforge.net](mailto:quannaus@users.sourceforge.net) (N)
- Gabrielle Sabatier [gsabatier@users.sourceforge.net](mailto:gsabatier@users.sourceforge.net) (N)

### 7.2 Retired Developers

- Mehdi Ayouni [mehdi.ayouni@gmail.com](mailto:mehdi.ayouni@gmail.com)

- Son Nguyen Kim [snguyenkim@users.sourceforge.net](mailto:snguyenkim@users.sourceforge.net) (N)

### 7.3 Contributors

- Emmanuel Bastien [ebastien@users.sourceforge.net](mailto:ebastien@users.sourceforge.net) (N)

### 7.4 Distribution Maintainers

- **Fedora/RedHat**: Denis Arnaud [denis\\_arnaud@users.sourceforge.net](mailto:denis_arnaud@users.sourceforge.net) (N)
- **Debian**: Emmanuel Bastien [ebastien@users.sourceforge.net](mailto:ebastien@users.sourceforge.net) (N)

#### Note

(N) - **Amadeus** employees.

## 8 Coding Rules

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

### 8.1 Default Naming Rules for Variables

Variables names follow Java naming conventions. Examples:

- `lNumberOfPassengers`
- `lSeatAvailability`

### 8.2 Default Naming Rules for Functions

Function names follow Java naming conventions. Example:

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

### 8.3 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`

### 8.4 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`

## 8.5 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.

## 9 Copyright and License

### 9.1 GNU LESSER GENERAL PUBLIC LICENSE

#### 9.1.1 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.]

### 9.2 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.

### 9.3 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.

1. 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.

1. 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.

1. 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.

1. 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.

1. 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.

1. 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.

2. 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.
3. 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.
4. 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.
5. 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.

1. 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.

2. 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.

1. 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.

### 9.3.1 NO WARRANTY

1. 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.
2. 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.

### 9.3.2 END OF TERMS AND CONDITIONS

## 9.4 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 Documentation Rules

### 10.1 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
};

```

## 10.2 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
 *
 * -----
 */

```

## 10.3 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);

```

## 11 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.

### 11.1 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

### 11.2 Architecture of the StdAir library

- Separate structure and content classes
- `Boost.Fusion`

## 12 Make a Difference

**Do not ask what StdAir can do for you. Ask what you can do for StdAir.**

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

- Start using StdAir
- Tell your friends about StdAir and help them to get started using it
- If you find a bug, report it to us (on the [dedicated Sourceforge's Trac Web site](#)). Without your help we can never hope to produce a bug free code.
- Help us to improve the documentation by providing information about documentation bugs
- Answer support requests in the StdAir [discussion forums](#) on SourceForge. If you know the answer to a question, help others to overcome their StdAir problems.
- Help us to improve our algorithms. If you know of a better way (e.g., that is faster or requires less memory) to implement some of our algorithms, then let us know.
- Help to port StdAir to new platforms. If you manage to compile StdAir on a new platform, then tell how you did it.
- Send your code. If you have a good StdAir compatible code, which you can release under the LGPL, and you think it should be included in StdAir, then send it to the community.
- Become an StdAir developer. Send us (see the [People](#) page) an e-mail and tell what you can do for StdAir.

## 13 Make a new release

### 13.1 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.

### 13.2 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
```

### 13.3 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
```

### 13.4 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
```

### 13.5 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
```



### 13.6 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`.

### 13.7 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-*
```

### 13.8 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](#).

### 13.9 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.

### 13.10 Upload the files to SourceForge

Upload the distribution and documentation packages to the SourceForge server. Check [SourceForge help page on uploading software](#).

## 13.11 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](#).

## 13.12 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](#).

## 13.13 Send an email on the announcement mailing-list

Finally, you should send an announcement to [stdair-announce@lists.sourceforge.net](mailto:stdair-announce@lists.sourceforge.net) (see <https://lists.sourceforge.net/lists/listinfo/stdair-announce> for the archives)

# 14 Installation

## 14.1 Table of Contents

- [Fedora/RedHat Linux distributions](#)
- [StdAir Requirements](#)
- [Basic Installation](#)
- [Compilers and Options](#)
- [Compiling For Multiple Architectures](#)
- [Installation Names](#)
- [Optional Features](#)
- [Particular systems](#)
- [Specifying the System Type](#)
- [Sharing Defaults](#)
- [Defining Variables](#)
- [‘cmake’ Invocation](#)

## 14.2 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](#).

## 14.3 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'`)
- [SOCHI](#) - 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.

## 14.4 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.

## 14.5 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.

## 14.6 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.

## 14.7 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.

## 14.8 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'`.

## 14.9 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
```

## 14.10 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'.

## 14.11 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.

## 14.12 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
```

## 14.13 'cmake' Invocation

'cmake' recognizes the following options to control how it operates.

- '-help', '-h' print a summary of all of the options to 'configure', and exit.
- '-help=short', '-help=recursive' print a summary of the options unique to this package's 'configure', and exit. The 'short' variant lists options used only in the top level, while the 'recursive' variant lists options also present in any nested packages.
- '-version', '-V' print the version of Autoconf used to generate the 'configure' script, and exit.
- '-cache-file=FILE' enable the cache: use and save the results of the tests in FILE, traditionally 'config.cache'. FILE defaults to '/dev/null' to disable caching.
- '-config-cache', '-C' alias for '-cache-file=config.cache'.
- '-quiet', '-silent', '-q' do not print messages saying which checks are being made. To suppress all normal output, redirect it to '/dev/null' (any error messages will still be shown).
- '-srcdir=DIR' look for the package's source code in directory DIR. Usually 'configure' can determine that directory automatically.
- '-prefix=DIR' use DIR as the installation prefix.

See also

[Installation Names](#) for more details, including other options available for fine-tuning the installation locations.

- '-no-create', '-n' run the configure checks, but stop before creating any output files.

'cmake' also accepts some other, not widely useful, options. Run 'cmake -help' for more details.

The 'cmake' script produces an output like this:

```
cmake -DCMAKE_INSTALL_PREFIX=/home/user/dev/deliveries/stdair-0.50.0 \
-DLIB_SUFFIX=64 -DCMAKE_BUILD_TYPE:String=Debug -DINSTALL_DOC:BOOL=ON ..
-- The C compiler identification is GNU
-- The CXX compiler identification is GNU
-- Check for working C compiler: /usr/lib64/ccache/gcc
-- Check for working C compiler: /usr/lib64/ccache/gcc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: /usr/lib64/ccache/c++
-- Check for working CXX compiler: /usr/lib64/ccache/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Requires Git without specifying any version
-- Current Git revision name: 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 'stdairuicllib' to CXX
-- Test 'StdAirTest' to be built with 'MPBomRoot.cpp;MPInventory.cpp;StandardAirlineITTestSuite.cpp'
--
-- =====
-- -----
-- ---      Project Information      ---
-- -----
-- PROJECT_NAME ..... : stdair
-- PACKAGE_PRETTY_NAME ..... : StdAir
-- PACKAGE ..... : stdair
-- PACKAGE_NAME ..... : STDAIR
-- PACKAGE_VERSION ..... : 0.50.0
-- GENERIC_LIB_VERSION ..... : 0.50.0
-- GENERIC_LIB_SOVERSION ..... : 99.99
--
-- -----
-- ---      Build Configuration      ---
-- -----
-- Modules to build ..... : stdair
-- Libraries to build/install ..... : stdairlib;stdairuicllib
-- Binaries to build/install ..... : stdair
-- Modules to test ..... : stdair
-- Binaries to test ..... : StdAirTesttst
--
-- * Module ..... : stdair
--   + Layers to build ..... : .;basic;bom;factory;dbadaptor;command;service
--   + Dependencies on other layers :
--   + Libraries to build/install . : stdairlib;stdairuicllib
--   + Executables to build/install : stdair
--   + Tests to perform ..... : StdAirTesttst
--
-- BUILD_SHARED_LIBS ..... : ON
-- CMAKE_BUILD_TYPE ..... : Debug
-- * CMAKE_C_FLAGS ..... :
-- * CMAKE_CXX_FLAGS ..... : -O2 -g -pipe -Wall -Wp,-D_FORTIFY_SOURCE=2 -fexceptions -fstack-protectio
-- * 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_test_f
--   - Boost required libraries ... : optimized;/usr/lib64/libboost_iostreams-mt.so;debug;/usr/lib64/libboost_
--
-- * MySQL:
--   - MYSQL_VERSION ..... : 5.5.14
--   - MYSQL_INCLUDE_DIR ..... : /usr/include/mysql
--   - MYSQL_LIBRARIES ..... : /usr/lib64/mysql/libmysqlclient.so
--
-- * SOCI:
--   - SOCI_VERSION ..... : 3.0.0
--   - SOCI_INCLUDE_DIR ..... : /usr/include/soci
--   - SOCI_MYSQL_INCLUDE_DIR ..... : /usr/include/soci
--   - SOCI_LIBRARIES ..... : /usr/lib64/libsoci_core.so
--   - SOCI_MYSQL_LIBRARIES ..... : /usr/lib64/libsoci_mysql.so
--
-- Change a value with: cmake -D<Variable>=<Value>
-- =====
--
-- Configuring done
-- Generating done
-- Build files have been written to: /home/user/dev/sim/stdair/stdairgithub/build

```

It is recommended that you check if your library has been compiled and linked properly and works as expected. To do so, you should execute the testing process 'make check'. As a result, you should obtain a similar report:

```

[ 0%] Built target hdr_cfg_stdair
[ 97%] Built target stdairlib
[100%] Built target StdAirTesttst
Scanning dependencies of target check_stdairtst
Test project /home/user/dev/sim/stdair/stdairgithub/build/test/stdair
  Start 1: StdAirTesttst
1/1 Test #1: StdAirTesttst ..... Passed    0.02 sec

100% tests passed, 0 tests failed out of 1

Total Test time (real) = 0.27 sec

```

```
[100%] Built target check_stdairtst
Scanning dependencies of target check
[100%] Built target check
```

Check if all the executed tests PASSED. If not, please contact us by filling a [bug-report](#).

Finally, you should install the compiled and linked library, include files and (optionally) HTML and PDF documentation by typing:

```
make install
```

Depending on the PREFIX settings during configuration, you might need the root (administrator) access to perform this step.

Eventually, you might invoke the following command

```
make clean
```

to remove all files created during compilation process, or even

```
cd ~/dev/sim/stdairgit
rm -rf build && mkdir build
cd build
```

to remove everything.

## 15 Linking with StdAir

### 15.1 Table of Contents

- [Introduction](#)
- [Using the pkg-config command](#)
- [Using the stdair-config script](#)
- [M4 macro for the GNU Autotools](#)
- [Using StdAir with dynamic linking](#)

### 15.2 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.

### 15.3 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.

## 15.4 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
```

## 15.5 M4 macro for the GNU Autotools

A M4 macro file is delivered with StdAir, namely `'stdair.m4'`, which can be found in, e.g., `'/usr/share/aclocal'`. When used by a `'configure'` script, thanks to the `'AM_PATH_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'`)

## 15.6 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
```

# 16 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.

## 16.1 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'`.

## 16.2 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.

## 16.3 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.

# 17 Users Guide

## 17.1 Table of Contents

- [Introduction](#)
- [Get Started](#)
  - [Get the StdAir library](#)
  - [Build the StdAir project](#)
  - [Build and Run the Tests](#)
  - [Install the StdAir Project \(Binaries, Documentation\)](#)
- [Exploring the Predefined BOM Tree](#)
  - [Airline Distribution BOM Tree](#)
  - [Airline Network BOM Tree](#)
  - [Airline Inventory BOM Tree](#)
- [Extending the BOM Tree](#)

## 17.2 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`.

## 17.3 Get Started

### 17.3.1 Get the StdAir library

### 17.3.2 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`

### 17.3.3 Build and Run the Tests

### 17.3.4 Install the StdAir Project (Binaries, Documentation)

## 17.4 Exploring the Predefined BOM Tree

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

### 17.4.1 Airline Distribution BOM Tree

- `stdair::TravelSolutionStruct`

### 17.4.2 Airline Network BOM Tree

- `stdair::FlightPeriod`

### 17.4.3 Airline Inventory BOM Tree

- `stdair::Inventory`
- `stdair::FlightDate`

#### 17.4.3.1 Airline Inventory Marketing BOM Tree

- `stdair::SegmentDate`
- `stdair::SegmentCabin`
- `stdair::FareFamily`
- `stdair::BookingClass`

#### 17.4.3.2 Airline Inventory Operating BOM Tree

- `stdair::LegDate`
- `stdair::LegCabin`
- `stdair::Bucket`

## 17.5 Extending the BOM Tree

# 18 Supported Systems

## 18.1 Table of Contents

- [Introduction](#)

- [StdAir 3.10.x](#)
  - [Linux Systems](#)
    - \* [Fedora Core 4 with ATLAS](#)
    - \* [Gentoo Linux with ACML](#)
    - \* [Gentoo Linux with ATLAS](#)
    - \* [Gentoo Linux with MKL](#)
    - \* [Gentoo Linux with NetLib's BLAS and LAPACK](#)
    - \* [Red Hat Enterprise Linux with StdAir External](#)
    - \* [SUSE Linux 10.0 with NetLib's BLAS and LAPACK](#)
    - \* [SUSE Linux 10.0 with MKL](#)
  - [Windows Systems](#)
    - \* [Microsoft Windows XP with Cygwin](#)
    - \* [Microsoft Windows XP with Cygwin and ATLAS](#)
    - \* [Microsoft Windows XP with Cygwin and ACML](#)
    - \* [Microsoft Windows XP with MinGW, MSYS and ACML](#)
    - \* [Microsoft Windows XP with MinGW, MSYS and StdAir External](#)
    - \* [Microsoft Windows XP with MS Visual C++ and Intel MKL](#)
  - [Unix Systems](#)
    - \* [SunOS 5.9 with StdAir External](#)
- [StdAir 3.9.1](#)
- [StdAir 3.9.0](#)
- [StdAir 3.8.1](#)

## 18.2 Introduction

This page is intended to provide a list of StdAir supported systems, i.e. the systems on which configuration, installation and testing process of the StdAir library has been successful. Results are grouped based on minor release number. Therefore, only the latest tests for bug-fix releases are included. Besides, the information on this page is divided into sections dependent on the operating system.

Where necessary, some extra information is given for each tested configuration, e.g. external libraries installed, configuration commands used, etc.

If you manage to compile, install and test the StdAir library on a system not mentioned below, please let us know, so we could update this database.

## 18.3 StdAir 3.10.x

### 18.3.1 Linux Systems

#### 18.3.1.1 Fedora Core 4 with ATLAS

- **Platform:** Intel Pentium 4
- **Operating System:** Fedora Core 4 (x86)
- **Compiler:** g++ (GCC) 4.0.2 20051125
- **StdAir release:** 3.10.0
- **External Libraries:** From FC4 distribution:
  - `fftw3.i386-3.0.1-3`

```

- fftw3-devel.i386-3.0.1-3
- atlas-sse2.i386-3.6.0-8.fc4
- atlas-sse2-devel.i386-3.6.0-8.fc4
- blas.i386-3.0-35.fc4
- lapack.i386-3.0-35.fc4

```

- **Tests Status:** All tests PASSED

- **Comments:** StdAir configured with:

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

- **Date:** March 7, 2006

- **Tester:** Tony Ottosson

#### 18.3.1.2 Gentoo Linux with ACML

- **Platform:** AMD Sempron 3000+
- **Operating System:** Gentoo Linux 2006.0 (x86 arch)
- **Compiler(s):** g++ (GCC) 3.4.5
- **StdAir release:** 3.10.1
- **External Libraries:** Compiled and installed from portage tree:

```
- sci-libs/acml-3.0.0
```

- **Tests Status:** All tests PASSED

- **Comments:** BLAS and LAPACK libs set by using the following system commands:

```
% eselect blas set ACML
% eselect lapack set ACML
```

StdAir configured with:

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

- **Date:** March 31, 2006

- **Tester:** Adam Piatyszek (ediap)

#### 18.3.1.3 Gentoo Linux with ATLAS

- **Platform:** Intel Pentium M Centrino
- **Operating System:** Gentoo Linux 2006.0 (x86)
- **Compiler:** g++ (GCC) 3.4.5
- **StdAir release:** 3.10.1
- **External Libraries:** Compiled and installed from portage tree:

```

- sci-libs/fftw-3.1
- sci-libs/blas-atlas-3.6.0-r1
- sci-libs/lapack-atlas-3.6.0

```

- **Tests Status:** All tests PASSED



- **Comments:** BLAS and LAPACK libs set by using the following system commands:

```
% eselect blas set ATLAS
% eselect lapack set ATLAS
```

StdAir configured with:

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

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

#### 18.3.1.4 Gentoo Linux with MKL

- **Platform:** Intel Pentium M Centrino
- **Operating System:** Gentoo Linux 2006.0 (x86 arch)
- **Compiler:** g++ (GCC) 3.4.5
- **StdAir release:** 3.10.0
- **External Libraries:** Intel Math Kernel Library (MKL) 8.0.1 installed manually in the following directory↵  
: /opt/intel/mkl/8.0.1
- **Tests Status:** All tests PASSED
- **Comments:** StdAir configured using the following commands:

```
% export LDFLAGS="-L/opt/intel/mkl/8.0.1/lib/32"
% export CPPFLAGS="-I/opt/intel/mkl/8.0.1/include"
% ./configure
```

- **Date:** February 28, 2006
- **Tester:** Adam Piatyszek (ediap)

#### 18.3.1.5 Gentoo Linux with NetLib's BLAS and LAPACK

- **Platform:** Intel Pentium M Centrino
- **Operating System:** Gentoo Linux 2006.0 (x86)
- **Compiler:** g++ (GCC) 3.4.5
- **StdAir release:** 3.10.1
- **External Libraries:** Compiled and installed from portage tree:
  - sci-libs/fftw-3.1
  - sci-libs/blas-reference-19940131-r2
  - sci-libs/cblas-reference-20030223
  - sci-libs/lapack-reference-3.0-r2
- **Tests Status:** All tests PASSED
- **Comments:** BLAS and LAPACK libs set by using the following system commands:

```
% blas-config reference
% lapack-config reference
```

StdAir configured with:

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

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

## 18.3.1.6 Red Hat Enterprise Linux with StdAir External

- **Platform:** Intel Pentium 4
- **Operating System:** Red Hat Enterprise Linux AS release 4 (Nahant Update 2)
- **Compiler:** g++ (GCC) 3.4.4 20050721 (Red Hat 3.4.4-2)
- **StdAir release:** 3.10.0
- **External Libraries:** BLAS, CBLAS, LAPACK and FFTW libraries from StdAir External 2.1.1 package
- **Tests Status:** All tests PASSED
- **Date:** March 7, 2006
- **Tester:** Erik G. Larsson

## 18.3.1.7 SUSE Linux 10.0 with NetLib's BLAS and LAPACK

- **Platform:** Intel Pentium 4 CPU 3.20GHz (64-bit)
- **Operating System:** SUSE Linux 10.0 (x86\_64)
- **Compiler(s):** g++ (GCC) 4.0.2
- **StdAir release:** 3.10.0
- **External Libraries:** BLAS, LAPACK and FFTW libraries installed from OpenSuse 10.0 RPM repository:
  - blas-3.0-926
  - lapack-3.0-926
  - fftw3-3.0.1-114
  - fftw3-threads-3.0.1-114
  - fftw3-devel-3.0.1-114
- **Tests Status:** All tests PASSED
- **Comments:** StdAir configured with:
 

```
% export CXXFLAGS="-m64 -march=nocona -O3 -pipe"
% ./configure --with-lapack="/usr/lib64/liblapack.so.3"
```
- **Date:** March 1, 2006
- **Tester:** Adam Piatyszek (ediap)

## 18.3.1.8 SUSE Linux 10.0 with MKL

- **Platform:** Intel Pentium 4 CPU 3.20GHz (64-bit)
- **Operating System:** SUSE Linux 10.0 (x86\_64)
- **Compiler(s):** g++ (GCC) 4.0.2
- **StdAir release:** 3.10.0
- **External Libraries:** Intel Math Kernel Library (MKL) 8.0.1 installed manually in the following directory↵  
: /opt/intel/mkl/8.0.1
- **Tests Status:** All tests PASSED
- **Comments:** StdAir configured with:
 

```
% export CXXFLAGS="-m64 -march=nocona -O3 -pipe"
% export LDFLAGS="-L/opt/intel/mkl/8.0.1/lib/em64t"
% export CPPFLAGS="-I/opt/intel/mkl/8.0.1/include"
% ./configure
```
- **Date:** March 1, 2006
- **Tester:** Adam Piatyszek (ediap)

### 18.3.2 Windows Systems

#### 18.3.2.1 Microsoft Windows XP with Cygwin

- **Platform:** AMD Sempron 3000+
- **Operating System:** Microsoft Windows XP SP2, Cygwin 1.5.19-4
- **Compiler(s):** g++ (GCC) 3.4.4 (cygming special)
- **StdAir release:** 3.10.1
- **External Libraries:** Installed from Cygwin's repository:
  - `fftw-3.0.1-2`
  - `fftw-dev-3.0.1-1`
  - `lapack-3.0-4`
- **Tests Status:** All tests PASSED
- **Comments:** Only static library can be built. StdAir configured with:

```
% ./configure
```

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

#### 18.3.2.2 Microsoft Windows XP with Cygwin and ATLAS

- **Platform:** AMD Sempron 3000+
- **Operating System:** Microsoft Windows XP SP2, Cygwin 1.5.19-4
- **Compiler(s):** g++ (GCC) 3.4.4 (cygming special)
- **StdAir release:** 3.10.1
- **External Libraries:** Installed from Cygwin's repository:

```
– fftw-3.0.1-2
– fftw-dev-3.0.1-1
```

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

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

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

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

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

## 18.3.2.3 Microsoft Windows XP with Cygwin and ACML

- **Platform:** AMD Sempron 3000+
- **Operating System:** Microsoft Windows XP SP2, Cygwin 1.5.19-4
- **Compiler(s):** g++ (GCC) 3.4.4 (cygming special)
- **StdAir release:** 3.10.2
- **External Libraries:** ACML version 3.1.0 (acml3.1.0-32-win32-g77.exe) installed into a default directory, i.e. "c:\Program Files\AMD\acml3.1.0"
- **Tests Status:** All tests PASSED
- **Comments:** Only static library can be built. StdAir configured with:
 

```
% export LDFLAGS="-L/cygdrive/c/Progra~1/AMD/acml3.1.0/gnu32/lib"
% export CPPFLAGS="-I/cygdrive/c/Progra~1/AMD/acml3.1.0/gnu32/include"
% ./configure --enable-debug
```
- **Date:** May 15, 2006
- **Tester:** Adam Piatyszek (ediap)

## 18.3.2.4 Microsoft Windows XP with MinGW, MSYS and ACML

- **Platform:** AMD Sempron 3000+
- **Operating System:** Microsoft Windows XP SP2, MinGW 5.0.2, MSYS 1.0.10
- **Compiler(s):** g++ (GCC) 3.4.4 (mingw special)
- **StdAir release:** 3.10.2
- **External Libraries:** ACML version 3.1.0 (acml3.1.0-32-win32-g77.exe) installed into a default directory, i.e. "c:\Program Files\AMD\acml3.1.0"
- **Tests Status:** All tests PASSED
- **Comments:** Only static library can be built. StdAir configured with:
 

```
% export LDFLAGS="-L/c/Progra~1/AMD/acml3.1.0/gnu32/lib"
% export CPPFLAGS="-I/c/Progra~1/AMD/acml3.1.0/gnu32/include"
% ./configure --enable-debug
```
- **Date:** May 15, 2006
- **Tester:** Adam Piatyszek (ediap)

## 18.3.2.5 Microsoft Windows XP with MinGW, MSYS and StdAir External

- **Platform:** AMD Sempron 3000+
- **Operating System:** Microsoft Windows XP SP2, MinGW 5.0.2, MSYS 1.0.10
- **Compiler(s):** g++ (GCC) 3.4.4 (mingw special)
- **StdAir release:** 3.10.5
- **External Libraries:** BLAS, CBLAS, LAPACK and FFTW libraries from StdAir External 2.2.0 package
- **Tests Status:** All tests PASSED
- **Comments:** Only static library can be built. StdAir configured with:
 

```
% export LDFLAGS="-L/usr/local/lib"
% export CPPFLAGS="-I/usr/local/include"
% export CXXFLAGS="-Wall -O3 -march=athlon-tbird -pipe"
% ./configure --disable-html-doc
```
- **Date:** August 11, 2006
- **Tester:** Adam Piatyszek (ediap)

### 18.3.2.6 Microsoft Windows XP with MS Visual C++ and Intel MKL

- **Platform:** AMD Sempron 3000+
- **Operating System:** Microsoft Windows XP SP2
- **Compiler(s):** Microsoft Visual C++ 2005 .NET
- **StdAir release:** 3.10.5
- **External Libraries:** Intel Math Kernel Library (MKL) 8.1 installed manually in the following directory: "C:\Program Files\Intel\MKL\8.1"
- **Tests Status:** Not fully tested. Some StdAir based programs compiled and run with success.
- **Comments:** Only static library can be built. StdAir built by opening the "win32\stdair.vcproj" project file in MSVC++ and executing "Build -> Build Solution" command from menu.
- **Date:** August 11, 2006
- **Tester:** Adam Piatyszek (ediap)

### 18.3.3 Unix Systems

#### 18.3.3.1 SunOS 5.9 with StdAir External

- **Platform:** SUNW, Sun-Blade-100 (SPARC)
- **Operating System:** SunOS 5.9 Generic\_112233-10
- **Compiler(s):** g++ (GCC) 3.4.5
- **StdAir release:** 3.10.2
- **External Libraries:** BLAS, CBLAS, LAPACK and FFTW libraries from StdAir External 2.1.1 package. The following configuration command has been used:

```
% export CFLAGS="-mcpu=ultrasparc -O2 -pipe -funroll-all-loops"
% ./configure
```

- **Tests Status:** All tests PASSED
- **Comments:** StdAir configured with:

```
% export LDFLAGS="-L/usr/local/lib"
% export CPPFLAGS="-I/usr/local/include"
% export CXXFLAGS="-mcpu=ultrasparc -O2 -pipe"
% ./configure --enable-debug
```

- **Date:** May 15, 2006
- **Tester:** Adam Piatyszek (ediap)

## 19 StdAir Supported Systems (Previous Releases)

### 19.1 StdAir 3.9.1

### 19.2 StdAir 3.9.0

### 19.3 StdAir 3.8.1

## 20 Tutorials

### 20.1 Table of Contents

- [Introduction](#)
  - [Preparing the StdAir Project for Development](#)
- [Build a Predefined BOM Tree](#)
  - [Instantiate the BOM Root Object](#)
  - [Instantiate the \(Airline\) Inventory Object](#)
  - [Link the Inventory Object with the BOM Root](#)
  - [Build Another Airline Inventory](#)
  - [Dump The BOM Tree Content](#)
  - [Result of the Tutorial Program](#)
- [Extend the Pre-Defined BOM Tree](#)
  - [Extend an Airline Inventory Object](#)
  - [Build the Specific BOM Objects](#)
  - [Result of the Tutorial Program](#)

### 20.2 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).

#### 20.2.1 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.

### 20.3 Build a Predefined BOM Tree

A few steps:

- [Instantiate the BOM Root Object](#)
- [Instantiate the \(Airline\) Inventory Object](#)
- [Link the Inventory Object with the BOM Root](#)

#### 20.3.1 Instantiate the BOM Root Object

First, a BOM root object (i.e., a root for all the classes in the project) is instantiated by the `stdair::STD←AIR_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) {
```

### 20.3.2 Instantiate the (Airline) Inventory Object

An airline inventory object can then be instantiated. Let us give it the "BA" airline code (corresponding to **British Airways**) as the object key. That is, an object (let us name it `lBAKey`) of type (class) `stdair::InventoryKey` has first to be instantiated.

```
const InventoryKey lBAKey (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);
```

### 20.3.3 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);
```

### 20.3.4 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.

### 20.3.5 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.

### 20.3.6 Result of the Tutorial Program

When the `stdair.cpp` program is run (with the `-b` option), the output should look like:

```
00001 [D].././batches/stdair.cpp:243: Welcome to stdair
00002 [D]../././stdair/command/CmdBomManager.cpp:41: StdAir will build the BOM tree from built-in
specifications.
00003 [D].././batches/stdair.cpp:286:
00004 =====
00005 BomRoot:  -- ROOT --
00006 =====
00007 ++++++
00008 Inventory: BA
00009 ++++++
00010 *****
00011 FlightDate: BA9, 2011-Jun-10
00012 *****
00013 *****
00014 Leg-Dates:
00015 -----
00016 Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, Elapsed, Distance,
Capacity,
00017 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,
00018 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,
00019 *****
00020 *****
00021 LegCabins:
00022 -----
00023 Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, CommSpace, AvPool, Avl,
NAV, GAV, ACP, ETB, BidPrice,
00024 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,
00025 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,
00026 *****
00027 *****
00028 Buckets:
00029 -----
00030 Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
00031 *****
00032 *****
00033 SegmentCabins:
00034 -----
00035 Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
00036 BA9 2011-Jun-10, LHR-SYD 2011-Jun-10, Y, EcoSaver, 0, 0, 0, 0, 9, 0,
00037 BA9 2011-Jun-10, LHR-BKK 2011-Jun-10, Y, EcoSaver, 0, 0, 0, 0, 9, 0,
00038 BA9 2011-Jun-10, BKK-SYD 2011-Jun-11, Y, EcoSaver, 0, 0, 0, 0, 9, 0,
00039 *****
00040 *****
00041 Subclasses:
00042 -----
00043 Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks (pdg), StfBkgs,
WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
00044 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,
00045 ++++++
00046 Inventory: AF
00047 ++++++
00048 *****
00049 FlightDate: AF84, 2011-Mar-20
00050 *****
00051 *****
00052 Leg-Dates:
00053 -----
00054 Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, Elapsed, Distance,
Capacity,
00055 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,
00056 *****
00057 *****
00058 LegCabins:
00059 -----
00060 Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, CommSpace, AvPool, Avl,
NAV, GAV, ACP, ETB, BidPrice,
00061 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,
```

```

00062 *****
00063 *****
00064 Buckets:
00065 -----
00066 Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
00067 *****
00068 *****
00069 SegmentCabins:
00070 -----
00071 Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
00072 AF84 2011-Mar-20, CDG-SFO 2011-Mar-20, Y, EcoSaver, 0, 0, 0, 0, 9, 0,
00073 *****
00074 *****
00075 Subclasses:
00076 -----
00077 Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks (pdg), StfBkgs,
      WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
00078 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,
00079

```

See the corresponding full program ([Command-Line Utility to Demonstrate Typical StdAir Usage](#)) for more details.

## 20.4 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.

### 20.4.1 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.

### 20.4.2 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 (lFAirlineCode);
myprovider::Inventory& lAFInv =
    stdair::FacBom<myprovider::Inventory>::instance().
        create (lAFKey);
stdair::FacBomManager::addToList (lBomRoot, lAFInv);
```

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

```
const myprovider::InventoryList_T& lInventoryList =
    stdair::BomManager::getList<myprovider::Inventory> (lBomRoot);
```

... and browsed:

```
for (myprovider::InventoryList_T::const_iterator itInv =
    lInventoryList.begin(); itInv != lInventoryList.end();
    ++itInv, ++idx) {
    const myprovider::Inventory* lInv_ptr = *itInv;
    BOOST_REQUIRE (lInv_ptr != NULL);

    BOOST_CHECK_EQUAL (lInventoryKeyArray[idx], lInv_ptr->describeKey());
    BOOST_CHECK_MESSAGE (lInventoryKeyArray[idx] == lInv_ptr->describeKey(),
        "They inventory key, '" << lInventoryKeyArray[idx]
        << "', does not match that of the Inventory object: '"
        << lInv_ptr->describeKey() << "'");
}
```

### 20.4.3 Result of the Tutorial Program

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

```
00001 Inventory: BA
00002 Inventory: AF
```

See the corresponding full program ([Command-Line Test to Demonstrate How To Extend StdAir BOM](#)) for more details.

## 21 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
            given with the -i/--input option")
        ("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_FILENAME),
            "(CVS) input file for the demand distributions")
        ("log,l",
            boost::program_options::value< std::string >(&ioLogFilename)->default_value(
                K_STDAIR_DEFAULT_LOG_FILENAME),
            "Filename for the logs")
        ;

    // Hidden options, will be allowed both on command line and
    // in config file, but will not be shown to the user.
    boost::program_options::options_description hidden ("Hidden options");
    hidden.add_options()
        ("copyright",
            boost::program_options::value< std::vector<std::string> >(),
            "Show the copyright (license)");

    boost::program_options::options_description cmdline_options;
    cmdline_options.add(generic).add(config).add(hidden);

    boost::program_options::options_description config_file_options;
    config_file_options.add(config).add(hidden);
    boost::program_options::options_description visible ("Allowed options");
    visible.add(generic).add(config);

    boost::program_options::positional_options_description p;
    p.add ("copyright", -1);

    boost::program_options::variables_map vm;
    boost::program_options::store (boost::program_options::command_line_parser (argc, argv).
        options (cmdline_options).positional(p).run(), vm);

    std::ifstream ifs ("stdair.cfg");
    boost::program_options::store (parse_config_file (ifs, config_file_options),
        vm);
    boost::program_options::notify (vm);

```

```

    if (vm.count ("help")) {
        std::cout << visible << std::endl;
        return K_STDAIR_EARLY_RETURN_STATUS;
    }

    if (vm.count ("version")) {
        std::cout << PACKAGE_NAME << ", version " << PACKAGE_VERSION << std::endl;
        return K_STDAIR_EARLY_RETURN_STATUS;
    }

    if (vm.count ("prefix")) {
        std::cout << "Installation prefix: " << PREFIXDIR << std::endl;
        return K_STDAIR_EARLY_RETURN_STATUS;
    }

    if (vm.count ("builtin")) {
        ioIsBuiltin = true;
    }

    if (vm.count ("rmol")) {
        ioIsForRMOL = true;

        // The RMOL sample tree takes precedence over the default built-in BOM tree
        ioIsBuiltin = false;
    }

    if (vm.count ("crs")) {
        ioIsForCRS = true;

        // The RMOL sample tree takes precedence over the default built-in BOM tree
        ioIsBuiltin = false;
    }

    const std::string isBuiltinStr = (ioIsBuiltin == true)?"yes":"no";
    std::cout << "The BOM should be built-in? " << isBuiltinStr << std::endl;

    const std::string isForRMOLStr = (ioIsForRMOL == true)?"yes":"no";
    std::cout << "The BOM should be built-in for RMOL? " << isForRMOLStr
        << std::endl;

    const std::string isForCRSStr = (ioIsForCRS == true)?"yes":"no";
    std::cout << "The BOM should be built-in for CRS? " << isForCRSStr
        << std::endl;

    if (ioIsBuiltin == false && ioIsForRMOL == false && ioIsForCRS == false) {
        if (vm.count ("input")) {
            ioInputFilename = vm["input"].as< std::string >();
            std::cout << "Input filename is: " << ioInputFilename << std::endl;
        } else {
            std::cerr << "Either one among the -b/--builtin, -r/--rmol, -c/--crs "
                << "or -i/--input options must be specified" << std::endl;
        }
    }

    if (vm.count ("log")) {
        ioLogFilename = vm["log"].as< std::string >();
        std::cout << "Log filename is: " << ioLogFilename << std::endl;
    }

    return 0;
}

// ////////////////////////////////// M A I N //////////////////////////////////
int main (int argc, char* argv[]) {

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

    // State whether a sample BOM tree should be built for RMOL.
    bool isForRMOL;

    // State whether a sample BOM tree should be built for the CRS.
    bool isForCRS;

    // Input file name
    stdair::Filename_T lInputFilename;

    // Output log File
    std::string lLogFilename;

    // Call the command-line option parser
    const int lOptionParserStatus =
        readConfiguration (argc, argv, isBuiltin, isForRMOL, isForCRS,

```

```

        lInputFilename, lLogFilename);

if (lOptionParserStatus == K_STDAIR_EARLY_RETURN_STATUS) {
    return 0;
}

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

const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG,
    logOutputFile);
stdair::STDAIR_Service stdairService (lLogParams);

// DEBUG
STDAIR_LOG_DEBUG ("Welcome to stdair");

// Check 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;
}

```

## 22 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 () {
    }

}
```

## 23 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&);
    };

}
```

## 24 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();
}
}

```

## 25 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;
}

```

## 26 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 lFAAirlineCode ("AF");
    const stdair::InventoryKey lAFKey (lFAAirlineCode);
    myprovider::Inventory& lAFInv =
        stdair::FacBom<myprovider::Inventory>::instance().
            create (lAFKey);
    stdair::FacBomManager::addToList (lBomRoot, lAFInv);

    BOOST_CHECK_EQUAL (lAFInv.describeKey(), lFAAirlineCode);
    BOOST_CHECK_MESSAGE (lAFInv.describeKey() == lFAAirlineCode,
        "The inventory key, '" << lAFInv.describeKey()
        << "', should be equal to '" << lFAAirlineCode
        << "', but is not");

    // Browse the inventories
    const myprovider::InventoryList_T& lInventoryList =
        stdair::BomManager::getList<myprovider::Inventory> (lBomRoot);
    const std::string lInventoryKeyArray[2] = {lBAAirlineCode, lFAAirlineCode};
    short idx = 0;
    for (myprovider::InventoryList_T::const_iterator itInv =
        lInventoryList.begin(); itInv != lInventoryList.end();
        ++itInv, ++idx) {
        const myprovider::Inventory* lInv_ptr = *itInv;
        BOOST_REQUIRE (lInv_ptr != NULL);

        BOOST_CHECK_EQUAL (lInventoryKeyArray[idx], lInv_ptr->describeKey());
        BOOST_CHECK_MESSAGE (lInventoryKeyArray[idx] == lInv_ptr->describeKey(),
            "They inventory key, '" << lInventoryKeyArray[idx]
            << "', does not match that of the Inventory object: '"

```

```

        << lInv_ptr->describeKey() << "'";
    }
}

BOOST_AUTO_TEST_CASE (bom_structure_serialisation_test) {

    // Backup (thanks to Boost.Serialisation) file
    const std::string lBackupFilename = "StandardAirlineITTestSuite_serial.txt";

    // Output log File
    const std::string lLogFilename ("StandardAirlineITTestSuite_serial.log");

    // Set the log parameters
    std::ofstream logOutputFile;

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

    // Initialise the stdair BOM
    const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG,
        logOutputFile);
    stdair::STDAIR_Service stdairService (lLogParams);

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

    // 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 original state
    stdair::BomRoot& lRestoredBomRoot =
        stdair::FacBom<stdair::BomRoot>::instance().
            create();
    {
        // create and open an archive for input
        std::ifstream ifs (lBackupFilename.c_str());
        boost::archive::text_iarchive ia(ifs);
        // read class state from archive
        ia >> lRestoredBomRoot;
        // archive and stream closed when destructors are called
    }

    // DEBUG: Display the whole 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

```

## 27 Namespace Index

### 27.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<b>boost</b>	
<b>Forward declarations</b>	??
<b>boost::serialization</b>	??
<b>bpt</b>	??
<b>soci</b>	??
<b>stdair</b>	
<b>Handle on the StdAir library context</b>	??
<b>stdair::LOG</b>	??
<b>stdair_test</b>	??
<b>swift</b>	
<b>The wrapper namespace</b>	??

## 28 Hierarchical Index

### 28.1 Class Hierarchy

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

std::allocator< T >	
std::array< T >	
std::auto_ptr< T >	
<b>stdair::BasChronometer</b>	??
<b>stdair::BasFileMgr</b>	??
std::basic_string< Char >	
std::string	
std::wstring	
std::basic_string< char >	

std::basic\_string< wchar\_t >  
std::bitset< Bits >

<b>stdair::BomAbstract</b>	??
<b>stdair::AirlineClassList</b>	??
<b>stdair::AirlineFeature</b>	??
<b>stdair::AirportPair</b>	??
<b>stdair::BomHolder&lt; BOM &gt;</b>	??
<b>stdair::BomRoot</b>	??
<b>stdair::BookingClass</b>	??
<b>stdair::Bucket</b>	??
<b>stdair::DatePeriod</b>	??
<b>stdair::FareFamily</b>	??
<b>stdair::FareFeatures</b>	??
<b>stdair::FlightDate</b>	??
<b>stdair::FlightPeriod</b>	??
<b>stdair::Inventory</b>	??
<b>stdair::LegCabin</b>	??
<b>stdair::LegDate</b>	??
<b>stdair::NestingNode</b>	??
<b>stdair::OnDDate</b>	??
<b>stdair::Policy</b>	??
<b>stdair::PosChannel</b>	??
<b>stdair::SegmentCabin</b>	??
<b>stdair::SegmentDate</b>	??
<b>stdair::SegmentPeriod</b>	??
<b>stdair::SegmentSnapshotTable</b>	??
<b>stdair::SimpleNestingStructure</b>	??
<b>stdair::TimePeriod</b>	??
<b>stdair::YieldFeatures</b>	??
<b>stdair::YieldStore</b>	??
<b>stdair::BomArchive</b>	??
<b>stdair::BomDisplay</b>	??

<b>stdair::BomID&lt; BOM &gt;</b>	<b>??</b>
<b>stdair::BomINIImport</b>	<b>??</b>
<b>stdair::BomJSONExport</b>	<b>??</b>
<b>stdair::BomJSONImport</b>	<b>??</b>
<b>stdair::BomKeyManager</b>	<b>??</b>
<b>stdair::BomManager</b>	<b>??</b>
<b>stdair::BomRetriever</b>	<b>??</b>
<b>stdair_test::BookingClass</b>	<b>??</b>
<b>stdair_test::Cabin</b>	<b>??</b>
<b>stdair::CmdAbstract</b>	<b>??</b>
<b>stdair::CmdBomManager</b>	<b>??</b>
<b>stdair::CmdBomSerialiser</b>	<b>??</b>
<b>stdair::CmdCloneBomManager</b>	<b>??</b>
<b>stdair::DBManagerForAirlines</b>	<b>??</b>
<b>COMMAND</b>	<b>??</b>
std::complex	
std::list< T >::const_iterator	
std::forward_list< T >::const_iterator	
std::map< K, T >::const_iterator	
std::unordered_map< K, T >::const_iterator	
std::basic_string< Char >::const_iterator	
std::multimap< K, T >::const_iterator	
std::unordered_multimap< K, T >::const_iterator	
std::set< K >::const_iterator	
std::string::const_iterator	
std::unordered_set< K >::const_iterator	
std::wstring::const_iterator	
std::multiset< K >::const_iterator	
std::unordered_multiset< K >::const_iterator	
std::vector< T >::const_iterator	
std::deque< T >::const_iterator	
std::list< T >::const_reverse_iterator	
std::forward_list< T >::const_reverse_iterator	
std::map< K, T >::const_reverse_iterator	
std::unordered_map< K, T >::const_reverse_iterator	
std::multimap< K, T >::const_reverse_iterator	
std::basic_string< Char >::const_reverse_iterator	
std::unordered_multimap< K, T >::const_reverse_iterator	
std::set< K >::const_reverse_iterator	
std::string::const_reverse_iterator	
std::unordered_set< K >::const_reverse_iterator	
std::multiset< K >::const_reverse_iterator	
std::wstring::const_reverse_iterator	
std::unordered_multiset< K >::const_reverse_iterator	
std::vector< T >::const_reverse_iterator	
std::deque< T >::const_reverse_iterator	

<b>stdair::ContinuousAttributeLite&lt; T &gt;</b>	??
<b>stdair::date_time_element&lt; MIN, MAX &gt;</b>	??
<b>stdair::DbAbstract</b>	??
<b>stdair::DBSessionManager</b>	??
<b>stdair::DefaultDCPList</b>	??
<b>stdair::DefaultDtdFratMap</b>	??
<b>stdair::DefaultDtdProbMap</b>	??
<b>stdair::DefaultMap</b>	??
<b>std::deque&lt; T &gt;</b>	
<b>stdair::DictionaryManager</b>	??
std::error_category	
std::error_code	
std::error_condition	
std::exception	
std::bad_alloc	
std::bad_cast	
std::bad_exception	
std::bad_typeid	
std::ios_base::failure	
std::logic_error	
std::domain_error	
std::invalid_argument	
std::length_error	
std::out_of_range	
std::runtime_error	
std::overflow_error	
std::range_error	
std::underflow_error	
<b>stdair::RootException</b>	??
<b>stdair::DocumentNotFoundException</b>	??
<b>stdair::EventException</b>	??
<b>stdair::FileNotFoundException</b>	??
<b>stdair::KeyNotFoundException</b>	??
<b>stdair::MemoryAllocationException</b>	??
<b>stdair::NonInitialisedContainerException</b>	??
<b>stdair::NonInitialisedDBSessionManagerException</b>	??
<b>stdair::NonInitialisedLogServiceException</b>	??
<b>stdair::NonInitialisedRelationshipException</b>	??
<b>stdair::NonInitialisedServiceException</b>	??
<b>stdair::ObjectLinkingException</b>	??
<b>stdair::ObjectNotFoundException</b>	??



<b>stdair::ParserException</b>	??
<b>stdair::CodeConversionException</b>	??
<b>stdair::CodeDuplicationException</b>	??
<b>stdair::KeyDuplicationException</b>	??
<b>stdair::ObjectCreationDuplicationException</b>	??
<b>stdair::ParsingFileFailedException</b>	??
<b>stdair::SerialisationException</b>	??
<b>stdair::SimpleNestingStructException</b>	??
<b>stdair::BookingClassListEmptyInNestingStructException</b>	??
<b>stdair::SQLDatabaseException</b>	??
<b>stdair::SQLDatabaseConnectionImpossibleException</b>	??
<b>stdair::FacAbstract</b>	??
<b>stdair::FacBom&lt; BOM &gt;</b>	??
<b>stdair::FacBomManager</b>	??
<b>stdair::FacCloneBom&lt; BOM &gt;</b>	??
<b>stdair::FacServiceAbstract</b>	??
<b>stdair::FacSTDAIRServiceContext</b>	??
<b>stdair::FacSupervisor</b>	??
<b>FloatingPoint&lt; RawType &gt;</b>	??
std::forward_list< T >	
std::ios_base	
basic_ios< char >	
basic_ios< wchar_t >	
std::basic_ios	
basic_istream< char >	
basic_istream< wchar_t >	
basic_ostream< char >	
basic_ostream< wchar_t >	
std::basic_istream	
basic_ifstream< char >	
basic_ifstream< wchar_t >	
basic_iostream< char >	
basic_iostream< wchar_t >	
basic_istringstream< char >	
basic_istringstream< wchar_t >	
std::basic_ifstream	
std::ifstream	
std::wifstream	
std::basic_iostream	
basic_fstream< char >	
basic_fstream< wchar_t >	
basic_stringstream< char >	
basic_stringstream< wchar_t >	

std::basic_fstream	
std::fstream	
std::wfstream	
std::basic_stringstream	
std::stringstream	
std::wstringstream	
std::basic_istream	
std::istream	
std::wistream	
std::basic_ostream	
basic_iostream< char >	
basic_iostream< wchar_t >	
basic_ofstream< char >	
basic_ofstream< wchar_t >	
basic_ostringstream< char >	
basic_ostringstream< wchar_t >	
std::basic_iostream	
std::basic_ofstream	
std::ofstream	
std::wofstream	
std::basic_ostringstream	
std::ostringstream	
std::wostringstream	
std::ostream	
std::wostream	
std::ios	
std::wios	
std::forward_list< T >::iterator	
std::map< K, T >::iterator	
std::unordered_map< K, T >::iterator	
std::multimap< K, T >::iterator	
std::basic_string< Char >::iterator	
std::unordered_multimap< K, T >::iterator	
std::set< K >::iterator	
std::string::iterator	
std::unordered_set< K >::iterator	
std::wstring::iterator	
std::multiset< K >::iterator	
std::list< T >::iterator	
std::unordered_multiset< K >::iterator	
std::vector< T >::iterator	
std::deque< T >::iterator	
<b>stdair::JSONString</b>	<b>??</b>
<b>stdair::KeyAbstract</b>	<b>??</b>
<b>stdair::AirlineClassListKey</b>	<b>??</b>
<b>stdair::AirlineFeatureKey</b>	<b>??</b>
<b>stdair::AirportPairKey</b>	<b>??</b>
<b>stdair::BomHolderKey</b>	<b>??</b>
<b>stdair::BomRootKey</b>	<b>??</b>
<b>stdair::BookingClassKey</b>	<b>??</b>

<b>stdair::BucketKey</b>	??
<b>stdair::DatePeriodKey</b>	??
<b>stdair::FareFamilyKey</b>	??
<b>stdair::FareFeaturesKey</b>	??
<b>stdair::FlightDateKey</b>	??
<b>stdair::FlightPeriodKey</b>	??
<b>stdair::InventoryKey</b>	??
<b>stdair::LegCabinKey</b>	??
<b>stdair::LegDateKey</b>	??
<b>stdair::NestingNodeKey</b>	??
<b>stdair::NestingStructureKey</b>	??
<b>stdair::OnDDateKey</b>	??
<b>stdair::ParsedKey</b>	??
<b>stdair::PolicyKey</b>	??
<b>stdair::PosChannelKey</b>	??
<b>stdair::SegmentCabinKey</b>	??
<b>stdair::SegmentDateKey</b>	??
<b>stdair::SegmentPeriodKey</b>	??
<b>stdair::SegmentSnapshotTableKey</b>	??
<b>stdair::TimePeriodKey</b>	??
<b>stdair::YieldFeaturesKey</b>	??
<b>stdair::YieldStoreKey</b>	??
std::list< T >	
std::list< BidPriceVector_T >	
std::list< BOM * >	
std::list< BookingClass * >	
std::list< BookingClassID_T >	
std::list< ClassAvailabilityMap_T >	
std::list< ClassBpvMap_T >	
std::list< ClassObjectIDMap_T >	
std::list< ClassYieldMap_T >	
std::list< FacAbstract * >	
std::list< FacServiceAbstract * >	
std::list< FareOptionStruct >	
std::list< OnDString_T >	
std::list< Policy * >	
std::list< SegmentDate * >	
std::list< std::string >	
std::list< VirtualClassStruct >	

<b>stdair::Logger</b>	??
std::map< K, T >	
std::map< CabinCode_T, ClassList_String_T >	
std::map< CabinCode_T, WTPDemandPair_T >	
std::map< const DTD_T, double >	
std::map< const DTD_T, FRAT5_T >	
std::map< const MapKey_T, BOM * >	
std::map< const MapKey_T, ClassIndex_T >	
std::map< const SegmentCabin *, SegmentDataID_T >	
std::map< const std::string, FFDisutilityCurve_T >	
std::map< const std::string, FRAT5Curve_T >	
std::map< const std::type_info *, BomAbstract * >	
std::map< const Yield_T, double >	
std::map< std::string, CabinClassPairList_T >	
std::map< std::string, YieldDemandPair_T >	
std::map< YieldLevel_T, MeanStdDevPair_T >	
std::multimap< K, T >	
std::multiset< K >	
std::priority_queue< T >	
std::queue< T >	
std::forward_list< T >::reverse_iterator	
std::list< T >::reverse_iterator	
std::unordered_multiset< K >::reverse_iterator	
std::unordered_multimap< K, T >::reverse_iterator	
std::unordered_map< K, T >::reverse_iterator	
std::set< K >::reverse_iterator	
std::map< K, T >::reverse_iterator	
std::string::reverse_iterator	
std::basic_string< Char >::reverse_iterator	
std::unordered_set< K >::reverse_iterator	
std::wstring::reverse_iterator	
std::multiset< K >::reverse_iterator	
std::multimap< K, T >::reverse_iterator	
std::deque< T >::reverse_iterator	
std::vector< T >::reverse_iterator	
<b>stdair::RootFilePath</b>	??
<b>stdair::InputFilePath</b>	??
<b>stdair::ConfigINIFile</b>	??
<b>stdair::FFDisutilityFilePath</b>	??
<b>stdair::FRAT5FilePath</b>	??
<b>stdair::ODFilePath</b>	??
<b>stdair::ScheduleFilePath</b>	??
<b>stdair::ServiceAbstract</b>	??
<b>stdair::STDAIR_ServiceContext</b>	??
std::set< K >	
<b>swift::SKeymap</b>	??
std::smart_ptr< T >	
<b>swift::SReadline</b>	??
std::stack< T >	

<b>stdair::STDAIR_Service</b>	<b>??</b>
<b>stdair::StructAbstract</b>	<b>??</b>
<b>stdair::AirlineStruct</b>	<b>??</b>
<b>stdair::BasDBParams</b>	<b>??</b>
<b>stdair::BasLogParams</b>	<b>??</b>
<b>stdair::BookingRequestStruct</b>	<b>??</b>
<b>stdair::BreakPointStruct</b>	<b>??</b>
<b>stdair::CancellationStruct</b>	<b>??</b>
<b>stdair::ConfigHolderStruct</b>	<b>??</b>
<b>stdair::DemandGenerationMethod</b>	<b>??</b>
<b>stdair::DoWStruct</b>	<b>??</b>
<b>stdair::EventStruct</b>	<b>??</b>
<b>stdair::EventType</b>	<b>??</b>
<b>stdair::FareOptionStruct</b>	<b>??</b>
<b>stdair::FFDisutilityCurveHolderStruct</b>	<b>??</b>
<b>stdair::ForecastingMethod</b>	<b>??</b>
<b>stdair::FRAT5CurveHolderStruct</b>	<b>??</b>
<b>stdair::JSoNCommand</b>	<b>??</b>
<b>stdair::OptimisationMethod</b>	<b>??</b>
<b>stdair::OptimisationNotificationStruct</b>	<b>??</b>
<b>stdair::PartnershipTechnique</b>	<b>??</b>
<b>stdair::PassengerChoiceModel</b>	<b>??</b>
<b>stdair::PassengerType</b>	<b>??</b>
<b>stdair::PeriodStruct</b>	<b>??</b>
<b>stdair::PreOptimisationMethod</b>	<b>??</b>
<b>stdair::ProgressStatus</b>	<b>??</b>
<b>stdair::ProgressStatusSet</b>	<b>??</b>
<b>stdair::RandomGeneration</b>	<b>??</b>
<b>stdair::RMEventStruct</b>	<b>??</b>
<b>stdair::SampleType</b>	<b>??</b>
<b>stdair::ServiceInitialisationType</b>	<b>??</b>
<b>stdair::SnapshotStruct</b>	<b>??</b>

<b>stdair::TravelSolutionStruct</b>	??
<b>stdair::UnconstrainingMethod</b>	??
<b>stdair::VirtualClassStruct</b>	??
<b>stdair::YieldRange</b>	??
std::system_error	
std::thread	
<b>soci::type_conversion&lt; stdair::AirlineStruct &gt;</b>	??
<b>TypeWithSize&lt; size &gt;</b>	??
<b>TypeWithSize&lt; 4 &gt;</b>	??
<b>TypeWithSize&lt; 8 &gt;</b>	??
<b>TypeWithSize&lt; sizeof(RawType)&gt;</b>	??
std::unique_ptr< T >	
std::unordered_map< K, T >	
std::unordered_multimap< K, T >	
std::unordered_multiset< K >	
std::unordered_set< K >	
std::valarray< T >	
std::vector< T >	
std::vector< AirlineCode_T >	
std::vector< BidPrice_T >	
std::vector< bool >	
std::vector< ClassList_String_T >	
std::vector< DictionaryKey_T >	
std::vector< double >	
std::vector< MeanStdDevPair_T >	
std::vector< ServiceAbstract * >	
std::weak_ptr< T >	
BOM *	
bool	
char	
char *	
char const *	
const Date_T	
const DateTime_T	
const DayDuration_T	
const DTD_T	
const Duration_T	
const NbOfSeats_T	
const PriceValue_T	
const size_t	
const WTP_T	
date	
date_duration	
date_period	
DBSession_T *	
double	
EN_DemandGenerationMethod	
EN_EventType	
EN_EventType	
EN_EventType	
EN_EventType	
EN_ForecastingMethod	

EN\_JSonCommand  
 EN\_LogLevel  
 EN\_LogLevel  
 EN\_OptimisationMethod  
 EN\_PartnershipTechnique  
 EN\_PassengerChoiceModel  
 EN\_PassengerType  
 EN\_PreOptimisationMethod  
 EN\_SampleType  
 EN\_ServiceInitialisationType  
 EN\_UnconstrainingMethod  
 int  
 K  
 Keymap  
 MeanStdDevPair\_T  
 minstd\_rand  
 multi\_array< double, 2 >  
 pt2Func \*  
 ptime  
 RawType  
 rl\_completion\_func\_t \*  
 shared\_ptr< BookingRequestStruct >  
 shared\_ptr< BreakPointStruct >  
 shared\_ptr< CancellationStruct >  
 shared\_ptr< ConfigHolderStruct >  
 shared\_ptr< OptimisationNotificationStruct >  
 shared\_ptr< RMEventStruct >  
 shared\_ptr< SnapshotStruct >  
 short  
 static const Bits  
 static const char  
 static const size\_t  
 T  
 time\_duration  
 type\_info \*  
 unsigned short  
 vector< T >  
 WTPDemandPair\_T  
 YieldDemandPair\_T  
 YieldLevel\_T

## 29 Class Index

### 29.1 Class List

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

<a href="#"><b>stdair::AirlineClassList</b></a>	
<b>Class representing the actual attributes for a segment-features</b>	<b>??</b>
<a href="#"><b>stdair::AirlineClassListKey</b></a>	
<b>Key of airport-pair</b>	<b>??</b>
<a href="#"><b>stdair::AirlineFeature</b></a>	
<b>Class representing various configuration parameters (e.g., revenue management methods such EMSRb or Monte-Carlo) for a given airline for the simulation</b>	<b>??</b>

<a href="#">stdair::AirlineFeatureKey</a>	??
<a href="#">stdair::AirlineStruct</a>	??
<a href="#">stdair::AirportPair</a> Class representing the actual attributes for an airport-pair	??
<a href="#">stdair::AirportPairKey</a> Key of airport-pair	??
<a href="#">stdair::BasChronometer</a>	??
<a href="#">stdair::BasDBParams</a> Structure holding the parameters for connection to a database	??
<a href="#">stdair::BasFileMgr</a>	??
<a href="#">stdair::BasLogParams</a> Structure holding parameters for logging	??
<a href="#">stdair::BomAbstract</a> Base class for the Business Object Model (BOM) layer	??
<a href="#">stdair::BomArchive</a> Utility class to archive/restore BOM objects with Boost serialisation	??
<a href="#">stdair::BomDisplay</a> Utility class to display StdAir objects with a pretty format	??
<a href="#">stdair::BomHolder&lt; BOM &gt;</a> Class representing the holder of BOM object containers (list and map)	??
<a href="#">stdair::BomHolderKey</a>	??
<a href="#">stdair::BomID&lt; BOM &gt;</a> Class wrapper of bom ID (e.g. pointer to object)	??
<a href="#">stdair::BomINIImport</a> Utility class to import StdAir objects in a INI format	??
<a href="#">stdair::BomJSONExport</a> Utility class to export StdAir objects in a JSON format	??
<a href="#">stdair::BomJSONImport</a> Utility class to import StdAir objects in a JSON format	??
<a href="#">stdair::BomKeyManager</a> Utility class to extract key structures from strings	??
<a href="#">stdair::BomManager</a> Utility class for StdAir-based objects	??
<a href="#">stdair::BomRetriever</a> Utility class to retrieve StdAir objects	??
<a href="#">stdair::BomRoot</a> Class representing the actual attributes for the Bom root	??
<a href="#">stdair::BomRootKey</a> Key of the BOM structure root	??
<a href="#">stdair_test::BookingClass</a>	??



<a href="#">stdair::BookingClass</a>	??
<a href="#">stdair::BookingClassKey</a>	??
<a href="#">stdair::BookingClassListEmptyInNestingStructException</a>	??
<a href="#">stdair::BookingRequestStruct</a>	
Structure holding the elements of a booking request	??
<a href="#">stdair::BreakPointStruct</a>	??
<a href="#">stdair::Bucket</a>	
Class representing the actual attributes for an airline booking class	??
<a href="#">stdair::BucketKey</a>	
Key of booking-class	??
<a href="#">stdair_test::Cabin</a>	??
<a href="#">stdair::CancellationStruct</a>	
Structure holding the elements of a travel solution	??
<a href="#">stdair::CmdAbstract</a>	??
<a href="#">stdair::CmdBomManager</a>	??
<a href="#">stdair::CmdBomSerialiser</a>	??
<a href="#">stdair::CmdCloneBomManager</a>	??
<a href="#">stdair::CodeConversionException</a>	??
<a href="#">stdair::CodeDuplicationException</a>	??
<a href="#">COMMAND</a>	??
<a href="#">stdair::ConfigHolderStruct</a>	??
<a href="#">stdair::ConfigINIFile</a>	??
<a href="#">stdair::ContinuousAttributeLite&lt; T &gt;</a>	
Class modeling the distribution of values that can be taken by a continuous attribute	??
<a href="#">stdair::date_time_element&lt; MIN, MAX &gt;</a>	??
<a href="#">stdair::DatePeriod</a>	
Class representing the actual attributes for a fare date-period	??
<a href="#">stdair::DatePeriodKey</a>	
Key of date-period	??
<a href="#">stdair::DbAbstract</a>	??
<a href="#">stdair::DBManagerForAirlines</a>	??
<a href="#">stdair::DBSessionManager</a>	??
<a href="#">stdair::DefaultDCPList</a>	??
<a href="#">stdair::DefaultDtdFratMap</a>	??
<a href="#">stdair::DefaultDtdProbMap</a>	??

<a href="#">stdair::DefaultMap</a>	??
<a href="#">stdair::DemandGenerationMethod</a> Enumeration of demand (booking request) generation methods	??
<a href="#">stdair::DictionaryManager</a> Class wrapper of dictionary business methods	??
<a href="#">stdair::DocumentNotFoundException</a>	??
<a href="#">stdair::DoWStruct</a>	??
<a href="#">stdair::EventException</a>	??
<a href="#">stdair::EventStruct</a>	??
<a href="#">stdair::EventType</a>	??
<a href="#">stdair::FacAbstract</a>	??
<a href="#">stdair::FacBom&lt; BOM &gt;</a> Base class for Factory layer	??
<a href="#">stdair::FacBomManager</a> Utility class for linking StdAir-based objects	??
<a href="#">stdair::FacCloneBom&lt; BOM &gt;</a> Base class for Factory layer	??
<a href="#">stdair::FacServiceAbstract</a>	??
<a href="#">stdair::FacSTDAIRServiceContext</a> Factory for <a href="#">Bucket</a>	??
<a href="#">stdair::FacSupervisor</a>	??
<a href="#">stdair::FareFamily</a> Class representing the actual attributes for a family fare	??
<a href="#">stdair::FareFamilyKey</a> Key of a given fare family, made of a fare family code	??
<a href="#">stdair::FareFeatures</a> Class representing the actual attributes for a fare date-period	??
<a href="#">stdair::FareFeaturesKey</a> Key of date-period	??
<a href="#">stdair::FareOptionStruct</a> Structure holding the elements of a fare option	??
<a href="#">stdair::FFDisutilityCurveHolderStruct</a>	??
<a href="#">stdair::FFDisutilityFilePath</a>	??
<a href="#">stdair::FileNotFoundException</a>	??
<a href="#">stdair::FlightDate</a> Class representing the actual attributes for an airline flight-date	??
<a href="#">stdair::FlightDateKey</a> Key of a given flight-date, made of a flight number and a departure date	??

<a href="#">stdair::FlightPeriod</a>	??
<a href="#">stdair::FlightPeriodKey</a>	??
<a href="#">FloatingPoint&lt; RawType &gt;</a>	??
<a href="#">stdair::ForecastingMethod</a>	??
<a href="#">stdair::FRAT5CurveHolderStruct</a>	??
<a href="#">stdair::FRAT5FilePath</a>	??
<a href="#">stdair::InputFilePath</a>	??
<a href="#">stdair::Inventory</a>	
Class representing the actual attributes for an airline inventory	??
<a href="#">stdair::InventoryKey</a>	
Key of a given inventory, made of the airline code	??
<a href="#">stdair::JSoNCommand</a>	
Enumeration of json commands	??
<a href="#">stdair::JSONString</a>	
JSON-formatted string	??
<a href="#">stdair::KeyAbstract</a>	
Base class for the keys of Business Object Model (BOM) layer	??
<a href="#">stdair::KeyDuplicationException</a>	??
<a href="#">stdair::KeyNotFoundException</a>	??
<a href="#">stdair::LegCabin</a>	
Class representing the actual attributes for an airline leg-cabin	??
<a href="#">stdair::LegCabinKey</a>	
Key of a given leg-cabin, made of a cabin code (only)	??
<a href="#">stdair::LegDate</a>	??
<a href="#">stdair::LegDateKey</a>	??
<a href="#">stdair::Logger</a>	??
<a href="#">stdair::MemoryAllocationException</a>	??
<a href="#">stdair::NestingNode</a>	??
<a href="#">stdair::NestingNodeKey</a>	
Key of a given policy, made of a policy code	??
<a href="#">stdair::NestingStructureKey</a>	
Key of a given policy, made of a policy code	??
<a href="#">stdair::NonInitialisedContainerException</a>	??
<a href="#">stdair::NonInitialisedDBSessionManagerException</a>	??
<a href="#">stdair::NonInitialisedLogServiceException</a>	??
<a href="#">stdair::NonInitialisedRelationShipException</a>	??

<a href="#">stdair::NonInitialisedServiceException</a>	??
<a href="#">stdair::ObjectCreationDuplicationException</a>	??
<a href="#">stdair::ObjectLinkingException</a>	??
<a href="#">stdair::ObjectNotFoundException</a>	??
<a href="#">stdair::ODFilePath</a>	??
<a href="#">stdair::OnDDate</a> Class representing the actual attributes for an airline flight-date	??
<a href="#">stdair::OnDDateKey</a> 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)	??
<a href="#">stdair::OptimisationMethod</a>	??
<a href="#">stdair::OptimisationNotificationStruct</a>	??
<a href="#">stdair::ParsedKey</a>	??
<a href="#">stdair::ParserException</a>	??
<a href="#">stdair::ParsingFileFailedException</a>	??
<a href="#">stdair::PartnershipTechnique</a> Enumeration of partnership techniques	??
<a href="#">stdair::PassengerChoiceModel</a>	??
<a href="#">stdair::PassengerType</a>	??
<a href="#">stdair::PeriodStruct</a>	??
<a href="#">stdair::Policy</a>	??
<a href="#">stdair::PolicyKey</a> Key of a given policy, made of a policy code	??
<a href="#">stdair::PosChannel</a> Class representing the actual attributes for a fare point of sale	??
<a href="#">stdair::PosChannelKey</a> Key of point of sale and channel	??
<a href="#">stdair::PreOptimisationMethod</a>	??
<a href="#">stdair::ProgressStatus</a>	??
<a href="#">stdair::ProgressStatusSet</a>	??
<a href="#">stdair::RandomGeneration</a> Class holding a random generator	??
<a href="#">stdair::RMEventStruct</a>	??
<a href="#">stdair::RootException</a> Root of the stdair exceptions	??

<a href="#"><code>stdair::RootFilePath</code></a>	Root of the input and output files	??
<a href="#"><code>stdair::SampleType</code></a>	Enumeration of BOM sample types	??
<a href="#"><code>stdair::ScheduleFilePath</code></a>		??
<a href="#"><code>stdair::SegmentCabin</code></a>	Class representing the actual attributes for an airline segment-cabin	??
<a href="#"><code>stdair::SegmentCabinKey</code></a>	Key of a given segment-cabin, made of a cabin code (only)	??
<a href="#"><code>stdair::SegmentDate</code></a>	Class representing the actual attributes for an airline segment-date	??
<a href="#"><code>stdair::SegmentDateKey</code></a>	Key of a given segment-date, made of an origin and a destination airports	??
<a href="#"><code>stdair::SegmentPeriod</code></a>		??
<a href="#"><code>stdair::SegmentPeriodKey</code></a>		??
<a href="#"><code>stdair::SegmentSnapshotTable</code></a>	Class representing the actual attributes for an airline segment data tables	??
<a href="#"><code>stdair::SegmentSnapshotTableKey</code></a>	Key of a given guillotine block, made of a guillotine number	??
<a href="#"><code>stdair::SerialisationException</code></a>		??
<a href="#"><code>stdair::ServiceAbstract</code></a>		??
<a href="#"><code>stdair::ServiceInitialisationType</code></a>	Enumeration of service initialisation types	??
<a href="#"><code>stdair::SimpleNestingStructException</code></a>		??
<a href="#"><code>stdair::SimpleNestingStructure</code></a>		??
<a href="#"><code>swift::SKeymap</code></a>	The readline keymap wrapper	??
<a href="#"><code>stdair::SnapshotStruct</code></a>		??
<a href="#"><code>stdair::SQLDatabaseConnectionImpossibleException</code></a>		??
<a href="#"><code>stdair::SQLDatabaseException</code></a>		??
<a href="#"><code>swift::SReadline</code></a>	The readline library wrapper	??
<a href="#"><code>stdair::STDAIR_Service</code></a>	Interface for the STDAIR Services	??
<a href="#"><code>stdair::STDAIR_ServiceContext</code></a>	Class holding the context of the Stdair services	??
<a href="#"><code>stdair::StructAbstract</code></a>	Base class for the light structures	??

<a href="#"><code>stdair::TimePeriod</code></a>	
Class representing the actual attributes for a fare time-period	??
<a href="#"><code>stdair::TimePeriodKey</code></a>	
Key of time-period	??
<a href="#"><code>stdair::TravelSolutionStruct</code></a>	
Structure holding the elements of a travel solution	??
<a href="#"><code>soci::type_conversion&lt; stdair::AirlineStruct &gt;</code></a>	??
<a href="#"><code>TypeWithSize&lt; size &gt;</code></a>	??
<a href="#"><code>TypeWithSize&lt; 4 &gt;</code></a>	??
<a href="#"><code>TypeWithSize&lt; 8 &gt;</code></a>	??
<a href="#"><code>stdair::UnconstrainingMethod</code></a>	??
<a href="#"><code>stdair::VirtualClassStruct</code></a>	??
<a href="#"><code>stdair::YieldFeatures</code></a>	
Class representing the actual attributes for a yield date-period	??
<a href="#"><code>stdair::YieldFeaturesKey</code></a>	
Key of date-period	??
<a href="#"><code>stdair::YieldRange</code></a>	??
<a href="#"><code>stdair::YieldStore</code></a>	??
<a href="#"><code>stdair::YieldStoreKey</code></a>	??

## 30 File Index

### 30.1 File List

Here is a list of all files with brief descriptions:

<code>batches/</code> <a href="#"><code>stdair.cpp</code></a>	??
<code>stdair/</code> <a href="#"><code>stdair_basic_types.hpp</code></a>	??
<code>stdair/</code> <a href="#"><code>stdair_date_time_types.hpp</code></a>	??
<code>stdair/</code> <a href="#"><code>stdair_db.hpp</code></a>	??
<code>stdair/</code> <a href="#"><code>stdair_demand_types.hpp</code></a>	??
<code>stdair/</code> <a href="#"><code>stdair_event_types.hpp</code></a>	??
<code>stdair/</code> <a href="#"><code>stdair_exceptions.hpp</code></a>	??
<code>stdair/</code> <a href="#"><code>stdair_fare_types.hpp</code></a>	??
<code>stdair/</code> <a href="#"><code>stdair_file.hpp</code></a>	??
<code>stdair/</code> <a href="#"><code>stdair_inventory_types.hpp</code></a>	??
<code>stdair/</code> <a href="#"><code>stdair_json.hpp</code></a>	??

<a href="#">stdair/stdair_log.hpp</a>	??
<a href="#">stdair/stdair_maths_types.hpp</a>	??
<a href="#">stdair/stdair_rm_types.hpp</a>	??
<a href="#">stdair/STDAIR_Service.hpp</a>	??
<a href="#">stdair/stdair_service_types.hpp</a>	??
<a href="#">stdair/stdair_types.hpp</a>	??
<a href="#">stdair/basic/BasChronometer.cpp</a>	??
<a href="#">stdair/basic/BasChronometer.hpp</a>	??
<a href="#">stdair/basic/BasConst.cpp</a>	??
<a href="#">stdair/basic/BasConst_BomDisplay.hpp</a>	??
<a href="#">stdair/basic/BasConst_BookingClass.hpp</a>	??
<a href="#">stdair/basic/BasConst_DefaultObject.hpp</a>	??
<a href="#">stdair/basic/BasConst_Event.hpp</a>	??
<a href="#">stdair/basic/BasConst_General.hpp</a>	??
<a href="#">stdair/basic/BasConst_Inventory.hpp</a>	??
<a href="#">stdair/basic/BasConst_Period_BOM.hpp</a>	??
<a href="#">stdair/basic/BasConst_Request.hpp</a>	??
<a href="#">stdair/basic/BasConst_SellUpCurves.hpp</a>	??
<a href="#">stdair/basic/BasConst_TravelSolution.hpp</a>	??
<a href="#">stdair/basic/BasConst_Yield.hpp</a>	??
<a href="#">stdair/basic/BasDBParams.cpp</a>	??
<a href="#">stdair/basic/BasDBParams.hpp</a>	??
<a href="#">stdair/basic/BasFileMgr.cpp</a>	??
<a href="#">stdair/basic/BasFileMgr.hpp</a>	??
<a href="#">stdair/basic/BasLogParams.cpp</a>	??
<a href="#">stdair/basic/BasLogParams.hpp</a>	??
<a href="#">stdair/basic/BasParserHelperTypes.hpp</a>	??
<a href="#">stdair/basic/BasParserTypes.hpp</a>	??
<a href="#">stdair/basic/BasTypes.hpp</a>	??
<a href="#">stdair/basic/ContinuousAttributeLite.hpp</a>	??
<a href="#">stdair/basic/DemandGenerationMethod.cpp</a>	??
<a href="#">stdair/basic/DemandGenerationMethod.hpp</a>	??

<a href="#">stdair/basic/DictionaryManager.cpp</a>	??
<a href="#">stdair/basic/DictionaryManager.hpp</a>	??
<a href="#">stdair/basic/EventType.cpp</a>	??
<a href="#">stdair/basic/EventType.hpp</a>	??
<a href="#">stdair/basic/float_utils.hpp</a>	??
<a href="#">stdair/basic/float_utils_google.hpp</a>	??
<a href="#">stdair/basic/ForecastingMethod.cpp</a>	??
<a href="#">stdair/basic/ForecastingMethod.hpp</a>	??
<a href="#">stdair/basic/JSonCommand.cpp</a>	??
<a href="#">stdair/basic/JSonCommand.hpp</a>	??
<a href="#">stdair/basic/OptimisationMethod.cpp</a>	??
<a href="#">stdair/basic/OptimisationMethod.hpp</a>	??
<a href="#">stdair/basic/PartnershipTechnique.cpp</a>	??
<a href="#">stdair/basic/PartnershipTechnique.hpp</a>	??
<a href="#">stdair/basic/PassengerChoiceModel.cpp</a>	??
<a href="#">stdair/basic/PassengerChoiceModel.hpp</a>	??
<a href="#">stdair/basic/PassengerType.cpp</a>	??
<a href="#">stdair/basic/PassengerType.hpp</a>	??
<a href="#">stdair/basic/PreOptimisationMethod.cpp</a>	??
<a href="#">stdair/basic/PreOptimisationMethod.hpp</a>	??
<a href="#">stdair/basic/ProgressStatus.cpp</a>	??
<a href="#">stdair/basic/ProgressStatus.hpp</a>	??
<a href="#">stdair/basic/ProgressStatusSet.cpp</a>	??
<a href="#">stdair/basic/ProgressStatusSet.hpp</a>	??
<a href="#">stdair/basic/RandomGeneration.cpp</a>	??
<a href="#">stdair/basic/RandomGeneration.hpp</a>	??
<a href="#">stdair/basic/SampleType.cpp</a>	??
<a href="#">stdair/basic/SampleType.hpp</a>	??
<a href="#">stdair/basic/ServiceInitialisationType.cpp</a>	??
<a href="#">stdair/basic/ServiceInitialisationType.hpp</a>	??
<a href="#">stdair/basic/StructAbstract.hpp</a>	??
<a href="#">stdair/basic/UnconstrainingMethod.cpp</a>	??



<a href="#">stdair/basic/UnconstrainingMethod.hpp</a>	??
<a href="#">stdair/basic/YieldRange.cpp</a>	??
<a href="#">stdair/basic/YieldRange.hpp</a>	??
<a href="#">stdair/bom/AirlineClassList.cpp</a>	??
<a href="#">stdair/bom/AirlineClassList.hpp</a>	??
<a href="#">stdair/bom/AirlineClassListKey.cpp</a>	??
<a href="#">stdair/bom/AirlineClassListKey.hpp</a>	??
<a href="#">stdair/bom/AirlineClassListTypes.hpp</a>	??
<a href="#">stdair/bom/AirlineFeature.cpp</a>	??
<a href="#">stdair/bom/AirlineFeature.hpp</a>	??
<a href="#">stdair/bom/AirlineFeatureKey.cpp</a>	??
<a href="#">stdair/bom/AirlineFeatureKey.hpp</a>	??
<a href="#">stdair/bom/AirlineFeatureTypes.hpp</a>	??
<a href="#">stdair/bom/AirlineStruct.cpp</a>	??
<a href="#">stdair/bom/AirlineStruct.hpp</a>	??
<a href="#">stdair/bom/AirportPair.cpp</a>	??
<a href="#">stdair/bom/AirportPair.hpp</a>	??
<a href="#">stdair/bom/AirportPairKey.cpp</a>	??
<a href="#">stdair/bom/AirportPairKey.hpp</a>	??
<a href="#">stdair/bom/AirportPairTypes.hpp</a>	??
<a href="#">stdair/bom/BomAbstract.hpp</a>	??
<a href="#">stdair/bom/BomArchive.cpp</a>	??
<a href="#">stdair/bom/BomArchive.hpp</a>	??
<a href="#">stdair/bom/BomDisplay.cpp</a>	??
<a href="#">stdair/bom/BomDisplay.hpp</a>	??
<a href="#">stdair/bom/BomHolder.hpp</a>	??
<a href="#">stdair/bom/BomHolderKey.cpp</a>	??
<a href="#">stdair/bom/BomHolderKey.hpp</a>	??
<a href="#">stdair/bom/BomID.hpp</a>	??
<a href="#">stdair/bom/BomIDTypes.hpp</a>	??
<a href="#">stdair/bom/BomINIImport.cpp</a>	??
<a href="#">stdair/bom/BomINIImport.hpp</a>	??

<a href="#">stdair/bom/BomJSONExport.cpp</a>	??
<a href="#">stdair/bom/BomJSONExport.hpp</a>	??
<a href="#">stdair/bom/BomJSONImport.cpp</a>	??
<a href="#">stdair/bom/BomJSONImport.hpp</a>	??
<a href="#">stdair/bom/BomKeyManager.cpp</a>	??
<a href="#">stdair/bom/BomKeyManager.hpp</a>	??
<a href="#">stdair/bom/BomManager.hpp</a>	??
<a href="#">stdair/bom/BomRetriever.cpp</a>	??
<a href="#">stdair/bom/BomRetriever.hpp</a>	??
<a href="#">stdair/bom/BomRoot.cpp</a>	??
<a href="#">stdair/bom/BomRoot.hpp</a>	??
<a href="#">stdair/bom/BomRootKey.cpp</a>	??
<a href="#">stdair/bom/BomRootKey.hpp</a>	??
<a href="#">stdair/bom/BookingClass.cpp</a>	??
<a href="#">stdair/bom/BookingClass.hpp</a>	??
<a href="#">stdair/bom/BookingClassKey.cpp</a>	??
<a href="#">stdair/bom/BookingClassKey.hpp</a>	??
<a href="#">stdair/bom/BookingClassTypes.hpp</a>	??
<a href="#">stdair/bom/BookingRequestStruct.cpp</a>	??
<a href="#">stdair/bom/BookingRequestStruct.hpp</a>	??
<a href="#">stdair/bom/BookingRequestTypes.hpp</a>	??
<a href="#">stdair/bom/BreakPointStruct.cpp</a>	??
<a href="#">stdair/bom/BreakPointStruct.hpp</a>	??
<a href="#">stdair/bom/BreakPointTypes.hpp</a>	??
<a href="#">stdair/bom/Bucket.cpp</a>	??
<a href="#">stdair/bom/Bucket.hpp</a>	??
<a href="#">stdair/bom/BucketKey.cpp</a>	??
<a href="#">stdair/bom/BucketKey.hpp</a>	??
<a href="#">stdair/bom/BucketTypes.hpp</a>	??
<a href="#">stdair/bom/CancellationStruct.cpp</a>	??
<a href="#">stdair/bom/CancellationStruct.hpp</a>	??
<a href="#">stdair/bom/CancellationTypes.hpp</a>	??

<a href="#">stdair/bom/ConfigHolderStruct.cpp</a>	??
<a href="#">stdair/bom/ConfigHolderStruct.hpp</a>	??
<a href="#">stdair/bom/ConfigHolderTypes.hpp</a>	??
<a href="#">stdair/bom/DatePeriod.cpp</a>	??
<a href="#">stdair/bom/DatePeriod.hpp</a>	??
<a href="#">stdair/bom/DatePeriodKey.cpp</a>	??
<a href="#">stdair/bom/DatePeriodKey.hpp</a>	??
<a href="#">stdair/bom/DatePeriodTypes.hpp</a>	??
<a href="#">stdair/bom/DoWStruct.cpp</a>	??
<a href="#">stdair/bom/DoWStruct.hpp</a>	??
<a href="#">stdair/bom/EventStruct.cpp</a>	??
<a href="#">stdair/bom/EventStruct.hpp</a>	??
<a href="#">stdair/bom/EventTypes.hpp</a>	??
<a href="#">stdair/bom/FareFamily.cpp</a>	??
<a href="#">stdair/bom/FareFamily.hpp</a>	??
<a href="#">stdair/bom/FareFamilyKey.cpp</a>	??
<a href="#">stdair/bom/FareFamilyKey.hpp</a>	??
<a href="#">stdair/bom/FareFamilyTypes.hpp</a>	??
<a href="#">stdair/bom/FareFeatures.cpp</a>	??
<a href="#">stdair/bom/FareFeatures.hpp</a>	??
<a href="#">stdair/bom/FareFeaturesKey.cpp</a>	??
<a href="#">stdair/bom/FareFeaturesKey.hpp</a>	??
<a href="#">stdair/bom/FareFeaturesTypes.hpp</a>	??
<a href="#">stdair/bom/FareOptionStruct.cpp</a>	??
<a href="#">stdair/bom/FareOptionStruct.hpp</a>	??
<a href="#">stdair/bom/FareOptionTypes.hpp</a>	??
<a href="#">stdair/bom/FFDisutilityCurveHolderStruct.cpp</a>	??
<a href="#">stdair/bom/FFDisutilityCurveHolderStruct.hpp</a>	??
<a href="#">stdair/bom/FlightDate.cpp</a>	??
<a href="#">stdair/bom/FlightDate.hpp</a>	??
<a href="#">stdair/bom/FlightDateKey.cpp</a>	??
<a href="#">stdair/bom/FlightDateKey.hpp</a>	??

<a href="#">stdair/bom/FlightDateTypes.hpp</a>	??
<a href="#">stdair/bom/FlightPeriod.cpp</a>	??
<a href="#">stdair/bom/FlightPeriod.hpp</a>	??
<a href="#">stdair/bom/FlightPeriodKey.cpp</a>	??
<a href="#">stdair/bom/FlightPeriodKey.hpp</a>	??
<a href="#">stdair/bom/FlightPeriodTypes.hpp</a>	??
<a href="#">stdair/bom/FRAT5CurveHolderStruct.cpp</a>	??
<a href="#">stdair/bom/FRAT5CurveHolderStruct.hpp</a>	??
<a href="#">stdair/bom/Inventory.cpp</a>	??
<a href="#">stdair/bom/Inventory.hpp</a>	??
<a href="#">stdair/bom/InventoryKey.cpp</a>	??
<a href="#">stdair/bom/InventoryKey.hpp</a>	??
<a href="#">stdair/bom/InventoryTypes.hpp</a>	??
<a href="#">stdair/bom/key_types.hpp</a>	??
<a href="#">stdair/bom/KeyAbstract.hpp</a>	??
<a href="#">stdair/bom/LegCabin.cpp</a>	??
<a href="#">stdair/bom/LegCabin.hpp</a>	??
<a href="#">stdair/bom/LegCabinKey.cpp</a>	??
<a href="#">stdair/bom/LegCabinKey.hpp</a>	??
<a href="#">stdair/bom/LegCabinTypes.hpp</a>	??
<a href="#">stdair/bom/LegDate.cpp</a>	??
<a href="#">stdair/bom/LegDate.hpp</a>	??
<a href="#">stdair/bom/LegDateKey.cpp</a>	??
<a href="#">stdair/bom/LegDateKey.hpp</a>	??
<a href="#">stdair/bom/LegDateTypes.hpp</a>	??
<a href="#">stdair/bom/NestingNode.cpp</a>	??
<a href="#">stdair/bom/NestingNode.hpp</a>	??
<a href="#">stdair/bom/NestingNodeKey.cpp</a>	??
<a href="#">stdair/bom/NestingNodeKey.hpp</a>	??
<a href="#">stdair/bom/NestingNodeTypes.hpp</a>	??
<a href="#">stdair/bom/NestingStructureKey.cpp</a>	??
<a href="#">stdair/bom/NestingStructureKey.hpp</a>	??

<a href="#">stdair/bom/OnDDate.cpp</a>	??
<a href="#">stdair/bom/OnDDate.hpp</a>	??
<a href="#">stdair/bom/OnDDateKey.cpp</a>	??
<a href="#">stdair/bom/OnDDateKey.hpp</a>	??
<a href="#">stdair/bom/OnDDateTypes.hpp</a>	??
<a href="#">stdair/bom/OptimisationNotificationStruct.cpp</a>	??
<a href="#">stdair/bom/OptimisationNotificationStruct.hpp</a>	??
<a href="#">stdair/bom/OptimisationNotificationTypes.hpp</a>	??
<a href="#">stdair/bom/ParsedKey.cpp</a>	??
<a href="#">stdair/bom/ParsedKey.hpp</a>	??
<a href="#">stdair/bom/PeriodStruct.cpp</a>	??
<a href="#">stdair/bom/PeriodStruct.hpp</a>	??
<a href="#">stdair/bom/Policy.cpp</a>	??
<a href="#">stdair/bom/Policy.hpp</a>	??
<a href="#">stdair/bom/PolicyKey.cpp</a>	??
<a href="#">stdair/bom/PolicyKey.hpp</a>	??
<a href="#">stdair/bom/PolicyTypes.hpp</a>	??
<a href="#">stdair/bom/PosChannel.cpp</a>	??
<a href="#">stdair/bom/PosChannel.hpp</a>	??
<a href="#">stdair/bom/PosChannelKey.cpp</a>	??
<a href="#">stdair/bom/PosChannelKey.hpp</a>	??
<a href="#">stdair/bom/PosChannelTypes.hpp</a>	??
<a href="#">stdair/bom/RMEventStruct.cpp</a>	??
<a href="#">stdair/bom/RMEventStruct.hpp</a>	??
<a href="#">stdair/bom/RMEventTypes.hpp</a>	??
<a href="#">stdair/bom/SegmentCabin.cpp</a>	??
<a href="#">stdair/bom/SegmentCabin.hpp</a>	??
<a href="#">stdair/bom/SegmentCabinKey.cpp</a>	??
<a href="#">stdair/bom/SegmentCabinKey.hpp</a>	??
<a href="#">stdair/bom/SegmentCabinTypes.hpp</a>	??
<a href="#">stdair/bom/SegmentDate.cpp</a>	??
<a href="#">stdair/bom/SegmentDate.hpp</a>	??

<a href="#">stdair/bom/SegmentDateKey.cpp</a>	??
<a href="#">stdair/bom/SegmentDateKey.hpp</a>	??
<a href="#">stdair/bom/SegmentDateTypes.hpp</a>	??
<a href="#">stdair/bom/SegmentPeriod.cpp</a>	??
<a href="#">stdair/bom/SegmentPeriod.hpp</a>	??
<a href="#">stdair/bom/SegmentPeriodKey.cpp</a>	??
<a href="#">stdair/bom/SegmentPeriodKey.hpp</a>	??
<a href="#">stdair/bom/SegmentPeriodTypes.hpp</a>	??
<a href="#">stdair/bom/SegmentSnapshotTable.cpp</a>	??
<a href="#">stdair/bom/SegmentSnapshotTable.hpp</a>	??
<a href="#">stdair/bom/SegmentSnapshotTableKey.cpp</a>	??
<a href="#">stdair/bom/SegmentSnapshotTableKey.hpp</a>	??
<a href="#">stdair/bom/SegmentSnapshotTableTypes.hpp</a>	??
<a href="#">stdair/bom/SimpleNestingStructure.cpp</a>	??
<a href="#">stdair/bom/SimpleNestingStructure.hpp</a>	??
<a href="#">stdair/bom/SimpleNestingStructureTypes.hpp</a>	??
<a href="#">stdair/bom/SnapshotStruct.cpp</a>	??
<a href="#">stdair/bom/SnapshotStruct.hpp</a>	??
<a href="#">stdair/bom/SnapshotTypes.hpp</a>	??
<a href="#">stdair/bom/TimePeriod.cpp</a>	??
<a href="#">stdair/bom/TimePeriod.hpp</a>	??
<a href="#">stdair/bom/TimePeriodKey.cpp</a>	??
<a href="#">stdair/bom/TimePeriodKey.hpp</a>	??
<a href="#">stdair/bom/TimePeriodTypes.hpp</a>	??
<a href="#">stdair/bom/TravelSolutionStruct.cpp</a>	??
<a href="#">stdair/bom/TravelSolutionStruct.hpp</a>	??
<a href="#">stdair/bom/TravelSolutionTypes.hpp</a>	??
<a href="#">stdair/bom/VirtualClassStruct.cpp</a>	??
<a href="#">stdair/bom/VirtualClassStruct.hpp</a>	??
<a href="#">stdair/bom/VirtualClassTypes.hpp</a>	??
<a href="#">stdair/bom/YieldFeatures.cpp</a>	??
<a href="#">stdair/bom/YieldFeatures.hpp</a>	??

<a href="#">stdair/bom/YieldFeaturesKey.cpp</a>	??
<a href="#">stdair/bom/YieldFeaturesKey.hpp</a>	??
<a href="#">stdair/bom/YieldFeaturesTypes.hpp</a>	??
<a href="#">stdair/bom/YieldStore.cpp</a>	??
<a href="#">stdair/bom/YieldStore.hpp</a>	??
<a href="#">stdair/bom/YieldStoreKey.cpp</a>	??
<a href="#">stdair/bom/YieldStoreKey.hpp</a>	??
<a href="#">stdair/bom/YieldStoreTypes.hpp</a>	??
<a href="#">stdair/command/CmdAbstract.cpp</a>	??
<a href="#">stdair/command/CmdAbstract.hpp</a>	??
<a href="#">stdair/command/CmdBomManager.cpp</a>	??
<a href="#">stdair/command/CmdBomManager.hpp</a>	??
<a href="#">stdair/command/CmdBomSerialiser.cpp</a>	??
<a href="#">stdair/command/CmdBomSerialiser.hpp</a>	??
<a href="#">stdair/command/CmdCloneBomManager.cpp</a>	??
<a href="#">stdair/command/CmdCloneBomManager.hpp</a>	??
<a href="#">stdair/command/DBManagerForAirlines.cpp</a>	??
<a href="#">stdair/command/DBManagerForAirlines.hpp</a>	??
<a href="#">stdair/dbadaptor/DbaAbstract.cpp</a>	??
<a href="#">stdair/dbadaptor/DbaAbstract.hpp</a>	??
<a href="#">stdair/dbadaptor/DbaAirline.cpp</a>	??
<a href="#">stdair/dbadaptor/DbaAirline.hpp</a>	??
<a href="#">stdair/factory/FacAbstract.cpp</a>	??
<a href="#">stdair/factory/FacAbstract.hpp</a>	??
<a href="#">stdair/factory/FacBom.hpp</a>	??
<a href="#">stdair/factory/FacBomManager.cpp</a>	??
<a href="#">stdair/factory/FacBomManager.hpp</a>	??
<a href="#">stdair/factory/FacCloneBom.hpp</a>	??
<a href="#">stdair/service/DBSessionManager.cpp</a>	??
<a href="#">stdair/service/DBSessionManager.hpp</a>	??
<a href="#">stdair/service/FacServiceAbstract.cpp</a>	??
<a href="#">stdair/service/FacServiceAbstract.hpp</a>	??

<a href="#">stdair/service/FacSTDAIRServiceContext.cpp</a>	??
<a href="#">stdair/service/FacSTDAIRServiceContext.hpp</a>	??
<a href="#">stdair/service/FacSupervisor.cpp</a>	??
<a href="#">stdair/service/FacSupervisor.hpp</a>	??
<a href="#">stdair/service/Logger.cpp</a>	??
<a href="#">stdair/service/Logger.hpp</a>	??
<a href="#">stdair/service/ServiceAbstract.cpp</a>	??
<a href="#">stdair/service/ServiceAbstract.hpp</a>	??
<a href="#">stdair/service/STDAIR_Service.cpp</a>	??
<a href="#">stdair/service/STDAIR_ServiceContext.cpp</a>	??
<a href="#">stdair/service/STDAIR_ServiceContext.hpp</a>	??
<a href="#">stdair/ui/cmdline/readline_autocomp.hpp</a>	??
<a href="#">stdair/ui/cmdline/SReadline.hpp</a> C++ wrapper around libreadline	??
<a href="#">test/stdair/MPBomRoot.cpp</a>	??
<a href="#">test/stdair/MPBomRoot.hpp</a>	??
<a href="#">test/stdair/MPInventory.cpp</a>	??
<a href="#">test/stdair/MPInventory.hpp</a>	??
<a href="#">test/stdair/StandardAirlineITTestSuite.cpp</a>	??
<a href="#">test/stdair/StdairTestLib.hpp</a>	??

## 31 Namespace Documentation

### 31.1 boost Namespace Reference

Forward declarations.

Namespaces

- [serialization](#)

#### 31.1.1 Detailed Description

Forward declarations.

### 31.2 boost::serialization Namespace Reference

### 31.3 bpt Namespace Reference



## Typedefs

- typedef char [ptree](#)

### 31.3.1 Typedef Documentation

#### 31.3.1.1 typedef char [bpt::ptree](#)

Definition at line 22 of file [BomINIImport.cpp](#).

## 31.4 soci Namespace Reference

### Classes

- struct [type\\_conversion< stdair::AirlineStruct >](#)

## 31.5 stdair Namespace Reference

Handle on the StdAir library context.

### Namespaces

- [LOG](#)

### Classes

- 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.*
- struct [BasChronometer](#)
- struct [BasDBParams](#)  
*Structure holding the parameters for connection to a database.*
- struct [BasFileMgr](#)
- struct [BasLogParams](#)  
*Structure holding parameters for logging.*
- 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](#)
- class [BookingClassListEmptyInNestingStructException](#)
- 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.*
- class [CmdAbstract](#)
- class [CmdBomManager](#)
- class [CmdBomSerialiser](#)
- class [CmdCloneBomManager](#)
- class [CodeConversionException](#)
- class [CodeDuplicationException](#)
- struct [ConfigHolderStruct](#)
- class [ConfigINIFile](#)
- struct [ContinuousAttributeLite](#)  
*Class modeling the distribution of values that can be taken by a continuous attribute.*
- struct [date\\_time\\_element](#)
- class [DatePeriod](#)  
*Class representing the actual attributes for a fare date-period.*
- struct [DatePeriodKey](#)  
*Key of date-period.*
- class [DbAbstract](#)
- class [DBManagerForAirlines](#)
- class [DBSessionManager](#)
- struct [DefaultDCPList](#)

- struct [DefaultDtdFratMap](#)
- struct [DefaultDtdProbMap](#)
- struct [DefaultMap](#)
- struct [DemandGenerationMethod](#)
  - Enumeration of demand (booking request) generation methods.*
- class [DictionaryManager](#)
  - Class wrapper of dictionary business methods.*
- class [DocumentNotFoundException](#)
- struct [DoWStruct](#)
- class [EventException](#)
- struct [EventStruct](#)
- struct [EventType](#)
- 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 [FacServiceAbstract](#)
- class [FacSTDAIRServiceContext](#)
  - Factory for [Bucket](#).*
- class [FacSupervisor](#)
- 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 [FFDisutilityFilePath](#)
- class [FileNotFoundException](#)
- 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 [ForecastingMethod](#)
- struct [FRAT5CurveHolderStruct](#)
- class [FRAT5FilePath](#)
- class [InputFilePath](#)
- class [Inventory](#)
  - Class representing the actual attributes for an airline inventory.*
- struct [InventoryKey](#)
  - Key of a given inventory, made of the airline code.*
- struct [JsonCommand](#)
  - Enumeration of json commands.*

- class [JSONString](#)  
*JSON-formatted string.*
- struct [KeyAbstract](#)  
*Base class for the keys of Business Object Model (BOM) layer.*
- class [KeyDuplicationException](#)
- class [KeyNotFoundException](#)
- 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 [Logger](#)
- class [MemoryAllocationException](#)
- 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 [NonInitialisedContainerException](#)
- class [NonInitialisedDBSessionManagerException](#)
- class [NonInitialisedLogServiceException](#)
- class [NonInitialisedRelationshipException](#)
- class [NonInitialisedServiceException](#)
- class [ObjectCreationDuplicationException](#)
- class [ObjectLinkingException](#)
- class [ObjectNotFoundException](#)
- class [ODFilePath](#)
- 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 [OptimisationMethod](#)
- struct [OptimisationNotificationStruct](#)
- struct [ParsedKey](#)
- class [ParserException](#)
- class [ParsingFileFailedException](#)
- struct [PartnershipTechnique](#)  
*Enumeration of partnership techniques.*
- struct [PassengerChoiceModel](#)
- struct [PassengerType](#)
- 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 [PreOptimisationMethod](#)
- struct [ProgressStatus](#)
- struct [ProgressStatusSet](#)

- struct [RandomGeneration](#)  
*Class holding a random generator.*
- struct [RMEventStruct](#)
- class [RootException](#)  
*Root of the stdair exceptions.*
- class [RootFilePath](#)  
*Root of the input and output files.*
- struct [SampleType](#)  
*Enumeration of BOM sample types.*
- class [ScheduleFilePath](#)
- 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 [SerialisationException](#)
- class [ServiceAbstract](#)
- struct [ServiceInitialisationType](#)  
*Enumeration of service initialisation types.*
- class [SimpleNestingStructException](#)
- class [SimpleNestingStructure](#)
- struct [SnapshotStruct](#)
- class [SQLDatabaseConnectionImpossibleException](#)
- class [SQLDatabaseException](#)
- class [STDAIR\\_Service](#)  
*Interface for the STDAIR Services.*
- class [STDAIR\\_ServiceContext](#)  
*Class holding the context of the Stdair services.*
- struct [StructAbstract](#)  
*Base class for the light structures.*
- 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 [UnconstrainingMethod](#)
- struct [VirtualClassStruct](#)
- class [YieldFeatures](#)  
*Class representing the actual attributes for a yield date-period.*
- struct [YieldFeaturesKey](#)  
*Key of date-period.*
- class [YieldRange](#)
- class [YieldStore](#)
- struct [YieldStoreKey](#)

## 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 struct [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 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 [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 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](#) [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](#) > [DTDFlatMap\\_T](#)
- typedef std::map< [FloatDuration\\_T](#), float > [DTDProbMap\\_T](#)
- typedef std::vector< [CensorshipFlag\\_T](#) > [CensorshipFlagList\\_T](#)
- typedef double [BookingRatio\\_T](#)
- typedef double [Yield\\_T](#)
- typedef unsigned int [YieldLevel\\_T](#)
- typedef std::map< [YieldLevel\\_T](#), [MeanStdDevPair\\_T](#) > [YieldLevelDemandMap\\_T](#)
- typedef std::pair< [Yield\\_T](#), [MeanStdDevPair\\_T](#) > [YieldDemandPair\\_T](#)
- typedef double [BidPrice\\_T](#)
- typedef std::vector< [BidPrice\\_T](#) > [BidPriceVector\\_T](#)
- typedef unsigned int [SeatIndex\\_T](#)
- typedef std::string [ControlMode\\_T](#)
- typedef double [OverbookingRate\\_T](#)
- typedef double [ProtectionLevel\\_T](#)
- typedef std::vector< double > [EmsrValueList\\_T](#)
- typedef std::vector< double > [BookingLimitVector\\_T](#)
- typedef std::vector< double > [ProtectionLevelVector\\_T](#)
- typedef boost::multi\_array< double, 2 > [SnapshotBlock\\_T](#)
- typedef [SnapshotBlock\\_T](#)::index\_range [SnapshotBlockRange\\_T](#)
- typedef [SnapshotBlock\\_T](#)::array\_view< 1 >::type [SegmentCabinDTDSnapshotView\\_T](#)
- typedef [SnapshotBlock\\_T](#)::array\_view< 2 >::type [SegmentCabinDTRangeSnapshotView\\_T](#)
- typedef [SnapshotBlock\\_T](#)::const\_array\_view< 1 >::type [ConstSegmentCabinDTDSnapshotView\\_T](#)
- typedef [SnapshotBlock\\_T](#)::const\_array\_view< 2 >::type [ConstSegmentCabinDTRangeSnapshotView\\_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 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, NUL↵  
L\_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\_REQU↵  
EST\_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 DTDFrattMap_T DEFAULT_DTD_FRAT5COEF_MAP`
- `const DTDProbMap_T DEFAULT_DTD_PROB_MAP`
- `const OnDStringList_T DEFAULT_OND_STRING_LIST`
- `const std::string DISPLAY_LEVEL_STRING_ARRAY [51]`
- `const std::string DEFAULT_KEY_FLD_DELIMITER`
- `const std::string DEFAULT_KEY_SUB_FLD_DELIMITER`
- `const boost::char_separator< char > DEFAULT_KEY_TOKEN_DELIMITER`
- `const Distance_T DEFAULT_DISTANCE_VALUE`
- `const ClassCode_T DEFAULT_CLOSED_CLASS_CODE`
- `const NbOfBookings_T DEFAULT_CLASS_NB_OF_BOOKINGS`
- `const NbOfBookings_T DEFAULT_CLASS_TOTAL_NB_OF_BOOKINGS`
- `const NbOfBookings_T DEFAULT_CLASS_UNCONSTRAINED_DEMAND`
- `const NbOfBookings_T DEFAULT_CLASS_REMAINING_DEMAND_MEAN`
- `const NbOfBookings_T DEFAULT_CLASS_REMAINING_DEMAND_STANDARD_DEVIATION`
- `const NbOfCancellations_T DEFAULT_CLASS_NB_OF_CANCELLATIONS`
- `const NbOfNoShows_T DEFAULT_CLASS_NB_OF_NOSHOWS`

- const CabinCapacity\_T DEFAULT\_CABIN\_CAPACITY
- const CommittedSpace\_T DEFAULT\_COMMITTED\_SPACE
- const BlockSpace\_T DEFAULT\_BLOCK\_SPACE
- const Availability\_T DEFAULT\_NULL\_AVAILABILITY
- const Availability\_T DEFAULT\_AVAILABILITY
- const CensorshipFlag\_T DEFAULT\_CLASS\_CENSORSHIPFLAG
- const BookingLimit\_T DEFAULT\_CLASS\_BOOKING\_LIMIT
- const AuthorizationLevel\_T DEFAULT\_CLASS\_AUTHORIZATION\_LEVEL
- const AuthorizationLevel\_T DEFAULT\_CLASS\_MAX\_AUTHORIZATION\_LEVEL
- const AuthorizationLevel\_T DEFAULT\_CLASS\_MIN\_AUTHORIZATION\_LEVEL
- const OverbookingRate\_T DEFAULT\_CLASS\_OVERBOOKING\_RATE
- const Fare\_T DEFAULT\_FARE\_VALUE
- const Revenue\_T DEFAULT\_REVENUE\_VALUE
- const PriceCurrency\_T DEFAULT\_CURRENCY
- const Percentage\_T DEFAULT\_LOAD\_FACTOR\_VALUE
- const DayDuration\_T DEFAULT\_DAY\_DURATION
- const double DEFAULT\_EPSILON\_VALUE
- const AirportCode\_T AIRPORT\_LHR
- const AirportCode\_T AIRPORT\_SYD
- const CityCode\_T POS\_LHR
- const DayDuration\_T NO\_ADVANCE\_PURCHASE
- const SaturdayStay\_T SATURDAY\_STAY
- const SaturdayStay\_T NO\_SATURDAY\_STAY
- const ChangeFees\_T CHANGE\_FEES
- const ChangeFees\_T NO\_CHANGE\_FEES
- const NonRefundable\_T NON\_REFUNDABLE
- const NonRefundable\_T NO\_NON\_REFUNDABLE
- const DayDuration\_T NO\_STAY\_DURATION
- const CabinCode\_T CABIN\_Y
- const AirlineCode\_T AIRLINE\_CODE\_BA
- const ClassCode\_T CLASS\_CODE\_Y
- const ClassCode\_T CLASS\_CODE\_Q
- const AirportCode\_T AIRPORT\_SIN
- const AirportCode\_T AIRPORT\_BKK
- const CityCode\_T POS\_SIN
- const CabinCode\_T CABIN\_ECO
- const FrequentFlyer\_T FREQUENT\_FLYER\_MEMBER
- const 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 NbOfFlightDates\_T DEFAULT\_NB\_OF\_FLIGHTDATES
- const unsigned int DEFAULT\_FLIGHT\_SPEED
- const BookingRatio\_T DEFAULT\_OND\_BOOKING\_RATE
- const Count\_T SECONDS\_IN\_ONE\_DAY
- const Count\_T MILLISECONDS\_IN\_ONE\_SECOND
- const Date\_T DEFAULT\_DATE
- const DateTime\_T DEFAULT\_DATETIME
- const Duration\_T DEFAULT\_EPSILON\_DURATION
- const RandomSeed\_T DEFAULT\_RANDOM\_SEED
- const Duration\_T NULL\_BOOST\_TIME\_DURATION
- const 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 ChannelLabel\_T DEFAULT\_CHANNEL
- const unsigned int DEFAULT\_NUMBER\_OF\_SUBDIVISIONS
- const AirlineCode\_T DEFAULT\_AIRLINE\_CODE
- const AirlineCode\_T DEFAULT\_NULL\_AIRLINE\_CODE
- const FlightNumber\_T DEFAULT\_FLIGHT\_NUMBER
- const 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 BidPrice\_T DEFAULT\_BID\_PRICE
- const unsigned short MAXIMAL\_NUMBER\_OF\_LEGS\_IN\_FLIGHT
- const unsigned short MAXIMAL\_NUMBER\_OF\_SEGMENTS\_IN\_OND
- const Availability\_T MAXIMAL\_AVAILABILITY
- const SeatIndex\_T DEFAULT\_SEAT\_INDEX
- const 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 DatePeriod\_T BOOST\_DEFAULT\_DATE\_PERIOD
- const DOW\_String\_T DEFAULT\_DOW\_STRING
- const DateOffset\_T DEFAULT\_DATE\_OFFSET
- const PartySize\_T DEFAULT\_PARTY\_SIZE
- const DayDuration\_T DEFAULT\_STAY\_DURATION
- const WTP\_T DEFAULT\_WTP
- const CityCode\_T DEFAULT\_POS
- const Date\_T DEFAULT\_PREFERRED\_DEPARTURE\_DATE
- const Duration\_T DEFAULT\_PREFERRED\_DEPARTURE\_TIME
- const DateOffset\_T DEFAULT\_ADVANCE\_PURCHASE
- const Date\_T DEFAULT\_REQUEST\_DATE
- const Duration\_T DEFAULT\_REQUEST\_TIME
- const DateTime\_T DEFAULT\_REQUEST\_DATE\_TIME
- const CabinCode\_T DEFAULT\_PREFERRED\_CABIN
- const ChannelLabel\_T CHANNEL\_DN
- const ChannelLabel\_T CHANNEL\_IN
- const TripType\_T TRIP\_TYPE\_ONE\_WAY
- const TripType\_T TRIP\_TYPE\_ROUND\_TRIP
- const TripType\_T TRIP\_TYPE\_INBOUND
- const TripType\_T TRIP\_TYPE\_OUTBOUND
- const FrequentFlyer\_T DEFAULT\_FF\_TIER
- const PriceValue\_T DEFAULT\_VALUE\_OF\_TIME

- const [IntDuration\\_T](#) HOUR\_CONVERTED\_IN\_SECONDS
- const [Duration\\_T](#) DEFAULT\_MINIMAL\_CONNECTION\_TIME
- const [Duration\\_T](#) DEFAULT\_MAXIMAL\_CONNECTION\_TIME
- const [FlightPathCode\\_T](#) DEFAULT\_FLIGHTPATH\_CODE
- const [Availability\\_T](#) DEFAULT\_CLASS\_AVAILABILITY
- const [AvailabilityStatus\\_T](#) DEFAULT\_AVAILABILITY\_STATUS
- const unsigned short DEFAULT\_NUMBER\_OF\_REQUIRED\_SEATS
- const [MatchingIndicator\\_T](#) DEFAULT\_MATCHING\_INDICATOR
- const [AirlineCode\\_T](#) DEFAULT\_DICO\_STUDIED\_AIRLINE
- const [Yield\\_T](#) DEFAULT\_YIELD\_VALUE
- const [Yield\\_T](#) DEFAULT\_YIELD\_MAX\_VALUE

### 31.5.1 Detailed Description

Handle on the StdAir library context.

#### Author

Anh Quan Nguyen [quannaus@users.sourceforge.net](mailto: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.

### 31.5.2 Typedef Documentation

#### 31.5.2.1 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](#).

#### 31.5.2.2 typedef date\_time\_element<0, 59> stdair::minute\_t

Definition at line 62 of file [BasParserHelperTypes.hpp](#).

#### 31.5.2.3 typedef date\_time\_element<0, 59> stdair::second\_t

Definition at line 63 of file [BasParserHelperTypes.hpp](#).

#### 31.5.2.4 typedef date\_time\_element<1900, 2100> stdair::year\_t

Definition at line 64 of file [BasParserHelperTypes.hpp](#).

#### 31.5.2.5 typedef date\_time\_element<1, 12> stdair::month\_t

Definition at line 65 of file [BasParserHelperTypes.hpp](#).

#### 31.5.2.6 typedef date\_time\_element<1, 31> stdair::day\_t

Definition at line 66 of file [BasParserHelperTypes.hpp](#).

#### 31.5.2.7 typedef std::istreambuf\_iterator<char> stdair::base\_iterator\_t

Definition at line 26 of file [BasParserTypes.hpp](#).

31.5.2.8 `typedef boost::spirit::multi_pass<base_iterator_t> stdair::iterator_t`

Definition at line 27 of file [BasParserTypes.hpp](#).

31.5.2.9 `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](#).

31.5.2.10 `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](#).

31.5.2.11 `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](#).

31.5.2.12 `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](#).

31.5.2.13 `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](#).

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

Definition at line 48 of file [BasParserTypes.hpp](#).

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

Definition at line 49 of file [BasParserTypes.hpp](#).

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

Definition at line 50 of file [BasParserTypes.hpp](#).

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

Definition at line 51 of file [BasParserTypes.hpp](#).

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

Definition at line 52 of file [BasParserTypes.hpp](#).

31.5.2.19 `typedef unsigned short stdair::DictionaryKey_T`

Dictionary key.

Definition at line 17 of file [DictionaryManager.hpp](#).

31.5.2.20 `typedef std::list<AirlineClassList*> stdair::AirlineClassListList_T`

Define the segment-features list.

Definition at line 17 of file [AirlineClassListTypes.hpp](#).

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

Define the segment-features map.

Definition at line 23 of file [AirlineClassListTypes.hpp](#).

31.5.2.22 `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](#).

31.5.2.23 `typedef std::list<AirlineClassListWithKey_T> stdair::AirlineClassListDetailedList_T`

Definition at line 27 of file [AirlineClassListTypes.hpp](#).

31.5.2.24 `typedef std::list<AirlineFeature*> stdair::AirlineFeatureList_T`

Define the airline feature list.

Definition at line 17 of file [AirlineFeatureTypes.hpp](#).

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

Define the airline feature map.

Definition at line 23 of file [AirlineFeatureTypes.hpp](#).

31.5.2.26 `typedef std::list<AirportPair*> stdair::AirportPairList_T`

Define the airport-pair list.

Definition at line 17 of file [AirportPairTypes.hpp](#).

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

Define the airport-pair map.

Definition at line 23 of file [AirportPairTypes.hpp](#).

31.5.2.28 `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](#).

31.5.2.29 `typedef std::list<AirportPairWithKey_T> stdair::AirportPairDetailedList_T`

Definition at line 27 of file [AirportPairTypes.hpp](#).

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

Definition at line 63 of file [BomAbstract.hpp](#).

31.5.2.31 `typedef struct BomID< BookingClass > stdair::BookingClassID_T`

Define the booking class ID.

Definition at line 21 of file [BomIDTypes.hpp](#).

31.5.2.32 `typedef std::list<BookingClassID_T> stdair::BookingClassIDList_T`

Define the list of booking class ID's.

Definition at line 24 of file [BomIDTypes.hpp](#).

**31.5.2.33** `typedef boost::tokenizer< boost::char_separator< char > > stdair::Tokeniser_T`

Boost Tokeniser.

Definition at line 28 of file [BomKeyManager.cpp](#).

**31.5.2.34** `typedef std::list<BookingClass*> stdair::BookingClassList_T`

Define the booking class list.

Definition at line 17 of file [BookingClassTypes.hpp](#).

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

Define the booking class map.

Definition at line 23 of file [BookingClassTypes.hpp](#).

**31.5.2.36** `typedef boost::shared_ptr<BookingRequestStruct> stdair::BookingRequestPtr_T`

Define the smart pointer to a booking request.

Definition at line 14 of file [BookingRequestTypes.hpp](#).

**31.5.2.37** `typedef std::string stdair::DemandGeneratorKey_T`

Define the hash key for the demand generator.

Definition at line 21 of file [BookingRequestTypes.hpp](#).

**31.5.2.38** `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](#).

**31.5.2.39** `typedef std::list<BreakPointStruct> stdair::BreakPointList_T`

Define the list of Break Points.

Definition at line 23 of file [BreakPointTypes.hpp](#).

**31.5.2.40** `typedef std::list<Bucket*> stdair::BucketList_T`

Define the bucket list.

Definition at line 17 of file [BucketTypes.hpp](#).

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

Define the bucket map.

Definition at line 23 of file [BucketTypes.hpp](#).

**31.5.2.42** `typedef boost::shared_ptr<CancellationStruct> stdair::CancellationPtr_T`

Define the smart pointer to a cancellation .

Definition at line 14 of file [CancellationTypes.hpp](#).

**31.5.2.43** `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](#).

**31.5.2.44** `typedef std::list<DatePeriod*> stdair::DatePeriodList_T`

Define the date-period list.

Definition at line 17 of file [DatePeriodTypes.hpp](#).

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

Define the date-period map.

Definition at line 23 of file [DatePeriodTypes.hpp](#).

**31.5.2.46** `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](#).

**31.5.2.47** `typedef std::list<DatePeriodWithKey_T> stdair::DatePeriodDetailedList_T`

Definition at line 27 of file [DatePeriodTypes.hpp](#).

**31.5.2.48** `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](#).

**31.5.2.49** `typedef std::map<const LongDuration_T, EventStruct> stdair::EventList_T`

Define a list of events.

Definition at line 32 of file [EventTypes.hpp](#).

**31.5.2.50** `typedef std::list<FareFamily*> stdair::FareFamilyList_T`

Define the fare family list.

Definition at line 17 of file [FareFamilyTypes.hpp](#).

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

Define the fare family map.

Definition at line 23 of file [FareFamilyTypes.hpp](#).

**31.5.2.52** `typedef std::list<FareFeatures*> stdair::FareFeaturesList_T`

Define the date-period list.

Definition at line 17 of file [FareFeaturesTypes.hpp](#).

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

Define the date-period map.

Definition at line 23 of file [FareFeaturesTypes.hpp](#).

**31.5.2.54** `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](#).

**31.5.2.55** `typedef std::list<FareFeaturesWithKey_T> stdair::FareFeaturesDetailedList_T`

Definition at line 27 of file [FareFeaturesTypes.hpp](#).

**31.5.2.56 typedef std::list<FareOptionStruct> stdair::FareOptionList\_T**

Define the booking class list.

Definition at line 18 of file [FareOptionTypes.hpp](#).

**31.5.2.57 typedef std::map<const std::string, FFDisutilityCurve\_T> stdair::FFDisutilityCurveHolder\_T**

Definition at line 16 of file [FFDisutilityCurveHolderStruct.hpp](#).

**31.5.2.58 typedef std::list<FlightDate\*> stdair::FlightDateList\_T**

Define the flight-date list.

Definition at line 17 of file [FlightDateTypes.hpp](#).

**31.5.2.59 typedef std::map<const MapKey\_T, FlightDate\*> stdair::FlightDateMap\_T**

Define the flight-date map.

Definition at line 24 of file [FlightDateTypes.hpp](#).

**31.5.2.60 typedef std::list<FlightPeriod\*> stdair::FlightPeriodList\_T**

Define the flight-period list.

Definition at line 17 of file [FlightPeriodTypes.hpp](#).

**31.5.2.61 typedef std::map<const MapKey\_T, FlightPeriod\*> stdair::FlightPeriodMap\_T**

Define the flight-period map.

Definition at line 23 of file [FlightPeriodTypes.hpp](#).

**31.5.2.62 typedef std::map<const std::string, FRAT5Curve\_T> stdair::FRAT5CurveHolder\_T**

Definition at line 16 of file [FRAT5CurveHolderStruct.hpp](#).

**31.5.2.63 typedef std::list<Inventory\*> stdair::InventoryList\_T**

Define the [Inventory](#) list.

Definition at line 17 of file [InventoryTypes.hpp](#).

**31.5.2.64 typedef std::map<const MapKey\_T, Inventory\*> stdair::InventoryMap\_T**

Define the [Inventory](#) map.

Definition at line 23 of file [InventoryTypes.hpp](#).

**31.5.2.65 typedef std::string stdair::MapKey\_T**

Key of a STL map.

Definition at line 15 of file [key\\_types.hpp](#).

**31.5.2.66 typedef std::list<std::string> stdair::KeyList\_T**

List of keys.

Definition at line 18 of file [key\\_types.hpp](#).

**31.5.2.67 typedef std::list<LegCabin\*> stdair::LegCabinList\_T**

Define the leg-cabin list.

Definition at line 17 of file [LegCabinTypes.hpp](#).

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

Define the leg-cabin map.

Definition at line 23 of file [LegCabinTypes.hpp](#).

**31.5.2.69** `typedef std::list<LegDate*> stdair::LegDateList_T`

Define the leg-date list.

Definition at line 17 of file [LegDateTypes.hpp](#).

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

Define the leg-date map.

Definition at line 23 of file [LegDateTypes.hpp](#).

**31.5.2.71** `typedef std::list<NestingNode*> stdair::NestingNodeList_T`

Define the fare family list.

Definition at line 17 of file [NestingNodeTypes.hpp](#).

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

Define the fare family map.

Definition at line 23 of file [NestingNodeTypes.hpp](#).

**31.5.2.73** `typedef std::list<OnDDate*> stdair::OnDDateList_T`

Define the O&D date list.

Definition at line 19 of file [OnDDateTypes.hpp](#).

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

Define the OnD date map.

Definition at line 25 of file [OnDDateTypes.hpp](#).

**31.5.2.75** `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](#).

**31.5.2.76** `typedef std::map<std::string, YieldDemandPair_T> stdair::StringDemandStructMap_T`

Definition at line 33 of file [OnDDateTypes.hpp](#).

**31.5.2.77** `typedef std::map<std::string, CabinClassPairList_T> stdair::StringCabinClassPairListMap_T`

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

Definition at line 41 of file [OnDDateTypes.hpp](#).

**31.5.2.78** `typedef std::pair<std::string, CabinClassPairList_T> stdair::StringCabinClassPair_T`

Definition at line 42 of file [OnDDateTypes.hpp](#).



**31.5.2.79** `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](#).

**31.5.2.80** `typedef std::pair<CabinCode_T, WTPDemandPair_T> stdair::CabinForecastPair_T`

Definition at line 49 of file [OnDDateTypes.hpp](#).

**31.5.2.81** `typedef boost::shared_ptr<OptimisationNotificationStruct> stdair::OptimisationNotificationPtr_T`

Define the smart pointer to a optimisation notification.

Definition at line 14 of file [OptimisationNotificationTypes.hpp](#).

**31.5.2.82** `typedef std::list<Policy*> stdair::PolicyList_T`

Define the fare family list.

Definition at line 17 of file [PolicyTypes.hpp](#).

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

Define the fare family map.

Definition at line 23 of file [PolicyTypes.hpp](#).

**31.5.2.84** `typedef std::list<PosChannel*> stdair::PosChannelList_T`

Define the fare-point\_of\_sale list.

Definition at line 17 of file [PosChannelTypes.hpp](#).

**31.5.2.85** `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](#).

**31.5.2.86** `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](#).

**31.5.2.87** `typedef std::list<PosChannelWithKey_T> stdair::PosChannelDetailedList_T`

Definition at line 27 of file [PosChannelTypes.hpp](#).

**31.5.2.88** `typedef boost::shared_ptr<RMEventStruct> stdair::RMEventPtr_T`

Define the smart pointer to a RM event .

Definition at line 16 of file [RMEventTypes.hpp](#).

**31.5.2.89** `typedef std::list<RMEventStruct> stdair::RMEventList_T`

Define the list of RM events.

Definition at line 23 of file [RMEventTypes.hpp](#).

**31.5.2.90** `typedef std::list<SegmentCabin*> stdair::SegmentCabinList_T`

Define the segment-cabin list.

Definition at line 17 of file [SegmentCabinTypes.hpp](#).

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

Define the segment-cabin map.

Definition at line 23 of file [SegmentCabinTypes.hpp](#).

31.5.2.92 `typedef std::list<std::string> stdair::RoutingLegKeyList_T`

Definition at line 27 of file [SegmentDate.hpp](#).

31.5.2.93 `typedef std::list<SegmentDate*> stdair::SegmentDateList_T`

Define the segment-date list.

Definition at line 17 of file [SegmentDateTypes.hpp](#).

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

Define the segment-date map.

Definition at line 23 of file [SegmentDateTypes.hpp](#).

31.5.2.95 `typedef std::list<SegmentPeriod*> stdair::SegmentPeriodList_T`

Define the segment-period list.

Definition at line 17 of file [SegmentPeriodTypes.hpp](#).

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

Define the segment-period map.

Definition at line 23 of file [SegmentPeriodTypes.hpp](#).

31.5.2.97 `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](#).

31.5.2.98 `typedef std::list<SegmentPeriodWithKey_T> stdair::SegmentPeriodDetailedList_T`

Definition at line 27 of file [SegmentPeriodTypes.hpp](#).

31.5.2.99 `typedef std::list<SegmentSnapshotTable*> stdair::SegmentSnapshotTableList_T`

Define the guillotine-block list.

Definition at line 20 of file [SegmentSnapshotTableTypes.hpp](#).

31.5.2.100 `typedef std::map<const MapKey_T, SegmentSnapshotTable*> stdair::SegmentSnapshotTable↵  
Map_T`

Define the guillotine-block map.

Definition at line 27 of file [SegmentSnapshotTableTypes.hpp](#).

31.5.2.101 `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](#).

31.5.2.102 `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](#).

31.5.2.103 `typedef std::list<SimpleNestingStructure*> stdair::SimpleNestingStructureList_T`

Define the fare family list.

Definition at line 17 of file [SimpleNestingStructureTypes.hpp](#).

31.5.2.104 `typedef std::map<const MapKey_T, SimpleNestingStructure*> stdair::SimpleNestingStructure↔  
Map_T`

Define the fare family map.

Definition at line 23 of file [SimpleNestingStructureTypes.hpp](#).

31.5.2.105 `typedef boost::shared_ptr<SnapshotStruct> stdair::SnapshotPtr_T`

Define the smart pointer to a snapshot .

Definition at line 14 of file [SnapshotTypes.hpp](#).

31.5.2.106 `typedef std::list<TimePeriod*> stdair::TimePeriodList_T`

Define the time-period list.

Definition at line 17 of file [TimePeriodTypes.hpp](#).

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

Define the time-period map.

Definition at line 23 of file [TimePeriodTypes.hpp](#).

31.5.2.108 `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](#).

31.5.2.109 `typedef std::list<TimePeriodWithKey_T> stdair::TimePeriodDetailedList_T`

Definition at line 27 of file [TimePeriodTypes.hpp](#).

31.5.2.110 `typedef std::list<TravelSolutionStruct> stdair::TravelSolutionList_T`

Define the booking class list.

Definition at line 20 of file [TravelSolutionTypes.hpp](#).

31.5.2.111 `typedef KeyList_T stdair::SegmentPath_T`

Define the segment path key.

Definition at line 26 of file [TravelSolutionTypes.hpp](#).

31.5.2.112 `typedef std::list<SegmentPath_T> stdair::SegmentPathList_T`

Define the list of segment paths.

Definition at line 29 of file [TravelSolutionTypes.hpp](#).

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

Define booking class - availability map.

Definition at line 32 of file [TravelSolutionTypes.hpp](#).

31.5.2.114 `typedef std::list<ClassAvailabilityMap_T> stdair::ClassAvailabilityMapHolder_T`

Define list of booking class - availability maps.

Definition at line 35 of file [TravelSolutionTypes.hpp](#).

31.5.2.115 `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](#).

31.5.2.116 `typedef std::list<ClassObjectIDMap_T> stdair::ClassObjectIDMapHolder_T`

Define list of booking class - object ID maps.

Definition at line 41 of file [TravelSolutionTypes.hpp](#).

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

Define booking class - yield map.

Definition at line 44 of file [TravelSolutionTypes.hpp](#).

31.5.2.118 `typedef std::list<ClassYieldMap_T> stdair::ClassYieldMapHolder_T`

Define list of booking class - yield maps.

Definition at line 47 of file [TravelSolutionTypes.hpp](#).

31.5.2.119 `typedef std::list<BidPriceVector_T> stdair::BidPriceVectorHolder_T`

Define list of bid price vectors.

Definition at line 50 of file [TravelSolutionTypes.hpp](#).

31.5.2.120 `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](#).

31.5.2.121 `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](#).

31.5.2.122 `typedef std::list<VirtualClassStruct> stdair::VirtualClassList_T`

Define the booking class list.

Definition at line 17 of file [VirtualClassTypes.hpp](#).

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

Define the booking class map.

Definition at line 23 of file [VirtualClassTypes.hpp](#).

**31.5.2.124** `typedef std::list<YieldFeatures*> stdair::YieldFeaturesList_T`

Define the date-period list.

Definition at line 17 of file [YieldFeaturesTypes.hpp](#).

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

Define the date-period map.

Definition at line 23 of file [YieldFeaturesTypes.hpp](#).

**31.5.2.126** `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](#).

**31.5.2.127** `typedef std::list<YieldFeaturesWithKey_T> stdair::YieldFeaturesDetailedList_T`

Definition at line 27 of file [YieldFeaturesTypes.hpp](#).

**31.5.2.128** `typedef std::list<YieldStore*> stdair::YieldStoreList_T`

Define the [Inventory](#) list.

Definition at line 17 of file [YieldStoreTypes.hpp](#).

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

Define the [Inventory](#) map.

Definition at line 23 of file [YieldStoreTypes.hpp](#).

**31.5.2.130** `typedef std::string stdair::LocationCode_T`

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

Definition at line 16 of file [stdair\\_basic\\_types.hpp](#).

**31.5.2.131** `typedef unsigned long int stdair::Distance_T`

Define a distance (kilometers).

Definition at line 19 of file [stdair\\_basic\\_types.hpp](#).

**31.5.2.132** `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](#).

**31.5.2.133** `typedef LocationCode_T stdair::CityCode_T`

City code

Definition at line 25 of file [stdair\\_basic\\_types.hpp](#).

**31.5.2.134** `typedef std::string stdair::KeyDescription_T`

Define the key description.

Definition at line 28 of file [stdair\\_basic\\_types.hpp](#).

**31.5.2.135** `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](#).

#### 31.5.2.136 typedef unsigned short stdair::FlightNumber\_T

Define the type for flight numbers.

Definition at line 34 of file [stdair\\_basic\\_types.hpp](#).

#### 31.5.2.137 typedef unsigned short stdair::TableID\_T

Define the type for data table numbers.

Definition at line 37 of file [stdair\\_basic\\_types.hpp](#).

#### 31.5.2.138 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](#).

#### 31.5.2.139 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](#).

#### 31.5.2.140 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](#).

#### 31.5.2.141 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](#).

#### 31.5.2.142 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](#).

#### 31.5.2.143 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](#).

#### 31.5.2.144 typedef unsigned long stdair::Identity\_T

Define a identity number.

Definition at line 60 of file [stdair\\_basic\\_types.hpp](#).

#### 31.5.2.145 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](#).

#### 31.5.2.146 typedef double stdair::MonetaryValue\_T

Monetary value

Definition at line 67 of file [stdair\\_basic\\_types.hpp](#).

**31.5.2.147 typedef double stdair::RealNumber\_T**

Real number

Definition at line 70 of file [stdair\\_basic\\_types.hpp](#).

**31.5.2.148 typedef double stdair::Percentage\_T**

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

Definition at line 73 of file [stdair\\_basic\\_types.hpp](#).

**31.5.2.149 typedef double stdair::PriceValue\_T**

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

Definition at line 76 of file [stdair\\_basic\\_types.hpp](#).

**31.5.2.150 typedef double stdair::YieldValue\_T**

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

Definition at line 79 of file [stdair\\_basic\\_types.hpp](#).

**31.5.2.151 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](#).

**31.5.2.152 typedef double stdair::Revenue\_T**

Define an amount of revenue.

Define the revenue of a policy

Definition at line 85 of file [stdair\\_basic\\_types.hpp](#).

**31.5.2.153 typedef double stdair::Multiplier\_T**

Define the name of a multiplier.

Definition at line 88 of file [stdair\\_basic\\_types.hpp](#).

**31.5.2.154 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](#).

**31.5.2.155 typedef unsigned int stdair::Count\_T**

Count

Definition at line 95 of file [stdair\\_basic\\_types.hpp](#).

**31.5.2.156 typedef short stdair::PartySize\_T**

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

Definition at line 98 of file [stdair\\_basic\\_types.hpp](#).

**31.5.2.157 typedef double stdair::NbOfRequests\_T**

Define a number of requests.

Definition at line 101 of file [stdair\\_basic\\_types.hpp](#).

31.5.2.158 `typedef NbOfRequests_T stdair::NbOfBookings_T`

Define a number of bookings.

Definition at line 104 of file [stdair\\_basic\\_types.hpp](#).

31.5.2.159 `typedef NbOfRequests_T stdair::NbOfCancellations_T`

Define a number of cancellations.

Define a number of cancellations (travellers).

Definition at line 107 of file [stdair\\_basic\\_types.hpp](#).

31.5.2.160 `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](#).

31.5.2.161 `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](#).

31.5.2.162 `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](#).

31.5.2.163 `typedef unsigned short stdair::NbOfAirlines_T`

Define a number of airlines (in a path).

Definition at line 120 of file [stdair\\_basic\\_types.hpp](#).

31.5.2.164 `typedef double stdair::Availability_T`

Define an availability.

Definition at line 123 of file [stdair\\_basic\\_types.hpp](#).

31.5.2.165 `typedef double stdair::Fare_T`

Define the price of a travel solution.

Definition at line 126 of file [stdair\\_basic\\_types.hpp](#).

31.5.2.166 `typedef bool stdair::Flag_T`

Define the censorship flag.

Definition at line 129 of file [stdair\\_basic\\_types.hpp](#).

31.5.2.167 `typedef unsigned int stdair::UnsignedIndex_T`

Define the unsigned index type.

Definition at line 132 of file [stdair\\_basic\\_types.hpp](#).

31.5.2.168 `typedef unsigned int stdair::NbOfClasses_T`

Define the number of booking classes.



Definition at line 135 of file [stdair\\_basic\\_types.hpp](#).

**31.5.2.169** `typedef unsigned int stdair::NbOfFareFamilies_T`

Define the number of fare families.

Definition at line 138 of file [stdair\\_basic\\_types.hpp](#).

**31.5.2.170** `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](#).

**31.5.2.171** `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](#).

**31.5.2.172** `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](#).

**31.5.2.173** `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](#).

**31.5.2.174** `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](#).

**31.5.2.175** `typedef boost::posix_time::time_duration stdair::Time_T`

Time

Definition at line 23 of file [stdair\\_date\\_time\\_types.hpp](#).

**31.5.2.176** `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](#).

**31.5.2.177** `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](#).

**31.5.2.178** `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](#).

**31.5.2.179** `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](#).

**31.5.2.180** `typedef int stdair::DayDuration_T`

Define a duration in number of days.

Definition at line 38 of file [stdair\\_date\\_time\\_types.hpp](#).

**31.5.2.181** `typedef bool stdair::SaturdayStay_T`

Define the Saturday stay status of a travel.

Define the saturday stay of a tickets.

Definition at line 41 of file [stdair\\_date\\_time\\_types.hpp](#).

**31.5.2.182** `typedef long int stdair::IntDuration_T`

Time duration in (integer) number of seconds

Definition at line 44 of file [stdair\\_date\\_time\\_types.hpp](#).

**31.5.2.183** `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](#).

**31.5.2.184** `typedef float stdair::FloatDuration_T`

Duration in (float) number of time units

Definition at line 50 of file [stdair\\_date\\_time\\_types.hpp](#).

**31.5.2.185** `typedef soci::session stdair::DBSession_T`

Database session handler.

Definition at line 20 of file [stdair\\_db.hpp](#).

**31.5.2.186** `typedef soci::statement stdair::DBRequestStatement_T`

Database request statement handler.

Definition at line 23 of file [stdair\\_db.hpp](#).

**31.5.2.187** `typedef std::string stdair::DBConnectionName_T`

Define the name of an database connection.

Definition at line 26 of file [stdair\\_db.hpp](#).

**31.5.2.188** `typedef bool stdair::ChangeFees_T`

Define the availability option allowing the ticket change.

Definition at line 29 of file [stdair\\_demand\\_types.hpp](#).

**31.5.2.189** `typedef bool stdair::NonRefundable_T`

Define the refundable availability of a tickets.

Definition at line 32 of file [stdair\\_demand\\_types.hpp](#).

**31.5.2.190 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](#).

**31.5.2.191 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](#).

**31.5.2.192 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](#).

**31.5.2.193 typedef double stdair::Disutility\_T**

Define the disutility of restriction.

Definition at line 50 of file [stdair\\_demand\\_types.hpp](#).

**31.5.2.194 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](#).

**31.5.2.195 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](#).

**31.5.2.196 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](#).

**31.5.2.197 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](#).

**31.5.2.198 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](#).

**31.5.2.199 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](#).

**31.5.2.200 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](#).

**31.5.2.201** `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](#).

**31.5.2.202** `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](#).

**31.5.2.203** `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](#).

**31.5.2.204** `typedef NbOfRequests_T stdair::NbOfNoShows_T`

Define a number of no-shows.

Definition at line 88 of file [stdair\\_demand\\_types.hpp](#).

**31.5.2.205** `typedef double stdair::MatchingIndicator_T`

Define a indicator of demand to class matching.

Definition at line 91 of file [stdair\\_demand\\_types.hpp](#).

**31.5.2.206** `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](#).

**31.5.2.207** `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](#).

**31.5.2.208** `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](#).

**31.5.2.209** `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](#).

**31.5.2.210** `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](#).

**31.5.2.211** `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](#).

**31.5.2.212** `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](#).

**31.5.2.213** `typedef double stdair::ProportionFactor_T`

Define the forecast booking requests proportion.

Definition at line 116 of file [stdair\\_demand\\_types.hpp](#).

**31.5.2.214** `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](#).

**31.5.2.215** `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](#).

**31.5.2.216** `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](#).

**31.5.2.217** `typedef std::string stdair::EventName_T`

Define the name of an event.

Definition at line 14 of file [stdair\\_event\\_types.hpp](#).

**31.5.2.218** `typedef double stdair::NbOfEvents_T`

Define a number of events.

Definition at line 17 of file [stdair\\_event\\_types.hpp](#).

**31.5.2.219** `typedef std::string stdair::EventGeneratorKey_T`

Define a key string of an event generator.

Definition at line 20 of file [stdair\\_event\\_types.hpp](#).

**31.5.2.220** `typedef double stdair::NbOfFareRules_T`

Define a number of fare rules.

Definition at line 12 of file [stdair\\_fare\\_types.hpp](#).

**31.5.2.221** `typedef std::string stdair::NetworkID_T`

Define the type for network ID.

Definition at line 23 of file [stdair\\_inventory\\_types.hpp](#).

**31.5.2.222** `typedef std::vector<AirlineCode_T> stdair::AirlineCodeList_T`

Define a list of airline code.

Definition at line 26 of file [stdair\\_inventory\\_types.hpp](#).

**31.5.2.223** `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](#).

**31.5.2.224** `typedef std::vector<ClassCode_T> stdair::ClassCodeList_T`

Define a list of class code.

Definition at line 32 of file [stdair\\_inventory\\_types.hpp](#).

**31.5.2.225** `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](#).

**31.5.2.226** `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](#).

**31.5.2.227** `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](#).

**31.5.2.228** `typedef std::string stdair::CurveKey_T`

Curve key for FRAT5 or FF Disutility.

Definition at line 47 of file [stdair\\_inventory\\_types.hpp](#).

**31.5.2.229** `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](#).

**31.5.2.230** `typedef double stdair::NbOfFlightDates_T`

Define a number of flight dates.

Definition at line 54 of file [stdair\\_inventory\\_types.hpp](#).

**31.5.2.231** `typedef double stdair::CommittedSpace_T`

Define the committed space of a cabin.

Definition at line 57 of file [stdair\\_inventory\\_types.hpp](#).

**31.5.2.232** `typedef double stdair::UPR_T`

Define the unsold protection (UPR).

Definition at line 60 of file [stdair\\_inventory\\_types.hpp](#).

**31.5.2.233** `typedef double stdair::BookingLimit_T`

Define the value of the booking limit.

Define the Booking Limit.

It is a double, as it allows for overbooking.

Definition at line 63 of file [stdair\\_inventory\\_types.hpp](#).

**31.5.2.234** `typedef double stdair::AuthorizationLevel_T`

Define the value of the authorization level.

Definition at line 66 of file [stdair\\_inventory\\_types.hpp](#).

**31.5.2.235** `typedef double stdair::CapacityAdjustment_T`

Define the value of the adjustment for cabin capacity.

Definition at line 69 of file [stdair\\_inventory\\_types.hpp](#).

**31.5.2.236** `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](#).

**31.5.2.237** `typedef bool stdair::AvailabilityStatus_T`

Define an availability status (AVS).

Definition at line 75 of file [stdair\\_inventory\\_types.hpp](#).

**31.5.2.238** `typedef std::vector<Availability_T> stdair::BucketAvailabilities_T`

Define a list of availabilities.

Definition at line 78 of file [stdair\\_inventory\\_types.hpp](#).

**31.5.2.239** `typedef double stdair::NbOfYields_T`

Define a number of yields.

Definition at line 81 of file [stdair\\_inventory\\_types.hpp](#).

**31.5.2.240** `typedef double stdair::NbOfInventoryControlRules_T`

Define a number of InventoryControlRules.

Definition at line 84 of file [stdair\\_inventory\\_types.hpp](#).

**31.5.2.241** `typedef bool stdair::CensorshipFlag_T`

Define availability of booking limit.

Definition at line 87 of file [stdair\\_inventory\\_types.hpp](#).

**31.5.2.242** `typedef short stdair::DTD_T`

Define the type of day-to-departure.

Definition at line 90 of file [stdair\\_inventory\\_types.hpp](#).

**31.5.2.243** `typedef short stdair::DCP_T`

Define the type of data collection point.

Definition at line 93 of file [stdair\\_inventory\\_types.hpp](#).

**31.5.2.244** `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](#).

31.5.2.245 `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](#).

31.5.2.246 `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](#).

31.5.2.247 `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](#).

31.5.2.248 `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](#).

31.5.2.249 `typedef double stdair::Yield_T`

Define the yield of a virtual class.

Definition at line 113 of file [stdair\\_inventory\\_types.hpp](#).

31.5.2.250 `typedef unsigned int stdair::YieldLevel_T`

Define the yield level (yield as an integer).

Definition at line 116 of file [stdair\\_inventory\\_types.hpp](#).

31.5.2.251 `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](#).

31.5.2.252 `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](#).

31.5.2.253 `typedef double stdair::BidPrice_T`

Define the Bid-Price.

Definition at line 125 of file [stdair\\_inventory\\_types.hpp](#).

31.5.2.254 `typedef std::vector<BidPrice_T> stdair::BidPriceVector_T`

Define a Bid-Price Vector.

Definition at line 128 of file [stdair\\_inventory\\_types.hpp](#).

31.5.2.255 `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](#).



**31.5.2.256 typedef std::string stdair::ControlMode\_T**

Mode of inventory control.

Definition at line 134 of file [stdair\\_inventory\\_types.hpp](#).

**31.5.2.257 typedef double stdair::OverbookingRate\_T**

Define the rate of overbooking

Definition at line 137 of file [stdair\\_inventory\\_types.hpp](#).

**31.5.2.258 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](#).

**31.5.2.259 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](#).

**31.5.2.260 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](#).

**31.5.2.261 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](#).

**31.5.2.262 typedef boost::multi\_array<double, 2> stdair::SnapshotBlock\_T**

Define a snapshot block.

Definition at line 159 of file [stdair\\_inventory\\_types.hpp](#).

**31.5.2.263 typedef SnapshotBlock\_T::index\_range stdair::SnapshotBlockRange\_T**

Define a range for array view.

Definition at line 162 of file [stdair\\_inventory\\_types.hpp](#).

**31.5.2.264 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](#).

**31.5.2.265 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](#).

**31.5.2.266 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](#).

31.5.2.267 `typedef SnapshotBlock_T::const_array_view<2>::type stdair::ConstSegmentCabinDTDRange↔  
SnapshotView_T`

Define a const view for a given range of DTD.

Definition at line 174 of file [stdair\\_inventory\\_types.hpp](#).

31.5.2.268 `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](#).

31.5.2.269 `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](#).

31.5.2.270 `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](#).

31.5.2.271 `typedef unsigned int stdair::ReplicationNumber_T`

Define the replication number.

Definition at line 24 of file [stdair\\_maths\\_types.hpp](#).

31.5.2.272 `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](#).

31.5.2.273 `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](#).

31.5.2.274 `typedef unsigned long int stdair::RandomSeed_T`

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

Definition at line 39 of file [stdair\\_maths\\_types.hpp](#).

31.5.2.275 `typedef boost::minstd_rand stdair::BaseGenerator_T`

Random number generator.

Definition at line 44 of file [stdair\\_maths\\_types.hpp](#).

31.5.2.276 `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](#).

**31.5.2.277** `typedef boost::variate_generator<BaseGenerator_T&, UniformDistribution_T>  
stdair::UniformGenerator_T`

Uniform random generator.

Definition at line 55 of file [stdair\\_maths\\_types.hpp](#).

**31.5.2.278** `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](#).

**31.5.2.279** `typedef boost::variate_generator<BaseGenerator_T&, NormalDistribution_T>  
stdair::NormalGenerator_T`

Normal random generator.

Definition at line 66 of file [stdair\\_maths\\_types.hpp](#).

**31.5.2.280** `typedef boost::exponential_distribution stdair::ExponentialDistribution_T`

Type definition for the exponential distribution (characteristics).

Definition at line 69 of file [stdair\\_maths\\_types.hpp](#).

**31.5.2.281** `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](#).

**31.5.2.282** `typedef double stdair::MeanValue_T`

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

Definition at line 79 of file [stdair\\_maths\\_types.hpp](#).

**31.5.2.283** `typedef double stdair::StdDevValue_T`

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

Definition at line 84 of file [stdair\\_maths\\_types.hpp](#).

**31.5.2.284** `typedef std::pair<MeanValue_T, StdDevValue_T> stdair::MeanStdDevPair_T`

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

Definition at line 89 of file [stdair\\_maths\\_types.hpp](#).

**31.5.2.285** `typedef std::vector<MeanStdDevPair_T> stdair::MeanStdDevPairVector_T`

Define a vector of couple (mean, standard deviation)

Definition at line 94 of file [stdair\\_maths\\_types.hpp](#).

**31.5.2.286** `typedef float stdair::Probability_T`

Probability.

Definition at line 99 of file [stdair\\_maths\\_types.hpp](#).

**31.5.2.287** `typedef std::string stdair::ForecasterMode_T`

Mode of the forecaster.

Definition at line 17 of file [stdair\\_rm\\_types.hpp](#).

**31.5.2.288** `typedef short stdair::HistoricalDataLimit_T`

Limit of similar flight-dates used in the forecaster.

Definition at line 24 of file [stdair\\_rm\\_types.hpp](#).

**31.5.2.289** `typedef std::string stdair::OptimizerMode_T`

Mode of the forecaster.

Definition at line 27 of file [stdair\\_rm\\_types.hpp](#).

**31.5.2.290** `typedef NbOfBookings_T stdair::PolicyDemand_T`

Define the demand for a policy.

Definition at line 30 of file [stdair\\_rm\\_types.hpp](#).

**31.5.2.291** `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](#).

**31.5.2.292** `typedef std::vector<GeneratedDemandVector_T> stdair::GeneratedDemandVectorHolder_T`

Define the holder of the generated demand vectors.

Definition at line 37 of file [stdair\\_rm\\_types.hpp](#).

**31.5.2.293** `typedef double stdair::SellupProbability_T`

Define the sellup probability.

Definition at line 40 of file [stdair\\_rm\\_types.hpp](#).

**31.5.2.294** `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](#).

**31.5.2.295** `typedef std::vector<NbOfBookings_T> stdair::BookingVector_T`

Define the vector of historical bookings.

Definition at line 46 of file [stdair\\_rm\\_types.hpp](#).

**31.5.2.296** `typedef double stdair::FRAT5_T`

Define the FRAT5 coefficient.

Definition at line 49 of file [stdair\\_rm\\_types.hpp](#).

**31.5.2.297** `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](#).

**31.5.2.298** `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](#).

**31.5.2.299** `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](#).

**31.5.2.300** `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](#).

**31.5.2.301** `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](#).

**31.5.2.302** `typedef std::map<BookingClass*, DispatchingCurve_T> stdair::BookingClassDispatchingCurve↵  
Map_T`

Define the map between class and dispatching factor curve.

Definition at line 67 of file [stdair\\_rm\\_types.hpp](#).

**31.5.2.303** `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](#).

**31.5.2.304** `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](#).

**31.5.2.305** `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](#).

### 31.5.3 Function Documentation

**31.5.3.1** `const std::string stdair::DEFAULT_BOM_ROOT_KEY ( " -- ROOT -- " )`

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

**31.5.3.2** `const double stdair::DEFAULT_EPSILON_VALUE ( 0.0001 )`

Default very small value.

**31.5.3.3** `const unsigned int stdair::DEFAULT_FLIGHT_SPEED ( 900 )`

Default flight speed (number of kilometers per hour).

**31.5.3.4** `const NbOfFlightDates_T stdair::DEFAULT_NB_OF_FLIGHTDATES ( 0.0 )`

Default number of generated flight dates.

**31.5.3.5** `const Duration_T stdair::NULL_BOOST_TIME_DURATION ( - 1, - 1, - 1 )`

Null time duration (in boost::time\_duration unit).

**31.5.3.6** `const Duration_T stdair::DEFAULT_NULL_DURATION ( 0, 0, 0 )`

Default null duration (in boost::time\_duration unit).

**31.5.3.7** `const unsigned int stdair::DEFAULT_NB_OF_DAYS_IN_A_YEAR ( 365 )`

Default number of days in a year.

**31.5.3.8** `const unsigned int stdair::DEFAULT_NUMBER_OF_SUBDIVISIONS ( 1000 )`

Higher value per thousand

**31.5.3.9** `const DayDuration_T stdair::DEFAULT_DAY_DURATION ( 0 )`

Default number of duration days.

**31.5.3.10** `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).

**31.5.3.11** `const DOW_String_T stdair::DEFAULT_DOW_STRING ( "0000000" )`

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

**31.5.3.12** `const DateOffset_T stdair::DEFAULT_DATE_OFFSET ( 0 )`

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

**31.5.3.13** `const Date_T stdair::DEFAULT_DATE ( 2010, boost::gregorian::Jan, 1 )`

Default date for the General.

**31.5.3.14** `const DateTime_T stdair::DEFAULT_DATETIME ( DEFAULT_DATE, NULL_BOOST_TIME_DURATION )`

Default date-time.

**31.5.3.15** `const Duration_T stdair::DEFAULT_EPSILON_DURATION ( 0, 0, 0, 1 )`

Default epsilon duration (1 nanosecond).

**31.5.3.16** `const Count_T stdair::SECONDS_IN_ONE_DAY ( 86400 )`

Number of seconds in one day.

**31.5.3.17** `const Count_T stdair::MILLISECONDS_IN_ONE_SECOND ( 1000 )`

Number of milliseconds in one second

**31.5.3.18** `const RandomSeed_T stdair::DEFAULT_RANDOM_SEED ( 120765987 )`

Default random seed.

**31.5.3.19** `const AirportCode_T stdair::AIRPORT_LHR ( "LHR" )`

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

31.5.3.20 **const AirportCode\_T** stdair::AIRPORT\_SYD ( "SYD" )

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

31.5.3.21 **const CityCode\_T** stdair::POS\_LHR ( "LHR" )

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

31.5.3.22 **const Date\_T** stdair::DATE\_20110115 ( 2011 , boost::gregorian::Jan , 15 )

Date.

31.5.3.23 **const Date\_T** stdair::DATE\_20111231 ( 2011 , boost::gregorian::Dec , 31 )

31.5.3.24 **const DayDuration\_T** stdair::NO\_ADVANCE\_PURCHASE ( 0 )

Advance purchase 0 day.

31.5.3.25 **const SaturdayStay\_T** stdair::SATURDAY\_STAY ( true )

Default saturdayStay value (true).

31.5.3.26 **const SaturdayStay\_T** stdair::NO\_SATURDAY\_STAY ( false )

Default saturdayStay value (false).

31.5.3.27 **const ChangeFees\_T** stdair::CHANGE\_FEES ( true )

Default change fees value (true).

31.5.3.28 **const ChangeFees\_T** stdair::NO\_CHANGE\_FEES ( false )

Default change fees value (false).

31.5.3.29 **const NonRefundable\_T** stdair::NON\_REFUNDABLE ( true )

Default non refundable value (true).

31.5.3.30 **const NonRefundable\_T** stdair::NO\_NON\_REFUNDABLE ( false )

Default refundable value (false).

31.5.3.31 **const SaturdayStay\_T** stdair::DEFAULT\_BOM\_TREE\_SATURDAY\_STAY ( true )

Default saturdayStay value (true).

31.5.3.32 **const ChangeFees\_T** stdair::DEFAULT\_BOM\_TREE\_CHANGE\_FEES ( true )

Default change fees value (true).

31.5.3.33 **const NonRefundable\_T** stdair::DEFAULT\_BOM\_TREE\_NON\_REFUNDABLE ( true )

Default non refundable value (true).

31.5.3.34 **const DayDuration\_T** stdair::NO\_STAY\_DURATION ( 0 )

Stay duration 0 day.

31.5.3.35 **const AirlineCode\_T** stdair::AIRLINE\_CODE\_BA ( "BA" )

Airline code "BA".

**31.5.3.36** `const CabinCode_T stdair::CABIN_Y ( "Y" )`

Cabin 'Y'.

**31.5.3.37** `const ClassCode_T stdair::CLASS_CODE_Y ( "Y" )`

Class code 'Y'.

**31.5.3.38** `const ClassCode_T stdair::CLASS_CODE_Q ( "Q" )`

Class code 'Q'.

**31.5.3.39** `const AirportCode_T stdair::AIRPORT_SIN ( "SIN" )`

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

**31.5.3.40** `const AirportCode_T stdair::AIRPORT_BKK ( "BKK" )`

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

**31.5.3.41** `const CityCode_T stdair::POS_SIN ( "SIN" )`

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

**31.5.3.42** `const CabinCode_T stdair::CABIN_ECO ( "Eco" )`

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

**31.5.3.43** `const FrequentFlyer_T stdair::FREQUENT_FLYER_MEMBER ( "M" )`

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

**31.5.3.44** `const FamilyCode_T stdair::DEFAULT_FAMILY_CODE ( "0" )`

Default family code value ("0").

**31.5.3.45** `const PolicyCode_T stdair::DEFAULT_POLICY_CODE ( "0" )`

Default policy code value ("0").

**31.5.3.46** `const NestingStructureCode_T stdair::DEFAULT_NESTING_STRUCTURE_CODE ( "DEFAULT" )`

Default Nesting Structure Code ("DEFAULT").

**31.5.3.47** `const NestingStructureCode_T stdair::DISPLAY_NESTING_STRUCTURE_CODE ( "Display Nesting" )`

Display Nesting Structure Code ("Display Nesting").

**31.5.3.48** `const NestingStructureCode_T stdair::YIELD_BASED_NESTING_STRUCTURE_CODE ( "Yield-Based Nesting" )`

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

**31.5.3.49** `const NestingNodeCode_T stdair::DEFAULT_NESTING_NODE_CODE ( "0" )`

Default Nesting Node Code ("0").

**31.5.3.50** `const NbOfAirlines_T stdair::DEFAULT_NBOFAIRLINES ( 0 )`

Default number of airlines.



31.5.3.51 `const FlightPathCode_T stdair::DEFAULT_FLIGHTPATH_CODE ( "" )`

Default flight-path code value ("").

31.5.3.52 `const Distance_T stdair::DEFAULT_DISTANCE_VALUE ( 0 )`

Default distance value (kilometers).

31.5.3.53 `const ClassCode_T stdair::DEFAULT_CLOSED_CLASS_CODE ( "CC" )`

Default closed class code.

31.5.3.54 `const NbOfBookings_T stdair::DEFAULT_CLASS_NB_OF_BOOKINGS ( 0 )`

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

31.5.3.55 `const NbOfBookings_T stdair::DEFAULT_CLASS_TOTAL_NB_OF_BOOKINGS ( 0 )`

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

31.5.3.56 `const NbOfBookings_T stdair::DEFAULT_CLASS_UNCONSTRAINED_DEMAND ( 0 )`

Default unconstrained demand for [BookingClass](#).

31.5.3.57 `const NbOfBookings_T stdair::DEFAULT_CLASS_REMAINING_DEMAND_MEAN ( 0 )`

Default remaining future demand mean for [BookingClass](#).

31.5.3.58 `const NbOfBookings_T stdair::DEFAULT_CLASS_REMAINING_DEMAND_STANDARD_DEVIATION ( 0 )`

Default remaining future demand standard deviation for [BookingClass](#).

31.5.3.59 `const NbOfCancellations_T stdair::DEFAULT_CLASS_NB_OF_CANCELLATIONS ( 0 )`

Default number of cancellations for [BookingClass](#).

31.5.3.60 `const NbOfNoShows_T stdair::DEFAULT_CLASS_NB_OF_NOSHOWS ( 0 )`

Default number of no-shows for [BookingClass](#).

31.5.3.61 `const CabinCapacity_T stdair::DEFAULT_CABIN_CAPACITY ( 100. 0 )`

Default cabin capacity for Leg cabins.

31.5.3.62 `const CommittedSpace_T stdair::DEFAULT_COMMITTED_SPACE ( 0. 0 )`

Default committed space value for Leg cabins.

31.5.3.63 `const BlockSpace_T stdair::DEFAULT_BLOCK_SPACE ( 0. 0 )`

Default committed space value for Leg cabins.

31.5.3.64 `const Availability_T stdair::DEFAULT_NULL_AVAILABILITY ( 0. 0 )`

Default null availability (0.0).

31.5.3.65 `const Availability_T stdair::DEFAULT_AVAILABILITY ( 9. 0 )`

Default availability (9.0).

31.5.3.66 `const Availability_T stdair::MAXIMAL_AVAILABILITY ( 9999. 0 )`

Maximal offered capacity in a cabin.

31.5.3.67 `const CensorshipFlag_T stdair::DEFAULT_CLASS_CENSORSHIPFLAG ( false )`

Default boolean for censorship flag given the status of availability for [BookingClass](#).

31.5.3.68 `const BookingLimit_T stdair::DEFAULT_CLASS_BOOKING_LIMIT ( 9999. 0 )`

Default booking limit value for [BookingClass](#).

31.5.3.69 `const AuthorizationLevel_T stdair::DEFAULT_CLASS_AUTHORIZATION_LEVEL ( 9999. 0 )`

Default authorization level for [BookingClass](#).

31.5.3.70 `const AuthorizationLevel_T stdair::DEFAULT_CLASS_MAX_AUTHORIZATION_LEVEL ( 9999. 0 )`

Default MAX value of authorization level for [BookingClass](#).

31.5.3.71 `const AuthorizationLevel_T stdair::DEFAULT_CLASS_MIN_AUTHORIZATION_LEVEL ( 0. 0 )`

Default MIN value of authorization level for [BookingClass](#).

31.5.3.72 `const OverbookingRate_T stdair::DEFAULT_CLASS_OVERBOOKING_RATE ( 0. 0 )`

Default over-booking rate for [BookingClass](#).

31.5.3.73 `const BookingRatio_T stdair::DEFAULT_OND_BOOKING_RATE ( 0. 0 )`

Default booking rate for OnD bookings over overall class bookings.

31.5.3.74 `const Fare_T stdair::DEFAULT_FARE_VALUE ( 0. 0 )`

Default Fare value.

31.5.3.75 `const Yield_T stdair::DEFAULT_CLASS_YIELD_VALUE ( 0. 0 )`

Default yield value for a virtual class.

31.5.3.76 `const Revenue_T stdair::DEFAULT_REVENUE_VALUE ( 0. 0 )`

Default Revenue value.

31.5.3.77 `const Percentage_T stdair::DEFAULT_LOAD_FACTOR_VALUE ( 100. 0 )`

Default load factor value (100%).

31.5.3.78 `const Yield_T stdair::DEFAULT_YIELD_VALUE ( 0. 0 )`

Default yield value.

31.5.3.79 `const Yield_T stdair::DEFAULT_YIELD_MAX_VALUE ( std::numeric_limits< double >:: max() )`

Default yield max value.

31.5.3.80 `const NbOfBookings_T stdair::DEFAULT_YIELD_NB_OF_BOOKINGS ( 0. 0 )`

Default number of bookings for YieldRangeStruct\_T.

31.5.3.81 **const Identity\_T** stdair::DEFAULT\_BOOKING\_NUMBER ( 0 )

Default booking number.

31.5.3.82 **const NbOfCancellations\_T** stdair::DEFAULT\_YIELD\_NB\_OF\_CANCELLATIONS ( 0.0 )

Default cancellation number for YieldRangeStruct\_T.

31.5.3.83 **const NbOfNoShows\_T** stdair::DEFAULT\_YIELD\_NB\_OF\_NOSHOWS ( 0.0 )

Default no-shows number for YieldRangeStruct\_T.

31.5.3.84 **const Availability\_T** stdair::DEFAULT\_YIELD\_AVAILABILITY ( 0.0 )

Default availability for YieldRangeStruct\_T.

31.5.3.85 **const CensorshipFlag\_T** stdair::DEFAULT\_YIELD\_CENSORSHIPFLAG ( false )

Default boolean for booking limit availability for YieldRangeStruct\_T.

31.5.3.86 **const BookingLimit\_T** stdair::DEFAULT\_YIELD\_BOOKING\_LIMIT ( 0.0 )

Default booking limit value for YieldRangeStruct\_T.

31.5.3.87 **const OverbookingRate\_T** stdair::DEFAULT\_YIELD\_OVERBOOKING\_RATE ( 0.0 )

Default over-booking rate for YieldRangeStruct\_T.

31.5.3.88 **const Fare\_T** stdair::DEFAULT\_OND\_FARE\_VALUE ( 0.0 )

Default value of Fare.

31.5.3.89 **const Count\_T** stdair::DEFAULT\_PROGRESS\_STATUS ( 0 )

Default progress status.

31.5.3.90 **const Percentage\_T** stdair::MAXIMUM\_PROGRESS\_STATUS ( 100 )

Maximum progress status.

31.5.3.91 **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.

31.5.3.92 **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.

31.5.3.93 **const PartySize\_T** stdair::DEFAULT\_PARTY\_SIZE ( 1 )

Default party size in a request.

31.5.3.94 **const DayDuration\_T** stdair::DEFAULT\_STAY\_DURATION ( 7 )

Default duration for a stay.

31.5.3.95 **const WTP\_T** stdair::DEFAULT\_WTP ( 1000.0 )

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

31.5.3.96 **const Date\_T** stdair::DEFAULT\_PREFERRED\_DEPARTURE\_DATE ( **DEFAULT\_DEPARTURE\_DATE** )

Default departure date.

31.5.3.97 **const Duration\_T** stdair::DEFAULT\_PREFERRED\_DEPARTURE\_TIME ( **8, 0, 0** )

Default preferred departure time (08:00).

31.5.3.98 **const DateOffset\_T** stdair::DEFAULT\_ADVANCE\_PURCHASE ( **22** )

Default advance purchase.

31.5.3.99 **const Date\_T** stdair::DEFAULT\_REQUEST\_DATE ( **DEFAULT\_PREFERRED\_DEPARTURE\_DATE-  
DEFAULT\_ADVANCE\_PURCHASE** )

Default request date.

31.5.3.100 **const Duration\_T** stdair::DEFAULT\_REQUEST\_TIME ( **8, 0, 0** )

Default preferred departure time (08:00).

31.5.3.101 **const DateTime\_T** stdair::DEFAULT\_REQUEST\_DATE\_TIME ( **DEFAULT\_REQUEST\_DATE ,  
DEFAULT\_REQUEST\_TIME** )

Default request date-time.

31.5.3.102 **const CabinCode\_T** stdair::DEFAULT\_PREFERRED\_CABIN ( **"M"** )

Default preferred cabin.

31.5.3.103 **const CityCode\_T** stdair::DEFAULT\_POS ( **"ALL"** )

Default point-of-sale.

31.5.3.104 **const ChannelLabel\_T** stdair::DEFAULT\_CHANNEL ( **"DC"** )

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

31.5.3.105 **const ChannelLabel\_T** stdair::CHANNEL\_DN ( **"DN"** )

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

31.5.3.106 **const ChannelLabel\_T** stdair::CHANNEL\_IN ( **"IN"** )

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

31.5.3.107 **const TripType\_T** stdair::TRIP\_TYPE\_ONE\_WAY ( **"OW"** )

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

31.5.3.108 **const TripType\_T** stdair::TRIP\_TYPE\_ROUND\_TRIP ( **"RT"** )

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

31.5.3.109 **const TripType\_T** stdair::TRIP\_TYPE\_INBOUND ( **"RI"** )

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

31.5.3.110 **const TripType\_T** stdair::TRIP\_TYPE\_OUTBOUND ( **"RO"** )

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

31.5.3.111 **const FrequentFlyer\_T** stdair::DEFAULT\_FF\_TIER ( "N" )

Default frequent flyer tier (non member).

31.5.3.112 **const PriceValue\_T** stdair::DEFAULT\_VALUE\_OF\_TIME ( 100. 0 )

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

31.5.3.113 **const IntDuration\_T** stdair::HOURL\_CONVERTED\_IN\_SECONDS ( 3600 )

Number of second in one hour

31.5.3.114 **const Duration\_T** stdair::DEFAULT\_MINIMAL\_CONNECTION\_TIME ( 0 , 30 , 0 )

Default Minimal connection time.

31.5.3.115 **const Duration\_T** stdair::DEFAULT\_MAXIMAL\_CONNECTION\_TIME ( 24 , 0 , 0 )

Default maximal connection time.

31.5.3.116 **const MatchingIndicator\_T** stdair::DEFAULT\_MATCHING\_INDICATOR ( 0. 0 )

Default Matching Indicator value.

31.5.3.117 **const PriceCurrency\_T** stdair::DEFAULT\_CURRENCY ( "EUR" )

Default currency (euro).

31.5.3.118 **const AvailabilityStatus\_T** stdair::DEFAULT\_AVAILABILITY\_STATUS ( false )

Default availability status for a travel solution.

31.5.3.119 **const AirlineCode\_T** stdair::DEFAULT\_AIRLINE\_CODE ( "XX" )

Default airline code value ("XX").

31.5.3.120 **const AirlineCode\_T** stdair::DEFAULT\_NULL\_AIRLINE\_CODE ( "" )

Default airline code value ("").

31.5.3.121 **const FlightNumber\_T** stdair::DEFAULT\_FLIGHT\_NUMBER ( 9999 )

Default flight number (9999).

31.5.3.122 **const FlightNumber\_T** stdair::DEFAULT\_FLIGHT\_NUMBER\_FF ( 255 )

Default flight number for fare families (255).

31.5.3.123 **const TableID\_T** stdair::DEFAULT\_TABLE\_ID ( 9999 )

Default data table number (9999).

31.5.3.124 **const Date\_T** stdair::DEFAULT\_DEPARTURE\_DATE ( 1900 , boost::gregorian::Jan , 1 )

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

31.5.3.125 **const AirportCode\_T** stdair::DEFAULT\_AIRPORT\_CODE ( "XXX" )

Default airport code value ("XXX").

31.5.3.126 `const AirportCode_T stdair::DEFAULT_NULL_AIRPORT_CODE ( "" )`

Default airport code value ("").

31.5.3.127 `const AirportCode_T stdair::DEFAULT_ORIGIN ( "XXX" )`

Default Origin.

31.5.3.128 `const AirportCode_T stdair::DEFAULT_DESTINATION ( "YYY" )`

Default destination.

31.5.3.129 `const CabinCode_T stdair::DEFAULT_CABIN_CODE ( "X" )`

Default cabin code.

31.5.3.130 `const FamilyCode_T stdair::DEFAULT_FARE_FAMILY_CODE ( "EcoSaver" )`

Default fare family Code.

31.5.3.131 `const FamilyCode_T stdair::DEFAULT_NULL_FARE_FAMILY_CODE ( "NoFF" )`

Default null fare family Code ("NoFF").

31.5.3.132 `const ClassCode_T stdair::DEFAULT_CLASS_CODE ( "X" )`

Default class code value ("X").

31.5.3.133 `const ClassCode_T stdair::DEFAULT_NULL_CLASS_CODE ( "" )`

Default null class code value ("").

31.5.3.134 `const BidPrice_T stdair::DEFAULT_BID_PRICE ( 0.0 )`

Default Bid-Price.

31.5.3.135 `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.

31.5.3.136 `const unsigned short stdair::MAXIMAL_NUMBER_OF_SEGMENTS_IN_OND ( 3 )`

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

31.5.3.137 `const SeatIndex_T stdair::DEFAULT_SEAT_INDEX ( 1 )`

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

31.5.3.138 `const NbOfSeats_T stdair::DEFAULT_NULL_BOOKING_NUMBER ( 0 )`

Default number of bookings.

31.5.3.139 `const CapacityAdjustment_T stdair::DEFAULT_NULL_CAPACITY_ADJUSTMENT ( 0 )`

Default capacity adjustment of the cabin.

31.5.3.140 `const UPR_T stdair::DEFAULT_NULL_UPR ( 0 )`

Default unsold Protection (UPR).

31.5.3.141 `const std::string stdair::DEFAULT_FARE_FAMILY_VALUE_TYPE ( "FF" )`

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

31.5.3.142 `const std::string stdair::DEFAULT_SEGMENT_CABIN_VALUE_TYPE ( "SC" )`

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

31.5.3.143 `const std::string stdair::DEFAULT_KEY_FLD_DELIMITER ( ";" )`

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

31.5.3.144 `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).

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

Default token for decoding a full string display.

31.5.3.146 `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](#).

31.5.3.147 `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](#).

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

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

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

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

31.5.3.152 `void stdair::intDisplay ( std::ostream & oStream, const int & iInt )`

Definition at line 159 of file [BookingRequestStruct.cpp](#).

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

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

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

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

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

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

- 31.5.3.159 `template void stdair::InventoryKey::serialize< ba::text_oarchive > ( ba::text_oarchive & , unsigned int )`
- 31.5.3.160 `template void stdair::InventoryKey::serialize< ba::text_iarchive > ( ba::text_iarchive & , unsigned int )`
- 31.5.3.161 `template void stdair::NestingNodeKey::serialize< ba::text_oarchive > ( ba::text_oarchive & , unsigned int )`
- 31.5.3.162 `template void stdair::NestingNodeKey::serialize< ba::text_iarchive > ( ba::text_iarchive & , unsigned int )`
- 31.5.3.163 `template void stdair::NestingStructureKey::serialize< ba::text_oarchive > ( ba::text_oarchive & , unsigned int )`
- 31.5.3.164 `template void stdair::NestingStructureKey::serialize< ba::text_iarchive > ( ba::text_iarchive & , unsigned int )`
- 31.5.3.165 `template void stdair::OnDDateKey::serialize< ba::text_oarchive > ( ba::text_oarchive & , unsigned int )`
- 31.5.3.166 `template void stdair::OnDDateKey::serialize< ba::text_iarchive > ( ba::text_iarchive & , unsigned int )`
- 31.5.3.167 `const boost::char_separator<char> stdair::TokeniserDashSeparator ( "-" )`

Dash delimiter for the tokenisation process.

Referenced by [stdair::ParsedKey::getFlightDateKey\(\)](#).

- 31.5.3.168 `const boost::char_separator<char> stdair::TokeniserTimeSeparator ( ":" )`

Time delimiter for the tokenisation process.

Referenced by [stdair::ParsedKey::getBoardingTime\(\)](#).

- 31.5.3.169 `template void stdair::PolicyKey::serialize< ba::text_oarchive > ( ba::text_oarchive & , unsigned int )`
- 31.5.3.170 `template void stdair::PolicyKey::serialize< ba::text_iarchive > ( ba::text_iarchive & , unsigned int )`
- 31.5.3.171 `template void stdair::SegmentCabinKey::serialize< ba::text_oarchive > ( ba::text_oarchive & , unsigned int )`
- 31.5.3.172 `template void stdair::SegmentCabinKey::serialize< ba::text_iarchive > ( ba::text_iarchive & , unsigned int )`
- 31.5.3.173 `template void stdair::SegmentDateKey::serialize< ba::text_oarchive > ( ba::text_oarchive & , unsigned int )`
- 31.5.3.174 `template void stdair::SegmentDateKey::serialize< ba::text_iarchive > ( ba::text_iarchive & , unsigned int )`
- 31.5.3.175 `template void stdair::SegmentSnapshotTableKey::serialize< ba::text_oarchive > ( ba::text_oarchive & , unsigned int )`
- 31.5.3.176 `template void stdair::SegmentSnapshotTableKey::serialize< ba::text_iarchive > ( ba::text_iarchive & , unsigned int )`
- 31.5.3.177 `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\(\)](#).

- 31.5.3.178 `template void stdair::BomRoot::serialize< ba::text_oarchive > ( ba::text_oarchive & , unsigned int )`
- 31.5.3.179 `template void stdair::BomRoot::serialize< ba::text_iarchive > ( ba::text_iarchive & , unsigned int )`
- 31.5.3.180 `template void stdair::Inventory::serialize< ba::text_oarchive > ( ba::text_oarchive & , unsigned int )`



- 31.5.3.181 `template void stdair::Inventory::serialize< ba::text_iarchive > ( ba::text_iarchive & , unsigned int )`
- 31.5.3.182 `template void stdair::FlightDate::serialize< ba::text_oarchive > ( ba::text_oarchive & , unsigned int )`
- 31.5.3.183 `template void stdair::FlightDate::serialize< ba::text_iarchive > ( ba::text_iarchive & , unsigned int )`
- 31.5.3.184 `template void stdair::SegmentDate::serialize< ba::text_oarchive > ( ba::text_oarchive & , unsigned int )`
- 31.5.3.185 `template void stdair::SegmentDate::serialize< ba::text_iarchive > ( ba::text_iarchive & , unsigned int )`
- 31.5.3.186 `template void stdair::SegmentCabin::serialize< ba::text_oarchive > ( ba::text_oarchive & , unsigned int )`
- 31.5.3.187 `template void stdair::SegmentCabin::serialize< ba::text_iarchive > ( ba::text_iarchive & , unsigned int )`

#### 31.5.4 Variable Documentation

##### 31.5.4.1 `const std::string stdair::DOW_STR`

###### Initial value:

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

Day names (in English).

Representation of Dow-Of-the-Week

Definition at line 53 of file [BasConst.cpp](#).

Referenced by [stdair::DoWStruct::describe\(\)](#).

##### 31.5.4.2 `const UnconstrainingMethod stdair::DEFAULT_UNCONSTRAINING_METHOD`

Default Unconstraining Method (By Expectation-Maximisation).

Default Unconstraining Method (By Time Frame).

Definition at line 140 of file [BasConst\\_Inventory.hpp](#).

##### 31.5.4.3 `const PartnershipTechnique stdair::DEFAULT_PARTNERSHIP_TECHNIQUE`

Default Partnership Technique (None).

Definition at line 149 of file [BasConst\\_Inventory.hpp](#).

##### 31.5.4.4 `const ForecastingMethod stdair::DEFAULT_FORECASTING_METHOD`

Default Forecasting Method (Q Forecasting).

Definition at line 137 of file [BasConst\\_Inventory.hpp](#).

##### 31.5.4.5 `const PreOptimisationMethod stdair::DEFAULT_PREOPTIMISATION_METHOD`

Default Pre-Optimisation Method (NONE).

Definition at line 143 of file [BasConst\\_Inventory.hpp](#).

##### 31.5.4.6 `const OptimisationMethod stdair::DEFAULT_OPTIMISATION_METHOD`

Default Optimisation Method (Leg Based Monte Carlo).

Default Optimisation Method (Leg Based EMSRb).

Definition at line 146 of file [BasConst\\_Inventory.hpp](#).

31.5.4.7 `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](#).

31.5.4.8 `const Date_T stdair::DEFAULT_DICO_STUDIED_DATE`

Default DICO studied date.

Definition at line 426 of file [BasConst.cpp](#).

31.5.4.9 `const AirlineCodeList_T stdair::DEFAULT_AIRLINE_CODE_LIST`

Default airline code list value (empty vector).

Definition at line 436 of file [BasConst.cpp](#).

31.5.4.10 `const ClassList_StringList_T stdair::DEFAULT_CLASS_CODE_LIST`

Default class code list value (empty vector).

Definition at line 478 of file [BasConst.cpp](#).

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

Default Bid-Price Vector.

Default Bid-Price Vector (empty vector).

Definition at line 484 of file [BasConst.cpp](#).

31.5.4.12 `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\(\)](#).

31.5.4.13 `const DCPList_T stdair::DEFAULT_DCP_LIST = DefaultDCPList::init()`

Default data collection point list.

Definition at line 517 of file [BasConst.cpp](#).

31.5.4.14 `const FRAT5Curve_T stdair::FRAT5_CURVE_A`**Initial value:**

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

FRAT5 curve A for forecasting and optimisation.

FRAT5 curves for forecasting and optimisation.

Definition at line 531 of file [BasConst.cpp](#).

31.5.4.15 `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](#).

#### 31.5.4.16 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](#).

#### 31.5.4.17 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](#).

#### 31.5.4.18 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.

Disutility curves for fare families.

Definition at line 591 of file [BasConst.cpp](#).

#### 31.5.4.19 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](#).

#### 31.5.4.20 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](#).

#### 31.5.4.21 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](#).

#### 31.5.4.22 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](#).

#### 31.5.4.23 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](#).

#### 31.5.4.24 const DTDFratMap\_T stdair::DEFAULT\_DTD\_FRAT5COEF\_MAP

##### Initial value:

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

Default frat5 coef map for demand to come forecaster.

Default frat5 coef map.

Definition at line 695 of file [BasConst.cpp](#).

#### 31.5.4.25 const DTDProbMap\_T stdair::DEFAULT\_DTD\_PROB\_MAP

##### Initial value:

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

Default arrival pattern map.

Definition at line 712 of file [BasConst.cpp](#).

#### 31.5.4.26 const OnDStringList\_T stdair::DEFAULT\_OND\_STRING\_LIST

Default list of full keys.

Definition at line 736 of file [BasConst.cpp](#).

#### 31.5.4.27 `const std::string stdair::DISPLAY_LEVEL_STRING_ARRAY`

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

Definition at line 742 of file [BasConst.cpp](#).

#### 31.5.4.28 `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::LegCabin::getFullerKey\(\)](#), [stdair::SegmentCabin::getFullerKey\(\)](#), and [stdair::ParsedKey::toString\(\)](#).

#### 31.5.4.29 `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::ParsedKey::toString\(\)](#), [stdair::AirportPairKey::toString\(\)](#), [stdair::PosChannelKey::toString\(\)](#), [stdair::SegmentDateKey::toString\(\)](#), [stdair::AirlineClassListKey::toString\(\)](#), and [stdair::FlightDateKey::toString\(\)](#).

#### 31.5.4.30 `const boost::char_separator<char> stdair::DEFAULT_KEY_TOKEN_DELIMITER`

Default token for decoding a full string display.

Referenced by [stdair::BomKeyManager::extractKeys\(\)](#).

#### 31.5.4.31 `const Distance_T stdair::DEFAULT_DISTANCE_VALUE`

Default distance value, in kilometers (0).

Default distance value (kilometers).

Definition at line 30 of file [BasConst\\_General.hpp](#).

#### 31.5.4.32 `const ClassCode_T stdair::DEFAULT_CLOSED_CLASS_CODE`

Default closed class code ("CC").

#### 31.5.4.33 `const NbOfBookings_T stdair::DEFAULT_CLASS_NB_OF_BOOKINGS`

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

Default number of bookings for [BookingClass](#).

Default number of bookings (0).

Definition at line 27 of file [BasConst\\_General.hpp](#).

#### 31.5.4.34 `const NbOfBookings_T stdair::DEFAULT_CLASS_TOTAL_NB_OF_BOOKINGS`

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

#### 31.5.4.35 `const NbOfBookings_T stdair::DEFAULT_CLASS_UNCONSTRAINED_DEMAND`

Default unconstrained demand for [BookingClass](#) (0).

#### 31.5.4.36 `const NbOfBookings_T stdair::DEFAULT_CLASS_REMAINING_DEMAND_MEAN`

Default remaining future demand mean for [BookingClass](#) (0).

#### 31.5.4.37 `const NbOfBookings_T stdair::DEFAULT_CLASS_REMAINING_DEMAND_STANDARD_DEVIATION`

Default remaining future demand standard deviation for [BookingClass](#) (0).

31.5.4.38 `const NbOfCancellations_T stdair::DEFAULT_CLASS_NB_OF_CANCELLATIONS`

Default number of cancellations for [BookingClass](#) (0).

31.5.4.39 `const NbOfNoShows_T stdair::DEFAULT_CLASS_NB_OF_NOSHOWS`

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

31.5.4.40 `const CabinCapacity_T stdair::DEFAULT_CABIN_CAPACITY`

Default cabin capacity for Leg cabins (0.0).

Default cabin capacity for Leg cabins.

Definition at line 21 of file [BasConst\\_General.hpp](#).

31.5.4.41 `const CommittedSpace_T stdair::DEFAULT_COMMITTED_SPACE`

Default committed space value for Leg cabins (0.0).

31.5.4.42 `const BlockSpace_T stdair::DEFAULT_BLOCK_SPACE`

Default committed space value for Leg cabins (0.0).

31.5.4.43 `const Availability_T stdair::DEFAULT_NULL_AVAILABILITY`

Default null availability (0.0).

31.5.4.44 `const Availability_T stdair::DEFAULT_AVAILABILITY`

Default availability (9.0).

31.5.4.45 `const CensorshipFlag_T stdair::DEFAULT_CLASS_CENSORSHIPFLAG`

Default boolean for censorship flag given the status of availability for [BookingClass](#).

31.5.4.46 `const BookingLimit_T stdair::DEFAULT_CLASS_BOOKING_LIMIT`

Default booking limit value for [BookingClass](#).

31.5.4.47 `const AuthorizationLevel_T stdair::DEFAULT_CLASS_AUTHORIZATION_LEVEL`

Default authorization level for [BookingClass](#).

31.5.4.48 `const AuthorizationLevel_T stdair::DEFAULT_CLASS_MAX_AUTHORIZATION_LEVEL`

Default MAX value of authorization level for [BookingClass](#).

31.5.4.49 `const AuthorizationLevel_T stdair::DEFAULT_CLASS_MIN_AUTHORIZATION_LEVEL`

Default MIN value of authorization level for [BookingClass](#).

31.5.4.50 `const OverbookingRate_T stdair::DEFAULT_CLASS_OVERBOOKING_RATE`

Default over-booking rate for [BookingClass](#).

31.5.4.51 `const Fare_T stdair::DEFAULT_FARE_VALUE`

Default fare.

Default value of Fare.

Definition at line 36 of file [BasConst\\_General.hpp](#).

**31.5.4.52 const Revenue\_T stdair::DEFAULT\_REVENUE\_VALUE**

Default revenue value for [BookingClass](#).

Default revenue value.

Definition at line 42 of file [BasConst\\_General.hpp](#).

**31.5.4.53 const PriceCurrency\_T stdair::DEFAULT\_CURRENCY**

Default currency (euro).

Definition at line 39 of file [BasConst\\_General.hpp](#).

**31.5.4.54 const Percentage\_T stdair::DEFAULT\_LOAD\_FACTOR\_VALUE**

Default load factor value (100%).

**31.5.4.55 const DayDuration\_T stdair::DEFAULT\_DAY\_DURATION**

Default number of duration days (0).

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

Definition at line 26 of file [BasConst\\_Period\\_BOM.hpp](#).

**31.5.4.56 const double stdair::DEFAULT\_EPSILON\_VALUE**

Default epsilon value between customer requirements and a fare rule.

Default epsilon value (1e-4).

Definition at line 18 of file [BasConst\\_General.hpp](#).

**31.5.4.57 const AirportCode\_T stdair::AIRPORT\_LHR**

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

**31.5.4.58 const AirportCode\_T stdair::AIRPORT\_SYD**

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

**31.5.4.59 const CityCode\_T stdair::POS\_LHR**

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

**31.5.4.60 const DayDuration\_T stdair::NO\_ADVANCE\_PURCHASE**

Advance purchase 0 day.

**31.5.4.61 const SaturdayStay\_T stdair::SATURDAY\_STAY**

Default saturdayStay value (true).

**31.5.4.62 const SaturdayStay\_T stdair::NO\_SATURDAY\_STAY**

Default saturdayStay value (false).

**31.5.4.63 const ChangeFees\_T stdair::CHANGE\_FEES**

Default change fees value (true).

**31.5.4.64 const ChangeFees\_T stdair::NO\_CHANGE\_FEES**

Default change fees value (false).

**31.5.4.65 const NonRefundable\_T stdair::NON\_REFUNDABLE**

Default non refundable value (true).

**31.5.4.66 const NonRefundable\_T stdair::NO\_NON\_REFUNDABLE**

Default refundable value (false).

**31.5.4.67 const DayDuration\_T stdair::NO\_STAY\_DURATION**

Stay duration 0 day.

**31.5.4.68 const CabinCode\_T stdair::CABIN\_Y**

Cabin 'Y'.

**31.5.4.69 const AirlineCode\_T stdair::AIRLINE\_CODE\_BA**

Airline code "BA".

**31.5.4.70 const ClassCode\_T stdair::CLASS\_CODE\_Y**

Class code 'Y'.

**31.5.4.71 const ClassCode\_T stdair::CLASS\_CODE\_Q**

Class code 'Q'.

**31.5.4.72 const AirportCode\_T stdair::AIRPORT\_SIN**

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

**31.5.4.73 const AirportCode\_T stdair::AIRPORT\_BKK**

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

**31.5.4.74 const CityCode\_T stdair::POS\_SIN**

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

**31.5.4.75 const CabinCode\_T stdair::CABIN\_ECO**

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

**31.5.4.76 const FrequentFlyer\_T stdair::FREQUENT\_FLYER\_MEMBER**

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

**31.5.4.77 const Count\_T stdair::DEFAULT\_PROGRESS\_STATUS**

Default progress status.

Referenced by [stdair::ProgressStatus::reset\(\)](#).

**31.5.4.78 const Date\_T stdair::DEFAULT\_EVENT\_OLDEST\_DATE**

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

**31.5.4.79 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::get←](#)



[EventTime\(\)](#).

**31.5.4.80** `const Percentage_T stdair::MAXIMUM_PROGRESS_STATUS`

Maximum progress status.

Referenced by [stdair::ProgressStatus::progress\(\)](#).

**31.5.4.81** `const std::string stdair::DEFAULT_BOM_ROOT_KEY`

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

**31.5.4.82** `const NbOfFlightDates_T stdair::DEFAULT_NB_OF_FLIGHTDATES`

Default number of generated flight dates (0).

**31.5.4.83** `const unsigned int stdair::DEFAULT_FLIGHT_SPEED`

Default flight speed (number of kilometers per hour).

**31.5.4.84** `const BookingRatio_T stdair::DEFAULT_OND_BOOKING_RATE`

Default booking rate for OnD bookings over overall class bookings.

**31.5.4.85** `const Count_T stdair::SECONDS_IN_ONE_DAY`

Number of seconds in one day (86400).

**31.5.4.86** `const Count_T stdair::MILLISECONDS_IN_ONE_SECOND`

Number of milliseconds in one second (1000).

**31.5.4.87** `const Date_T stdair::DEFAULT_DATE`

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

**31.5.4.88** `const DateTime_T stdair::DEFAULT_DATETIME`

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

**31.5.4.89** `const Duration_T stdair::DEFAULT_EPSILON_DURATION`

Default epsilon duration (1 nanosecond).

**31.5.4.90** `const RandomSeed_T stdair::DEFAULT_RANDOM_SEED`

Default random seed (120765987).

Referenced by [stdair::BookingClass::generateDemandSamples\(\)](#).

**31.5.4.91** `const Duration_T stdair::NULL_BOOST_TIME_DURATION`

Null time duration (in boost::time\_duration unit).

Definition at line 23 of file [BasConst\\_TravelSolution.hpp](#).

**31.5.4.92** `const Duration_T stdair::DEFAULT_NULL_DURATION`

Default null duration (in boost::time\_duration unit).

**31.5.4.93** `const Fare_T stdair::DEFAULT_CLASS_FARE_VALUE`

Default value of Availability.

**31.5.4.94 const NbOfAirlines\_T stdair::DEFAULT\_NBOFAIRLINES**

Default number of airlines (0).

**31.5.4.95 const unsigned int stdair::DEFAULT\_NB\_OF\_DAYS\_IN\_A\_YEAR**

Default number of days in a year (365).

**31.5.4.96 const ChannelLabel\_T stdair::DEFAULT\_CHANNEL**

Default channel.

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

Definition at line 48 of file [BasConst\\_Request.hpp](#).

**31.5.4.97 const unsigned int stdair::DEFAULT\_NUMBER\_OF\_SUBDIVISIONS**

Higher value per thousand

Referenced by [stdair::DictionaryManager::keyToValue\(\)](#), and [stdair::DictionaryManager::valueToKey\(\)](#).

**31.5.4.98 const AirlineCode\_T stdair::DEFAULT\_AIRLINE\_CODE**

Default airline code value ("XX").

Referenced by [stdair::BomRetriever::retrieveDummyLegCabin\(\)](#), and [stdair::BomRetriever::retrieveDummy↵SegmentCabin\(\)](#).

**31.5.4.99 const AirlineCode\_T stdair::DEFAULT\_NULL\_AIRLINE\_CODE**

Default airline code value ("").

**31.5.4.100 const FlightNumber\_T stdair::DEFAULT\_FLIGHT\_NUMBER**

Default flight number (9999).

Referenced by [stdair::BomRetriever::retrieveDummyLegCabin\(\)](#), and [stdair::BomRetriever::retrieveDummy↵SegmentCabin\(\)](#).

**31.5.4.101 const FlightNumber\_T stdair::DEFAULT\_FLIGHT\_NUMBER\_FF**

Default flight number for fare families (255).

Referenced by [stdair::BomRetriever::retrieveDummyLegCabin\(\)](#), and [stdair::BomRetriever::retrieveDummy↵SegmentCabin\(\)](#).

**31.5.4.102 const TableID\_T stdair::DEFAULT\_TABLE\_ID**

Default data table ID (9999).

**31.5.4.103 const Date\_T stdair::DEFAULT\_DEPARTURE\_DATE**

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

Referenced by [stdair::BomRetriever::retrieveDummyLegCabin\(\)](#), and [stdair::BomRetriever::retrieveDummy↵SegmentCabin\(\)](#).

**31.5.4.104 const AirportCode\_T stdair::DEFAULT\_AIRPORT\_CODE**

Default airport code value ("XXX").

**31.5.4.105 const AirportCode\_T stdair::DEFAULT\_NULL\_AIRPORT\_CODE**

Default airport code value ("").

**31.5.4.106 const AirportCode\_T stdair::DEFAULT\_ORIGIN**

Default Origin ("XXX").

Referenced by [stdair::BomRetriever::retrieveDummyLegCabin\(\)](#), and [stdair::BomRetriever::retrieveDummySegmentCabin\(\)](#).

**31.5.4.107 const AirportCode\_T stdair::DEFAULT\_DESTINATION**

Default Destination ("XXX").

Referenced by [stdair::BomRetriever::retrieveDummySegmentCabin\(\)](#).

**31.5.4.108 const CabinCode\_T stdair::DEFAULT\_CABIN\_CODE**

Default Cabin Code ("X").

Referenced by [stdair::BomRetriever::retrieveDummyLegCabin\(\)](#), and [stdair::BomRetriever::retrieveDummySegmentCabin\(\)](#).

**31.5.4.109 const FamilyCode\_T stdair::DEFAULT\_FARE\_FAMILY\_CODE**

Default Fare Family Code ("EcoSaver").

**31.5.4.110 const FamilyCode\_T stdair::DEFAULT\_NULL\_FARE\_FAMILY\_CODE**

Default null fare family Code ("NoFF").

**31.5.4.111 const PolicyCode\_T stdair::DEFAULT\_POLICY\_CODE**

Default [Policy](#) Code ("0").

**31.5.4.112 const NestingStructureCode\_T stdair::DEFAULT\_NESTING\_STRUCTURE\_CODE**

Default Nesting Structure Code ("DEFAULT").

**31.5.4.113 const NestingStructureCode\_T stdair::DISPLAY\_NESTING\_STRUCTURE\_CODE**

Display Nesting Structure Code ("Display Nesting").

**31.5.4.114 const NestingStructureCode\_T stdair::YIELD\_BASED\_NESTING\_STRUCTURE\_CODE**

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

Referenced by [stdair::FacBomManager::resetYieldBasedNestingStructure\(\)](#).

**31.5.4.115 const NestingNodeCode\_T stdair::DEFAULT\_NESTING\_NODE\_CODE**

Default Nesting Node Code ("0").

**31.5.4.116 const ClassCode\_T stdair::DEFAULT\_CLASS\_CODE**

Default class code value ("X").

**31.5.4.117 const ClassCode\_T stdair::DEFAULT\_NULL\_CLASS\_CODE**

Default null class code value ("").

**31.5.4.118 const BidPrice\_T stdair::DEFAULT\_BID\_PRICE**

Default Bid-Price (0.0).

**31.5.4.119 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.

**31.5.4.120 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).

**31.5.4.121 const Availability\_T stdair::MAXIMAL\_AVAILABILITY**

Maximal offered capacity in a cabin.

**31.5.4.122 const SeatIndex\_T stdair::DEFAULT\_SEAT\_INDEX**

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

**31.5.4.123 const NbOfSeats\_T stdair::DEFAULT\_NULL\_BOOKING\_NUMBER**

Default number of bookings.

**31.5.4.124 const CapacityAdjustment\_T stdair::DEFAULT\_NULL\_CAPACITY\_ADJUSTMENT**

Default capacity adjustment of the cabin.

**31.5.4.125 const UPR\_T stdair::DEFAULT\_NULL\_UPR**

Default unsold Protection (UPR).

**31.5.4.126 const std::string stdair::DEFAULT\_FARE\_FAMILY\_VALUE\_TYPE**

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

**31.5.4.127 const std::string stdair::DEFAULT\_SEGMENT\_CABIN\_VALUE\_TYPE**

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

**31.5.4.128 const DatePeriod\_T stdair::BOOST\_DEFAULT\_DATE\_PERIOD**

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

**31.5.4.129 const DOW\_String\_T stdair::DEFAULT\_DOW\_STRING**

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

Referenced by [stdair::DoWStruct::intersection\(\)](#), and [stdair::DoWStruct::shift\(\)](#).

**31.5.4.130 const DateOffset\_T stdair::DEFAULT\_DATE\_OFFSET**

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

**31.5.4.131 const PartySize\_T stdair::DEFAULT\_PARTY\_SIZE**

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

**31.5.4.132 const DayDuration\_T stdair::DEFAULT\_STAY\_DURATION**

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

**31.5.4.133 const WTP\_T stdair::DEFAULT\_WTP**

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

**31.5.4.134 const CityCode\_T stdair::DEFAULT\_POS**

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

**31.5.4.135 const Date\_T stdair::DEFAULT\_PREFERRED\_DEPARTURE\_DATE**

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

**31.5.4.136 const Duration\_T stdair::DEFAULT\_PREFERRED\_DEPARTURE\_TIME**

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

**31.5.4.137 const DateOffset\_T stdair::DEFAULT\_ADVANCE\_PURCHASE**

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

**31.5.4.138 const Date\_T stdair::DEFAULT\_REQUEST\_DATE**

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

**31.5.4.139 const Duration\_T stdair::DEFAULT\_REQUEST\_TIME**

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

**31.5.4.140 const DateTime\_T stdair::DEFAULT\_REQUEST\_DATE\_TIME**

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

**31.5.4.141 const CabinCode\_T stdair::DEFAULT\_PREFERRED\_CABIN**

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

**31.5.4.142 const ChannelLabel\_T stdair::CHANNEL\_DN**

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

**31.5.4.143 const ChannelLabel\_T stdair::CHANNEL\_IN**

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

**31.5.4.144 const TripType\_T stdair::TRIP\_TYPE\_ONE\_WAY**

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

Referenced by [stdair::BookingRequestStruct::display\(\)](#).

**31.5.4.145 const TripType\_T stdair::TRIP\_TYPE\_ROUND\_TRIP**

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

Referenced by [stdair::YieldFeatures::isTripTypeValid\(\)](#), and [stdair::FareFeatures::isTripTypeValid\(\)](#).

**31.5.4.146 const TripType\_T stdair::TRIP\_TYPE\_INBOUND**

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

Referenced by [stdair::YieldFeatures::isTripTypeValid\(\)](#), and [stdair::FareFeatures::isTripTypeValid\(\)](#).

**31.5.4.147 const TripType\_T stdair::TRIP\_TYPE\_OUTBOUND**

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

Referenced by [stdair::YieldFeatures::isTripTypeValid\(\)](#), and [stdair::FareFeatures::isTripTypeValid\(\)](#).

31.5.4.148 `const FrequentFlyer_T stdair::DEFAULT_FF_TIER`

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

31.5.4.149 `const PriceValue_T stdair::DEFAULT_VALUE_OF_TIME`

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

31.5.4.150 `const IntDuration_T stdair::HOURL_CONVERTED_IN_SECONDS`

Number of second in one hour

31.5.4.151 `const Duration_T stdair::DEFAULT_MINIMAL_CONNECTION_TIME`

Default Minimal connection time.

31.5.4.152 `const Duration_T stdair::DEFAULT_MAXIMAL_CONNECTION_TIME`

Default maximal connection time.

31.5.4.153 `const FlightPathCode_T stdair::DEFAULT_FLIGHTPATH_CODE`

Default flightPathCode value ("").

31.5.4.154 `const Availability_T stdair::DEFAULT_CLASS_AVAILABILITY`

Default value of Availability.

31.5.4.155 `const AvailabilityStatus_T stdair::DEFAULT_AVAILABILITY_STATUS`

Default availability status for a travel solution.

31.5.4.156 `const unsigned short stdair::DEFAULT_NUMBER_OF_REQUIRED_SEATS`

Default number of required seats by the demand.

31.5.4.157 `const MatchingIndicator_T stdair::DEFAULT_MATCHING_INDICATOR`

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

31.5.4.158 `const AirlineCode_T stdair::DEFAULT_DICO_STUDIED_AIRLINE`

Default DICO studied airline.

31.5.4.159 `const Yield_T stdair::DEFAULT_YIELD_VALUE`

Default yield value.

31.5.4.160 `const Yield_T stdair::DEFAULT_YIELD_MAX_VALUE`

Default yield max value.

## 31.6 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](#)]

## 31.6.1 Detailed Description

Level of logs.

## 31.6.2 Enumeration Type Documentation

## 31.6.2.1 enum stdair::LOG::EN\_LogLevel

## Enumerator

***CRITICAL***

***ERROR***

***NOTIFICATION***

***WARNING***

***DEBUG***

***VERBOSE***

***LAST\_VALUE***

Definition at line 18 of file [stdair\\_log.hpp](#).

## 31.6.3 Variable Documentation

31.6.3.1 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\(\)](#).

## 31.7 stdair\_test Namespace Reference

## Classes

- struct [BookingClass](#)
- struct [Cabin](#)

## 31.7.1 Detailed Description

Namespace gathering classes and structures for test purposes

## 31.8 swift Namespace Reference

The wrapper namespace.

## Classes

- class [SKeymap](#)  
*The readline keymap wrapper.*
- class [SReadline](#)  
*The readline library wrapper.*

## 31.8.1 Detailed Description

The wrapper namespace.

The namespace is also used for other library elements.

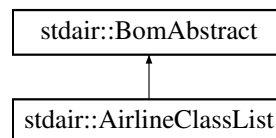
## 32 Class Documentation

### 32.1 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

- template<typename BOM >  
class [FacBom](#)
- template<typename BOM >  
class [FacCloneBom](#)
- class [FacBomManager](#)
- class [boost::serialization::access](#)

#### 32.1.1 Detailed Description

Class representing the actual attributes for a segment-features.

Definition at line 27 of file [AirlineClassList.hpp](#).

#### 32.1.2 Member Typedef Documentation

##### 32.1.2.1 typedef [AirlineClassListKey](#) [stdair::AirlineClassList::Key\\_T](#)

Definition allowing to retrieve the associated BOM key type.

Definition at line 38 of file [AirlineClassList.hpp](#).

#### 32.1.3 Constructor & Destructor Documentation

##### 32.1.3.1 [stdair::AirlineClassList::AirlineClassList](#) ( const [Key\\_T](#) & *iKey* ) [protected]

Main constructor.

Definition at line 34 of file [AirlineClassList.cpp](#).

##### 32.1.3.2 [stdair::AirlineClassList::~~AirlineClassList](#) ( ) [protected],[virtual]

Destructor.

Definition at line 39 of file [AirlineClassList.cpp](#).

#### 32.1.4 Member Function Documentation

##### 32.1.4.1 const [Key\\_T](#)& [stdair::AirlineClassList::getKey](#) ( ) const [inline]

Get the airline class list key.

Definition at line 44 of file [AirlineClassList.hpp](#).

References [\\_key](#).

**32.1.4.2 BomAbstract\*** `const stdair::AirlineClassList::getParent ( ) const` [inline]

Get the parent object.

Definition at line 49 of file [AirlineClassList.hpp](#).

References [\\_parent](#).

**32.1.4.3 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\(\)](#).

**32.1.4.4 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\(\)](#).

**32.1.4.5 const HolderMap\_T&** `stdair::AirlineClassList::getHolderMap ( ) const` [inline]

Get the map of children holders.

Definition at line 64 of file [AirlineClassList.hpp](#).

References [\\_holderMap](#).

**32.1.4.6 const stdair::Yield\_T&** `stdair::AirlineClassList::getYield ( ) const` [inline]

Get the yield.

Definition at line 69 of file [AirlineClassList.hpp](#).

References [\\_yield](#).

**32.1.4.7 const stdair::Fare\_T&** `stdair::AirlineClassList::getFare ( ) const` [inline]

Get the fare.

Definition at line 74 of file [AirlineClassList.hpp](#).

References [\\_fare](#).

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

Definition at line 80 of file [AirlineClassList.hpp](#).

References [\\_yield](#).

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

Definition at line 84 of file [AirlineClassList.hpp](#).

References [\\_fare](#).

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

Dump a Business Object into an output stream.

## Parameters

<i>ostream</i> &	the output stream.
------------------	--------------------

Implements [stdair::BomAbstract](#).

Definition at line 95 of file [AirlineClassList.hpp](#).

References [toString\(\)](#).

32.1.4.11 `void stdair::AirlineClassList::fromStream ( std::istream & ioln ) [inline], [virtual]`

Read a Business Object from an input stream.

## Parameters

<i>istream</i> &	the input stream.
------------------	-------------------

Implements [stdair::BomAbstract](#).

Definition at line 104 of file [AirlineClassList.hpp](#).

32.1.4.12 `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\(\)](#).

32.1.4.13 `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\(\)](#).

32.1.4.14 `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](#).

## 32.1.5 Friends And Related Function Documentation

32.1.5.1 `template<typename BOM > friend class FacBom [friend]`

Definition at line 28 of file [AirlineClassList.hpp](#).

32.1.5.2 `template<typename BOM > friend class FacCloneBom [friend]`

Definition at line 29 of file [AirlineClassList.hpp](#).

32.1.5.3 `friend class FacBomManager [friend]`

Definition at line 30 of file [AirlineClassList.hpp](#).

## 32.1.5.4 friend class boost::serialization::access [friend]

Definition at line 31 of file [AirlineClassList.hpp](#).

## 32.1.6 Member Data Documentation

## 32.1.6.1 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\(\)](#).

## 32.1.6.2 BomAbstract\* stdair::AirlineClassList::\_parent [protected]

Pointer on the parent class ([Inventory](#)).

Definition at line 170 of file [AirlineClassList.hpp](#).

Referenced by [getParent\(\)](#).

## 32.1.6.3 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\(\)](#).

## 32.1.6.4 Yield\_T stdair::AirlineClassList::\_yield [protected]

Definition at line 180 of file [AirlineClassList.hpp](#).

Referenced by [getYield\(\)](#), [serialize\(\)](#), [setYield\(\)](#), and [toString\(\)](#).

## 32.1.6.5 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:

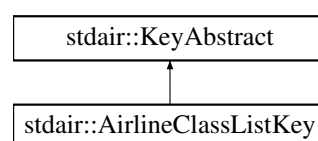
- [stdair/bom/AirlineClassList.hpp](#)
- [stdair/bom/AirlineClassList.cpp](#)

## 32.2 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](#)

## 32.2.1 Detailed Description

Key of airport-pair.

Definition at line 25 of file [AirlineClassListKey.hpp](#).

## 32.2.2 Constructor &amp; Destructor Documentation

32.2.2.1 `stdair::AirlineClassListKey::AirlineClassListKey ( const AirlineCodeList_T & iAirlineCodeList, const ClassList_StringList_T & iClassCodeList )`

Constructor.

Definition at line 24 of file [AirlineClassListKey.cpp](#).

32.2.2.2 `stdair::AirlineClassListKey::AirlineClassListKey ( const AirlineClassListKey & iKey )`

Copy constructor.

Definition at line 30 of file [AirlineClassListKey.cpp](#).

32.2.2.3 `stdair::AirlineClassListKey::~~AirlineClassListKey ( )`

Destructor.

Definition at line 36 of file [AirlineClassListKey.cpp](#).

## 32.2.3 Member Function Documentation

32.2.3.1 `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()`.

32.2.3.2 `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()`.

32.2.3.3 `void stdair::AirlineClassListKey::toStream ( std::ostream & ioOut ) const` [virtual]

Dump a Business Object Key into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 40 of file [AirlineClassListKey.cpp](#).

References [toString\(\)](#).

32.2.3.4 `void stdair::AirlineClassListKey::fromStream ( std::istream & ioIn )` [virtual]

Read a Business Object Key from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 45 of file [AirlineClassListKey.cpp](#).

32.2.3.5 `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\(\)](#).

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

Serialisation.

Definition at line 86 of file [AirlineClassListKey.cpp](#).

## 32.2.4 Friends And Related Function Documentation

32.2.4.1 `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:

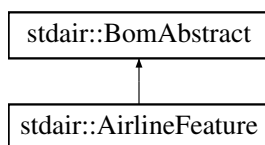
- [stdair/bom/AirlineClassListKey.hpp](#)
- [stdair/bom/AirlineClassListKey.cpp](#)

## 32.3 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

- template<typename BOM >  
class [FacBom](#)
- template<typename BOM >  
class [FacCloneBom](#)
- class [FacBomManager](#)

## 32.3.1 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](#).

## 32.3.2 Member Typedef Documentation

## 32.3.2.1 typedef AirlineFeatureKey stdair::AirlineFeature::Key\_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 35 of file [AirlineFeature.hpp](#).

## 32.3.3 Constructor &amp; Destructor Documentation

## 32.3.3.1 stdair::AirlineFeature::AirlineFeature ( const Key\_T &amp; iKey ) [protected]

Main constructor.

Definition at line 14 of file [AirlineFeature.cpp](#).

## 32.3.3.2 stdair::AirlineFeature::~~AirlineFeature ( ) [protected], [virtual]

Destructor.

Definition at line 34 of file [AirlineFeature.cpp](#).

## 32.3.4 Member Function Documentation

## 32.3.4.1 void stdair::AirlineFeature::toStream ( std::ostream &amp; ioOut ) const [inline], [virtual]

Dump a Business Object into an output stream.

## Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Implements [stdair::BomAbstract](#).

Definition at line 44 of file [AirlineFeature.hpp](#).

References [toString\(\)](#).

## 32.3.4.2 void stdair::AirlineFeature::fromStream ( std::istream &amp; ioIn ) [inline], [virtual]

Read a Business Object from an input stream.



## Parameters

<i>istream</i> &	the input stream.
------------------	-------------------

Implements [stdair::BomAbstract](#).

Definition at line 53 of file [AirlineFeature.hpp](#).

32.3.4.3 `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\(\)](#).

32.3.4.4 `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\(\)](#).

32.3.4.5 `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](#).

32.3.4.6 `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](#).

32.3.4.7 `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](#).

32.3.4.8 `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\(\)](#).

32.3.4.9 `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\(\)](#).

**32.3.4.10 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\(\)](#).

**32.3.4.11 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\(\)](#).

**32.3.4.12 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\(\)](#).

**32.3.4.13** void [stdair::AirlineFeature::init \( const ForecastingMethod & iForecastingMethod, const UnconstrainingMethod & iUnconstrainingMethod, const PreOptimisationMethod & iPreOptimisationMethod, const OptimisationMethod & iOptimisationMethod, const HistoricalDataLimit\\_T & iHistoricalDataLimit, const ControlMode\\_T & iControlMode, const PartnershipTechnique & iPartnershipTechnique \)](#)

Initialization method.

Parameters

const	<a href="#">ForecastingMethod</a> & Forecasting method.
const	<a href="#">UnconstrainingMethod</a> & Unconstraining method.
const	<a href="#">PreOptimisationMethod</a> & Pre-optimisation method.
const	<a href="#">OptimisationMethodGet</a> & Optimisation method.
const	<a href="#">HistoricalDataLimit_T</a> & Historical Data Limit
const	<a href="#">ControlMode_T</a> & Control Mode
const	<a href="#">PartnershipTechnique</a> & Partnership method.

Definition at line 38 of file [AirlineFeature.cpp](#).

References [\\_controlMode](#), [\\_forecastingMethod](#), [\\_historicalDataLimit](#), [\\_optimisationMethod](#), [\\_partnershipTechnique](#), [\\_preOptimisationMethod](#), and [\\_unconstrainingMethod](#).

**32.3.4.14** 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\(\)](#).

**32.3.4.15** `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\(\)](#).

**32.3.4.16** `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\(\)](#).

**32.3.4.17** `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\(\)](#).

**32.3.4.18** `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\(\)](#).

### 32.3.5 Friends And Related Function Documentation

**32.3.5.1** `template<typename BOM > friend class FacBom` [friend]

Definition at line 26 of file [AirlineFeature.hpp](#).

**32.3.5.2** `template<typename BOM > friend class FacCloneBom` [friend]

Definition at line 27 of file [AirlineFeature.hpp](#).

**32.3.5.3** `friend class FacBomManager` [friend]

Definition at line 28 of file [AirlineFeature.hpp](#).

### 32.3.6 Member Data Documentation

**32.3.6.1 Key\_T** stdair::AirlineFeature::\_key [protected]

Primary key (date period).

Definition at line 209 of file [AirlineFeature.hpp](#).

Referenced by [describeKey\(\)](#), and [getKey\(\)](#).

**32.3.6.2 BomAbstract\*** stdair::AirlineFeature::\_parent [protected]

Pointer on the parent class.

Definition at line 214 of file [AirlineFeature.hpp](#).

Referenced by [getParent\(\)](#).

**32.3.6.3 HolderMap\_T** stdair::AirlineFeature::\_holderMap [protected]

Map holding the children.

Definition at line 219 of file [AirlineFeature.hpp](#).

Referenced by [getHolderMap\(\)](#).

**32.3.6.4 ForecastingMethod** stdair::AirlineFeature::\_forecastingMethod [protected]

The type of forecaster.

Definition at line 224 of file [AirlineFeature.hpp](#).

Referenced by [getForecastingMethod\(\)](#), [init\(\)](#), [setForecastingMethod\(\)](#), and [toString\(\)](#).

**32.3.6.5 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\(\)](#).

**32.3.6.6 ControlMode\_T** stdair::AirlineFeature::\_controlMode [protected]

The type of inventory control.

Definition at line 234 of file [AirlineFeature.hpp](#).

Referenced by [init\(\)](#).

**32.3.6.7 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\(\)](#).

**32.3.6.8 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\(\)](#).

**32.3.6.9 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\(\)](#).

#### 32.3.6.10 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\(\)](#).

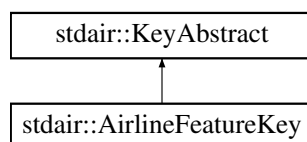
The documentation for this class was generated from the following files:

- [stdair/bom/AirlineFeature.hpp](#)
- [stdair/bom/AirlineFeature.cpp](#)

### 32.4 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

#### 32.4.1 Detailed Description

Key of [AirlineFeature](#).

Definition at line 15 of file [AirlineFeatureKey.hpp](#).

#### 32.4.2 Constructor & Destructor Documentation

##### 32.4.2.1 stdair::AirlineFeatureKey::AirlineFeatureKey ( const AirlineCode\_T &iAirlineCode )

Constructor.

Definition at line 12 of file [AirlineFeatureKey.cpp](#).

##### 32.4.2.2 stdair::AirlineFeatureKey::~~AirlineFeatureKey ( )

Destructor.

Definition at line 17 of file [AirlineFeatureKey.cpp](#).

## 32.4.3 Member Function Documentation

## 32.4.3.1 const AirlineCode\_T&amp; stdair::AirlineFeatureKey::getAirlineCode ( ) const [inline]

Get the airline code.

Definition at line 27 of file [AirlineFeatureKey.hpp](#).

## 32.4.3.2 void stdair::AirlineFeatureKey::toStream ( std::ostream &amp; ioOut ) const [virtual]

Dump a Business Object Key into an output stream.

## Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 21 of file [AirlineFeatureKey.cpp](#).

References [toString\(\)](#).

## 32.4.3.3 void stdair::AirlineFeatureKey::fromStream ( std::istream &amp; ioIn ) [virtual]

Read a Business Object Key from an input stream.

## Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 26 of file [AirlineFeatureKey.cpp](#).

## 32.4.3.4 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\(\)](#).

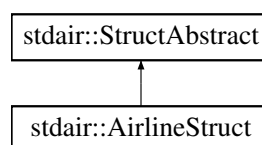
The documentation for this struct was generated from the following files:

- [stdair/bom/AirlineFeatureKey.hpp](#)
- [stdair/bom/AirlineFeatureKey.cpp](#)

## 32.5 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](#) ()

### 32.5.1 Detailed Description

Structure holding parameters describing an airline.

Definition at line 18 of file [AirlineStruct.hpp](#).

### 32.5.2 Constructor & Destructor Documentation

#### 32.5.2.1 `stdair::AirlineStruct::AirlineStruct ( const AirlineCode_T & iAirlineCode, const std::string & iAirlineName )`

Main constructor.

Definition at line 24 of file [AirlineStruct.cpp](#).

#### 32.5.2.2 `stdair::AirlineStruct::AirlineStruct ( )`

Default constructor.

Definition at line 15 of file [AirlineStruct.cpp](#).

#### 32.5.2.3 `stdair::AirlineStruct::AirlineStruct ( const AirlineStruct & iAirlineStruct )`

Default copy constructor.

Definition at line 19 of file [AirlineStruct.cpp](#).

#### 32.5.2.4 `stdair::AirlineStruct::~~AirlineStruct ( )`

Destructor.

Definition at line 30 of file [AirlineStruct.cpp](#).

### 32.5.3 Member Function Documentation

#### 32.5.3.1 `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\(\)](#).

32.5.3.2 `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\(\)](#).

32.5.3.3 `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\(\)](#).

32.5.3.4 `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\(\)](#).

32.5.3.5 `void stdair::AirlineStruct::toStream ( std::ostream & ioOut ) const`

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Definition at line 34 of file [AirlineStruct.cpp](#).

References [describe\(\)](#).

32.5.3.6 `void stdair::AirlineStruct::fromStream ( std::istream & ioIn )` `[virtual]`

Read a Business Object from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented from [stdair::StructAbstract](#).

Definition at line 39 of file [AirlineStruct.cpp](#).

32.5.3.7 `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](#)

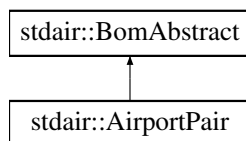
## 32.6 stdair::AirportPair Class Reference

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

- template<typename BOM >  
class [FacBom](#)
- template<typename BOM >  
class [FacCloneBom](#)
- class [FacBomManager](#)

#### 32.6.1 Detailed Description

Class representing the actual attributes for an airport-pair.

Definition at line 18 of file [AirportPair.hpp](#).

## 32.6.2 Member Typedef Documentation

## 32.6.2.1 typedef AirportPairKey stdair::AirportPair::Key\_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 28 of file [AirportPair.hpp](#).

## 32.6.3 Constructor &amp; Destructor Documentation

## 32.6.3.1 stdair::AirportPair::AirportPair ( const Key\_T &amp; iKey ) [protected]

Main constructor.

Definition at line 27 of file [AirportPair.cpp](#).

## 32.6.3.2 stdair::AirportPair::~~AirportPair ( ) [protected],[virtual]

Destructor.

Definition at line 32 of file [AirportPair.cpp](#).

## 32.6.4 Member Function Documentation

## 32.6.4.1 void stdair::AirportPair::toStream ( std::ostream &amp; ioOut ) const [inline],[virtual]

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Implements [stdair::BomAbstract](#).

Definition at line 37 of file [AirportPair.hpp](#).

References [toString\(\)](#).

## 32.6.4.2 void stdair::AirportPair::fromStream ( std::istream &amp; ioIn ) [inline],[virtual]

Read a Business Object from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Implements [stdair::BomAbstract](#).

Definition at line 46 of file [AirportPair.hpp](#).

## 32.6.4.3 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\(\)](#).

## 32.6.4.4 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\(\)](#).

**32.6.4.5** `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](#).

**32.6.4.6** `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\(\)](#).

**32.6.4.7** `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\(\)](#).

**32.6.4.8** `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](#).

**32.6.4.9** `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](#).

## 32.6.5 Friends And Related Function Documentation

**32.6.5.1** `template<typename BOM > friend class FacBom` `[friend]`

Definition at line 19 of file [AirportPair.hpp](#).

**32.6.5.2** `template<typename BOM > friend class FacCloneBom` `[friend]`

Definition at line 20 of file [AirportPair.hpp](#).

**32.6.5.3** `friend class FacBomManager` `[friend]`

Definition at line 21 of file [AirportPair.hpp](#).

## 32.6.6 Member Data Documentation

**32.6.6.1** `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\(\)](#).

#### 32.6.6.2 BomAbstract\* stdair::AirportPair::\_parent [protected]

Pointer on the parent class ([Inventory](#)).

Definition at line 129 of file [AirportPair.hpp](#).

Referenced by [getParent\(\)](#).

#### 32.6.6.3 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:

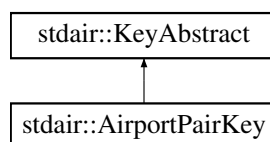
- [stdair/bom/AirportPair.hpp](#)
- [stdair/bom/AirportPair.cpp](#)

## 32.7 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

#### 32.7.1 Detailed Description

Key of airport-pair.

Definition at line 16 of file [AirportPairKey.hpp](#).

#### 32.7.2 Constructor & Destructor Documentation

**32.7.2.1** `stdair::AirportPairKey::AirportPairKey ( const stdair::AirportCode_T & iBoardingPoint, const stdair::AirportCode_T & iOffPoint )`

Main constructor.

Definition at line 22 of file [AirportPairKey.cpp](#).

**32.7.2.2** `stdair::AirportPairKey::AirportPairKey ( const AirportPairKey & iKey )`

Copy constructor.

Definition at line 28 of file [AirportPairKey.cpp](#).

**32.7.2.3** `stdair::AirportPairKey::~~AirportPairKey ( )`

Destructor.

Definition at line 34 of file [AirportPairKey.cpp](#).

### 32.7.3 Member Function Documentation

**32.7.3.1** `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\(\)](#).

**32.7.3.2** `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\(\)](#).

**32.7.3.3** `void stdair::AirportPairKey::toStream ( std::ostream & ioOut ) const` `[virtual]`

Dump a Business Object Key into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 38 of file [AirportPairKey.cpp](#).

References [toString\(\)](#).

**32.7.3.4** `void stdair::AirportPairKey::fromStream ( std::istream & ioln )` `[virtual]`

Read a Business Object Key from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 43 of file [AirportPairKey.cpp](#).

**32.7.3.5** `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](#)

## 32.8 stdair::BasChronometer Struct Reference

```
#include <stdair/basic/BasChronometer.hpp>
```

### Public Member Functions

- [BasChronometer](#) ()
- void [start](#) ()
- std::string [getStart](#) () const
- double [elapsed](#) () const

#### 32.8.1 Detailed Description

Structure allowing measuring the time elapsed between two events.

Definition at line 14 of file [BasChronometer.hpp](#).

#### 32.8.2 Constructor & Destructor Documentation

##### 32.8.2.1 stdair::BasChronometer::BasChronometer ( )

Constructor.

Definition at line 12 of file [BasChronometer.cpp](#).

#### 32.8.3 Member Function Documentation

##### 32.8.3.1 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](#).

##### 32.8.3.2 std::string stdair::BasChronometer::getStart ( ) const [inline]

Get the start time.

Definition at line 24 of file [BasChronometer.hpp](#).

##### 32.8.3.3 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:

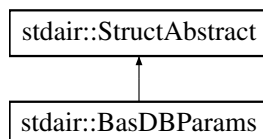
- [stdair/basic/BasChronometer.hpp](#)
- [stdair/basic/BasChronometer.cpp](#)

## 32.9 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)`

### 32.9.1 Detailed Description

Structure holding the parameters for connection to a database.

Definition at line 19 of file [BasDBParams.hpp](#).

### 32.9.2 Constructor & Destructor Documentation

#### 32.9.2.1 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](#).

### 32.9.2.2 stdair::BasDBParams::BasDBParams ( )

Default Constructor.

Definition at line 13 of file [BasDBParams.cpp](#).

### 32.9.2.3 stdair::BasDBParams::BasDBParams ( const BasDBParams & iDBParams )

Default copy constructor.

Definition at line 17 of file [BasDBParams.cpp](#).

### 32.9.2.4 stdair::BasDBParams::~~BasDBParams ( )

Destructor.

Definition at line 34 of file [BasDBParams.cpp](#).

## 32.9.3 Member Function Documentation

### 32.9.3.1 const std::string& stdair::BasDBParams::getUser ( ) const [inline]

Get the database user name.

Definition at line 23 of file [BasDBParams.hpp](#).

### 32.9.3.2 const std::string& stdair::BasDBParams::getPassword ( ) const [inline]

Get the database user password.

Definition at line 28 of file [BasDBParams.hpp](#).

### 32.9.3.3 const std::string& stdair::BasDBParams::getHost ( ) const [inline]

Get the database host name.

Definition at line 33 of file [BasDBParams.hpp](#).

### 32.9.3.4 const std::string& stdair::BasDBParams::getPort ( ) const [inline]

Get the database port number.

Definition at line 38 of file [BasDBParams.hpp](#).

### 32.9.3.5 const std::string& stdair::BasDBParams::getDBName ( ) const [inline]

Get the database name.

Definition at line 43 of file [BasDBParams.hpp](#).

### 32.9.3.6 void stdair::BasDBParams::setUser ( const std::string & iUser ) [inline]

Set the database user name.

Definition at line 50 of file [BasDBParams.hpp](#).

### 32.9.3.7 void stdair::BasDBParams::setPassword ( const std::string & iPasswd ) [inline]

Set the database password.

Definition at line 55 of file [BasDBParams.hpp](#).

### 32.9.3.8 void stdair::BasDBParams::setHost ( const std::string & iHost ) [inline]

Set the database host name.



Definition at line 60 of file [BasDBParams.hpp](#).

**32.9.3.9** `void stdair::BasDBParams::setPort ( const std::string & iPort ) [inline]`

Set the database port number.

Definition at line 65 of file [BasDBParams.hpp](#).

**32.9.3.10** `void stdair::BasDBParams::setDBName ( const std::string & iDBName ) [inline]`

Set the database name.

Definition at line 70 of file [BasDBParams.hpp](#).

**32.9.3.11** `bool stdair::BasDBParams::check ( ) const`

Check that all the parameters are fine.

Definition at line 57 of file [BasDBParams.cpp](#).

**32.9.3.12** `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\(\)](#).

**32.9.3.13** `std::string stdair::BasDBParams::toShortString ( ) const`

Get a short display of the DBParams structure.

Definition at line 43 of file [BasDBParams.cpp](#).

**32.9.3.14** `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\(\)](#).

**32.9.3.15** `void stdair::StructAbstract::toStream ( std::ostream & ioOut ) const [inline],[inherited]`

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Definition at line 29 of file [StructAbstract.hpp](#).

References [stdair::StructAbstract::describe\(\)](#).

**32.9.3.16** `virtual void stdair::StructAbstract::fromStream ( std::istream & iIn ) [inline],[virtual],[inherited]`

Read a Business Object from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented in [stdair::BookingRequestStruct](#), [stdair::EventStruct](#), [stdair::TravelSolutionStruct](#), [stdair::VirtualClassStruct](#), [stdair::ConfigHolderStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::ProgressStatusSet](#), [stdair::YieldRange](#), [stdair::CancellationStruct](#), [stdair::AirlineStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::FFDisutilityCurveHolderStruct](#), [stdair::FRAT5CurveHolderStruct](#), and [stdair::BreakPointStruct](#).

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](#)

## 32.10 stdair::BasFileMgr Struct Reference

```
#include <stdair/basic/BasFileMgr.hpp>
```

### Static Public Member Functions

- static bool [doesExistAndIsReadable](#) (const std::string &iFilepath)

#### 32.10.1 Detailed Description

Helper class for operations on files and on the file-system.

Definition at line 13 of file [BasFileMgr.hpp](#).

#### 32.10.2 Member Function Documentation

**32.10.2.1** 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:

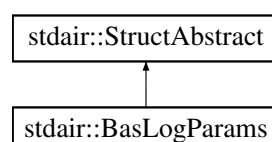
- [stdair/basic/BasFileMgr.hpp](#)
- [stdair/basic/BasFileMgr.cpp](#)

## 32.11 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 &iLogOutputStream, const bool iForceMultipleInstance=false)
- [BasLogParams](#) (const [BasLogParams](#) &)
- [~BasLogParams](#) ()
- void [toStream](#) (std::ostream &iOut) const
- virtual void [fromStream](#) (std::istream &iIn)

## Friends

- class [Logger](#)

## 32.11.1 Detailed Description

Structure holding parameters for logging.

Definition at line 19 of file [BasLogParams.hpp](#).

## 32.11.2 Constructor &amp; Destructor Documentation

32.11.2.1 `stdair::BasLogParams::BasLogParams ( const LOG::EN\_LogLevel iLogLevel, std::ostream &iLogOutputStream, const bool iForceMultipleInstance = false )`

Main Constructor.

## Parameters

<i>in</i>	<i>const</i>	<a href="#">LOG::EN_LogLevel</a> Level of the log (e.g., DEBUG)
<i>in, out</i>	<i>std::ostream&amp;</i>	(STL) Stream to log into.
<i>in</i>	<i>const</i>	bool Whether or not multiple initialisation should be forced.

Definition at line 27 of file [BasLogParams.cpp](#).

32.11.2.2 `stdair::BasLogParams::BasLogParams ( const BasLogParams &iLogParams )`

Copy constructor.

Definition at line 21 of file [BasLogParams.cpp](#).

32.11.2.3 `stdair::BasLogParams::~~BasLogParams ( )`

Destructor.

Definition at line 35 of file [BasLogParams.cpp](#).

## 32.11.3 Member Function Documentation

32.11.3.1 `const LOG::EN\_LogLevel& stdair::BasLogParams::getLogLevel ( ) const` `[inline]`

Get the log level.

Definition at line 26 of file [BasLogParams.hpp](#).

32.11.3.2 `std::ostream& stdair::BasLogParams::getLogStream ( ) const [inline]`

Get the log output stream.

Definition at line 33 of file [BasLogParams.hpp](#).

32.11.3.3 `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](#).

32.11.3.4 `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](#).

32.11.3.5 `bool stdair::BasLogParams::check ( ) const`

Check that all the parameters are fine.

32.11.3.6 `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\(\)](#).

32.11.3.7 `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](#).

32.11.3.8 `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\(\)](#).

32.11.3.9 `void stdair::StructAbstract::toStream ( std::ostream & ioOut ) const [inline],[inherited]`

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Definition at line 29 of file [StructAbstract.hpp](#).

References [stdair::StructAbstract::describe\(\)](#).

32.11.3.10 `virtual void stdair::StructAbstract::fromStream ( std::istream & ioIn ) [inline],[virtual],[inherited]`

Read a Business Object from an input stream.

## Parameters

<i>istream</i> &	the input stream.
------------------	-------------------

Reimplemented in [stdair::BookingRequestStruct](#), [stdair::EventStruct](#), [stdair::TravelSolutionStruct](#), [stdair::VirtualClassStruct](#), [stdair::ConfigHolderStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::ProgressStatusSet](#), [stdair::YieldRange](#), [stdair::CancellationStruct](#), [stdair::AirlineStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::FFDisutilityCurveHolderStruct](#), [stdair::FRAT5CurveHolderStruct](#), and [stdair::BreakPointStruct](#).

Definition at line 38 of file [StructAbstract.hpp](#).

Referenced by [operator>>\(\)](#).

### 32.11.4 Friends And Related Function Documentation

#### 32.11.4.1 friend class **Logger** [[friend](#)]

Definition at line 20 of file [BasLogParams.hpp](#).

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

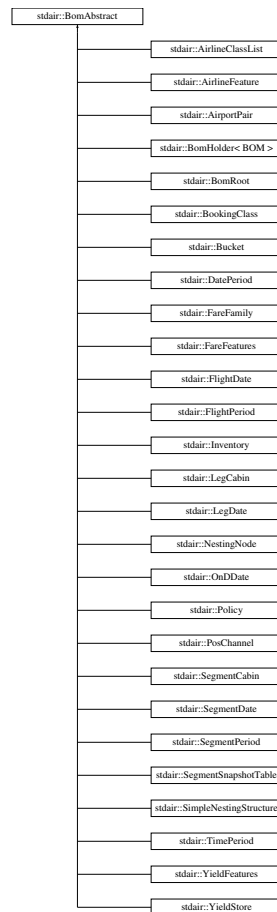
- [stdair/basic/BasLogParams.hpp](#)
- [stdair/basic/BasLogParams.cpp](#)

## 32.12 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](#) &)

#### 32.12.1 Detailed Description

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

Definition at line 24 of file [BomAbstract.hpp](#).

#### 32.12.2 Constructor & Destructor Documentation

##### 32.12.2.1 stdair::BomAbstract::BomAbstract ( ) [inline], [protected]

Protected Default Constructor to ensure this class is abstract.

Definition at line 53 of file [BomAbstract.hpp](#).

32.12.2.2 `stdair::BomAbstract::BomAbstract ( const BomAbstract & ) [inline],[protected]`

Definition at line 54 of file [BomAbstract.hpp](#).

32.12.2.3 `virtual stdair::BomAbstract::~~BomAbstract ( ) [inline],[virtual]`

Destructor.

Definition at line 59 of file [BomAbstract.hpp](#).

### 32.12.3 Member Function Documentation

32.12.3.1 `virtual void stdair::BomAbstract::toStream ( std::ostream & ioOut ) const [pure virtual]`

Dump a Business Object into an output stream.

#### Parameters

<i>ostream&amp;</i>	The input/output stream.
---------------------	--------------------------

Implemented in [stdair::LegCabin](#), [stdair::SegmentSnapshotTable](#), [stdair::SegmentDate](#), [stdair::BookingClass](#), [stdair::SegmentCabin](#), [stdair::LegDate](#), [stdair::OnDDate](#), [stdair::FlightDate](#), [stdair::Inventory](#), [stdair::Policy](#), [stdair::BomRoot](#), [stdair::FareFamily](#), [stdair::Bucket](#), [stdair::SegmentPeriod](#), [stdair::AirlineClassList](#), [stdair::NestingNode](#), [stdair::SimpleNestingStructure](#), [stdair::BomHolder< BOM >](#), [stdair::FlightPeriod](#), [stdair::AirlineFeature](#), [stdair::PosChannel](#), [stdair::TimePeriod](#), [stdair::YieldFeatures](#), [stdair::AirportPair](#), [stdair::DatePeriod](#), [stdair::FareFeatures](#), and [stdair::YieldStore](#).

32.12.3.2 `virtual void stdair::BomAbstract::fromStream ( std::istream & ioIn ) [pure virtual]`

Read a Business Object from an input stream.

#### Parameters

<i>istream&amp;</i>	The input stream.
---------------------	-------------------

Implemented in [stdair::LegCabin](#), [stdair::SegmentSnapshotTable](#), [stdair::SegmentDate](#), [stdair::BookingClass](#), [stdair::SegmentCabin](#), [stdair::LegDate](#), [stdair::OnDDate](#), [stdair::FlightDate](#), [stdair::Inventory](#), [stdair::Policy](#), [stdair::BomRoot](#), [stdair::FareFamily](#), [stdair::Bucket](#), [stdair::AirlineClassList](#), [stdair::SegmentPeriod](#), [stdair::NestingNode](#), [stdair::SimpleNestingStructure](#), [stdair::BomHolder< BOM >](#), [stdair::AirlineFeature](#), [stdair::FlightPeriod](#), [stdair::PosChannel](#), [stdair::TimePeriod](#), [stdair::YieldFeatures](#), [stdair::AirportPair](#), [stdair::DatePeriod](#), [stdair::FareFeatures](#), and [stdair::YieldStore](#).

Referenced by [operator>>\(\)](#).

32.12.3.3 `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::LegCabin](#), [stdair::SegmentSnapshotTable](#), [stdair::SegmentDate](#), [stdair::BookingClass](#), [stdair::SegmentCabin](#), [stdair::LegDate](#), [stdair::OnDDate](#), [stdair::FlightDate](#), [stdair::Inventory](#), [stdair::Policy](#), [stdair::BomRoot](#), [stdair::FareFamily](#), [stdair::Bucket](#), [stdair::AirlineClassList](#), [stdair::SegmentPeriod](#), [stdair::NestingNode](#), [stdair::SimpleNestingStructure](#), [stdair::BomHolder< BOM >](#), [stdair::AirlineFeature](#), [stdair::FlightPeriod](#), [stdair::PosChannel](#), [stdair::TimePeriod](#), [stdair::YieldFeatures](#), [stdair::AirportPair](#), [stdair::DatePeriod](#), [stdair::FareFeatures](#), and [stdair::YieldStore](#).

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

- [stdair/bom/BomAbstract.hpp](#)

## 32.13 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](#) &)

### 32.13.1 Detailed Description

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

Definition at line 28 of file [BomArchive.hpp](#).

### 32.13.2 Member Function Documentation

**32.13.2.1** void stdair::BomArchive::archive ( const [BomRoot](#) & *iBomRoot* ) [static]

Recursively archive (dump in the underlying STL string) the objects of the BOM tree.

Parameters

<i>const</i>	<a href="#">BomRoot</a> & Root of the BOM tree to be archived.
--------------	--

Definition at line 32 of file [BomArchive.cpp](#).

**32.13.2.2** std::string stdair::BomArchive::archive ( const [Inventory](#) & *iInventory* ) [static]

Recursively archive (dump in the underlying STL string) the objects of the BOM tree.

Parameters

<i>const</i>	<a href="#">Inventory</a> & Root of the BOM tree to be archived.
--------------	--

Definition at line 36 of file [BomArchive.cpp](#).

**32.13.2.3** void stdair::BomArchive::restore ( const std::string & *iArchive*, [Inventory](#) & *ioInventory* ) [static]

Recursively restore (from the underlying STL string) the objects of the BOM tree.

Parameters

<i>const</i>	std::string& String holding the serialised objects.
<i>Inventory</i> &	Root of the BOM tree to be restored.

Definition at line 44 of file [BomArchive.cpp](#).

**32.13.2.4** void stdair::BomArchive::archive ( const [FlightDate](#) & *iFlightDate* ) [static]

Recursively archive (dump in the underlying STL string) the objects of the BOM tree.

Parameters



<i>const</i>	<a href="#">FlightDate</a> & 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](#)

## 32.14 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](#) &)

### 32.14.1 Detailed Description

Utility class to display StdAir objects with a pretty format.

Definition at line 38 of file [BomDisplay.hpp](#).

### 32.14.2 Member Function Documentation

32.14.2.1 `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

<i>std::ostream&amp;</i>	Output stream in which the flight-date keys should be logged/dumped.
<i>const</i>	<a href="#">BomRoot</a> & Root of the BOM tree to be displayed.
<i>const</i>	AirlineCode& Airline for which the flight-dates should be displayed. If set to "all" (default), all the inventories will be displayed.
<i>const</i>	FlightNumber_T& Flight number for which all the departure dates should be displayed. If set to 0 (the default), all the flight numbers will be displayed.

Referenced by [stdair::STDAIR\\_Service::list\(\)](#).

32.14.2.2 `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

<i>std::ostream&amp;</i>	Output stream in which the flight-date keys should be logged/dumped.
<i>const</i>	<a href="#">Inventory</a> & Root of the BOM tree to be displayed.
<i>const</i>	unsigned short Index, within the list, of the inventory. It is 0 when that inventory is displayed alone.
<i>const</i>	FlightNumber_T& Flight number for which all the departure dates should be displayed. If set to 0 (the default), all the flight numbers will be displayed.

32.14.2.3 `static void stdair::BomDisplay::listAirportPairDateRange ( std::ostream & , const BomRoot & ) [static]`

Display the list of airports pairs and date ranges (contained within the BOM tree)

## Parameters

<i>std::ostream&amp;</i>	Output stream in which the airport pairs and date ranges are logged/dumped.
<i>const</i>	<a href="#">BomRoot</a> & Root of the BOM tree to be displayed.

Referenced by [stdair::STDAIR\\_Service::listAirportPairDateRange\(\)](#).

32.14.2.4 `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

<i>std::ostream&amp;</i>	Output stream in which the BOM tree should be logged/dumped.
<i>const</i>	<a href="#">BomRoot</a> & Root of the BOM tree to be displayed.

Referenced by [stdair::STDAIR\\_Service::csvDisplay\(\)](#).

32.14.2.5 `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

<i>std::ostream&amp;</i>	Output stream in which the BOM tree should be logged/dumped.
<i>const</i>	<a href="#">Inventory</a> & Root of the BOM tree to be displayed.

32.14.2.6 `static void stdair::BomDisplay::csvDisplay ( std::ostream & , const OnDDate & ) [static]`

Display the O&D date object information.

## Parameters

<i>std::ostream&amp;</i>	Output stream in which the BOM tree should be logged/dumped.
<i>const</i>	<a href="#">OnDDate</a> & the BOM to be displayed.

32.14.2.7 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

<i>std::ostream&amp;</i>	Output stream in which the BOM tree should be logged/dumped.
<i>const</i>	<a href="#">FlightDate</a> & Root of the BOM tree to be displayed.

32.14.2.8 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

<i>std::ostream&amp;</i>	Output stream in which the BOM tree should be logged/dumped.
<i>const</i>	<a href="#">FlightDate</a> & Root of the BOM tree to be displayed.

32.14.2.9 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

<i>std::ostream&amp;</i>	Output stream in which the BOM tree should be logged/dumped.
<i>const</i>	<a href="#">FlightDate</a> & Root of the BOM tree to be displayed.

32.14.2.10 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

<i>std::ostream&amp;</i>	Output stream in which the BOM tree should be logged/dumped.
<i>const</i>	<a href="#">FlightDate</a> & Root of the BOM tree to be displayed.

32.14.2.11 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

<i>std::ostream&amp;</i>	Output stream in which the BOM tree should be logged/dumped.
<i>const</i>	<a href="#">FlightDate</a> & Root of the BOM tree to be displayed.

32.14.2.12 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

<i>std::ostream&amp;</i>	Output stream in which the BOM tree should be logged/dumped.
<i>const</i>	<a href="#">FlightDate</a> & Root of the BOM tree to be displayed.

32.14.2.13 `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

<i>std::ostream&amp;</i>	Output stream in which the BOM tree should be logged/dumped.
<i>const</i>	<a href="#">FlightDate</a> & Root of the BOM tree to be displayed.

32.14.2.14 `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

<i>std::ostream&amp;</i>	Output stream in which the BOM tree should be logged/dumped.
<i>const</i>	<a href="#">BookingClass</a> & Root of the BOM tree to be displayed.
<i>const</i>	std::string& Leading string to be displayed.

32.14.2.15 `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

<i>std::ostream&amp;</i>	Output stream in which the BOM tree should be logged/dumped.
<i>const</i>	<a href="#">FlightDate</a> & Root of the BOM tree to be displayed.

32.14.2.16 `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

<i>std::ostream&amp;</i>	Output stream in which the list of travel solutions is logged/dumped.
<i>TravelSolutionList_T&amp;</i>	List of travel solutions to display.

32.14.2.17 `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

<i>std::ostream&amp;</i>	Output stream in which the list of travel solutions is logged/dumped.
<i>DatePeriodList_T&amp;</i>	List of date period to display.

32.14.2.18 `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

<i>std::ostream&amp;</i>	Output stream in which the BOM tree should be logged/dumped.
<i>const</i>	<a href="#">BomRoot</a> & Root of the BOM tree to be displayed.

32.14.2.19 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

<i>std::ostream&amp;</i>	Output stream in which the BOM tree should be logged/dumped.
<i>const</i>	<a href="#">AirportPair</a> & Root of the BOM tree to be displayed.

32.14.2.20 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

<i>std::ostream&amp;</i>	Output stream in which the BOM tree should be logged/dumped.
<i>const</i>	<a href="#">DatePeriod</a> & Root of the BOM tree to be displayed.

32.14.2.21 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

<i>std::ostream&amp;</i>	Output stream in which the BOM tree should be logged/dumped.
<i>const</i>	<a href="#">PosChannel</a> & Root of the BOM tree to be displayed.

32.14.2.22 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

<i>std::ostream&amp;</i>	Output stream in which the BOM tree should be logged/dumped.
<i>const</i>	<a href="#">TimePeriod</a> & Root of the BOM tree to be displayed.

32.14.2.23 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

<i>std::ostream&amp;</i>	Output stream in which the BOM tree should be logged/dumped.
<i>const</i>	<a href="#">TimePeriod</a> & Root of the BOM tree to be displayed.

32.14.2.24 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

<i>std::ostream&amp;</i>	Output stream in which the BOM tree should be logged/dumped.
<i>const</i>	FEATURE_TYPE& Root of the BOM tree to be displayed.

32.14.2.25 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

<i>std::ostream&amp;</i>	Output stream in which the BOM tree should be logged/dumped.
<i>const</i>	<a href="#">AirlineClassList</a> & Root of the BOM tree to be displayed.

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

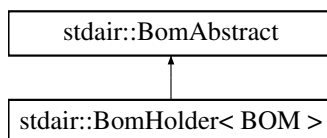
- [stdair/bom/BomDisplay.hpp](#)

## 32.15 stdair::BomHolder&lt; BOM &gt; 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

- `template<typename >`  
class [FacBom](#)
- class [FacBomManager](#)

#### 32.15.1 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](#).

#### 32.15.2 Member Typedef Documentation

32.15.2.1 `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](#).

32.15.2.2 `template<typename BOM> typedef std::list<BOM*> stdair::BomHolder< BOM >::BomList_T`

(STL) list of children.

Definition at line 39 of file [BomHolder.hpp](#).

32.15.2.3 `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](#).

#### 32.15.3 Constructor & Destructor Documentation

32.15.3.1 `template<typename BOM> stdair::BomHolder< BOM >::BomHolder ( ) [protected]`

Constructor.

32.15.3.2 `template<typename BOM> stdair::BomHolder< BOM >::BomHolder ( const BomHolder< BOM > & ) [protected]`

Copy constructor.

32.15.3.3 `template<typename BOM> stdair::BomHolder< BOM >::BomHolder ( const Key_T & iKey ) [inline], [protected]`

Main constructor.

Definition at line 94 of file [BomHolder.hpp](#).



32.15.3.4 `template<typename BOM> stdair::BomHolder< BOM >::~~BomHolder ( ) [inline],  
[protected]`

Destructor.

Definition at line 99 of file [BomHolder.hpp](#).

#### 32.15.4 Member Function Documentation

32.15.4.1 `template<typename BOM> void stdair::BomHolder< BOM >::toStream ( std::ostream & ioOut ) const  
[inline], [virtual]`

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Implements [stdair::BomAbstract](#).

Definition at line 54 of file [BomHolder.hpp](#).

References [stdair::BomHolder< BOM >::toString\(\)](#).

32.15.4.2 `template<typename BOM> void stdair::BomHolder< BOM >::fromStream ( std::istream & ioIn )  
[inline], [virtual]`

Read a Business Object from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Implements [stdair::BomAbstract](#).

Definition at line 63 of file [BomHolder.hpp](#).

32.15.4.3 `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\(\)](#).

32.15.4.4 `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](#).

#### 32.15.5 Friends And Related Function Documentation

32.15.5.1 `template<typename BOM> template<typename > friend class FacBom [friend]`

Friend classes.

Definition at line 26 of file [BomHolder.hpp](#).

32.15.5.2 `template<typename BOM> friend class FacBomManager` [*friend*]

Definition at line 27 of file [BomHolder.hpp](#).

### 32.15.6 Member Data Documentation

32.15.6.1 `template<typename BOM> Key_T stdair::BomHolder< BOM >::_key`

Key.

Definition at line 99 of file [BomHolder.hpp](#).

32.15.6.2 `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\(\)](#), [stdair::FacBomManager::resetYieldBasedNestingStructure\(\)](#), and [stdair::serialiseHelper\(\)](#).

32.15.6.3 `template<typename BOM> BomMap_T stdair::BomHolder< BOM >::_bomMap`

(STL) map of children.

Definition at line 116 of file [BomHolder.hpp](#).

Referenced by [stdair::FacBomManager::cloneHolder\(\)](#), [stdair::BomManager::getMap\(\)](#), [stdair::BomManager::getObjectPtr\(\)](#), [stdair::BomManager::hasMap\(\)](#), and [stdair::serialiseHelper\(\)](#).

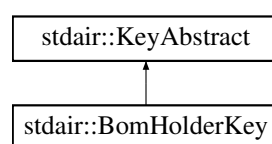
The documentation for this class was generated from the following file:

- [stdair/bom/BomHolder.hpp](#)

## 32.16 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

### 32.16.1 Detailed Description

Key of the BOM structure holder.

Definition at line 12 of file [BomHolderKey.hpp](#).

### 32.16.2 Constructor & Destructor Documentation

#### 32.16.2.1 `stdair::BomHolderKey::BomHolderKey ( )`

Constructor.

Definition at line 13 of file [BomHolderKey.cpp](#).

#### 32.16.2.2 `stdair::BomHolderKey::~~BomHolderKey ( )`

Destructor.

Definition at line 17 of file [BomHolderKey.cpp](#).

### 32.16.3 Member Function Documentation

#### 32.16.3.1 `void stdair::BomHolderKey::toStream ( std::ostream & ioOut ) const` [virtual]

Dump a Business Object Key into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 21 of file [BomHolderKey.cpp](#).

References [toString\(\)](#).

#### 32.16.3.2 `void stdair::BomHolderKey::fromStream ( std::istream & ioln )` [virtual]

Read a Business Object Key from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 26 of file [BomHolderKey.cpp](#).

#### 32.16.3.3 `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\(\)](#).

#### 32.16.3.4 `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](#)

## 32.17 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](#) ()

### 32.17.1 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](#).

### 32.17.2 Constructor & Destructor Documentation

32.17.2.1 `template<typename BOM > stdair::BomID< BOM >::BomID ( BOM & iBOM )`

Main constructor.

Definition at line 58 of file [BomID.hpp](#).

32.17.2.2 `template<typename BOM > stdair::BomID< BOM >::BomID ( const BomID< BOM > & iBomID )`

Copy constructor.

Definition at line 61 of file [BomID.hpp](#).

32.17.2.3 `template<typename BOM > stdair::BomID< BOM >::~~BomID ( )`

Destructor.

Definition at line 65 of file [BomID.hpp](#).

### 32.17.3 Member Function Documentation

32.17.3.1 `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](#)

## 32.18 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](#) &)

### 32.18.1 Detailed Description

Utility class to import StdAir objects in a INI format.

Definition at line 21 of file [BomINIImport.hpp](#).

### 32.18.2 Member Function Documentation

**32.18.2.1** void [stdair::BomINIImport::importINIConfig](#) ( [ConfigHolderStruct](#) & *iConfigHolder*, const [ConfigINIFile](#) & *iConfigINIFile* ) [static]

Extract a boost property tree from an INI config file.

#### Parameters

<i>ConfigHolderStruct</i> &	Holder of the configuration tree.
const	<a href="#">ConfigINIFile</a> & INI config file.

Definition at line 29 of file [BomINIImport.cpp](#).

References [stdair::ConfigHolderStruct::add\(\)](#), [stdair::BasFileMgr::doesExistAndIsReadable\(\)](#), [stdair::RootFile↔Path::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](#)

## 32.19 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](#) & *iAirline↔Code*="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](#) &)

### 32.19.1 Detailed Description

Utility class to export StdAir objects in a JSON format.

Definition at line 42 of file [BomJSONExport.hpp](#).

### 32.19.2 Member Function Documentation

32.19.2.1 void stdair::BomJSONExport::jsonExportFlightDateList ( std::ostream & *oStream*, const BomRoot & *iBomRoot*, 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

<i>std::ostream&amp;</i>	Output stream in which the flight date objects should be logged/dumped.
<i>const</i>	<a href="#">BomRoot</a> & Root of the BOM tree containing flight-dates to be exported.
<i>const</i>	<a href="#">AirlineCode</a> & Airline for which the flight-dates should be displayed. If set to "all" (default), all the inventories will be displayed.
<i>const</i>	<a href="#">FlightNumber_T</a> & 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](#).

References [stdair::Inventory::getAirlineCode\(\)](#), [stdair::FlightDate::getDepartureDate\(\)](#), and [stdair::FlightDate::getFlightNumber\(\)](#).

Referenced by [stdair::STDAIR\\_Service::jsonExportFlightDateList\(\)](#).

**32.19.2.2** `void stdair::BomJSONExport::jsonExportFlightDateObjects ( std::ostream & oStream, const FlightDate & iFlightDate ) [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

<i>std::ostream&amp;</i>	Output stream in which the BOM tree should be logged/dumped.
<i>const</i>	<a href="#">FlightDate</a> & 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\(\)](#).

**32.19.2.3** `void stdair::BomJSONExport::jsonExportBookingRequestObject ( std::ostream & oStream, const EventStruct & iEventStruct ) [static]`

Export (dump in the underlying output log stream and in JSON format) the booking request object contained in the event structure.

## Parameters

<i>std::ostream&amp;</i>	Output stream in which the events should be logged/dumped.
<i>const</i>	<a href="#">EventStruct</a> & 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::getPreferredDepartureDate\(\)](#), [stdair::BookingRequestStruct::getPreferredCabin\(\)](#), [stdair::BookingRequestStruct::getPreferredDepartureTime\(\)](#), [stdair::BookingRequestStruct::getRequestDateTime\(\)](#), [stdair::BookingRequestStruct::getStayDuration\(\)](#), and [stdair::BookingRequestStruct::getWTP\(\)](#).

Referenced by [stdair::STDAIR\\_Service::jsonExportEventObject\(\)](#).

**32.19.2.4** `void stdair::BomJSONExport::jsonExportBreakPointObject ( std::ostream & oStream, const EventStruct & iEventStruct ) [static]`

Export (dump in the underlying output log stream and in JSON format) the break point object contained in the event structure.

## Parameters

<i>std::ostream</i> &	Output stream in which the events should be logged/dumped.
<i>const</i>	<a href="#">EventStruct</a> & 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](#)

## 32.20 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](#) &)

### 32.20.1 Detailed Description

Utility class to import StdAir objects in a JSON format.

Definition at line 26 of file [BomJSONImport.hpp](#).

### 32.20.2 Member Function Documentation

32.20.2.1 bool [stdair::BomJSONImport::jsonImportCommand](#) ( const [JSONString](#) & *iBomJSONStr*, [JSonCommand::EN\\_JSonCommand](#) & *ioEnumJSoCommand* ) [static]

Extract the JSON command from a given JSON-formatted string.

## Parameters

<i>const</i>	<a href="#">JSONString</a> & JSON-formatted string.
<i>JSonCommand::EN_JSonCommand</i> &	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::JSoCommand::getCommand\(\)](#), and [stdair::JSONString::getString\(\)](#).

**32.20.2.2** `bool stdair::BomJSONImport::jsonImportInventoryKey ( const JSONString & iBomJSONStr, AirlineCode_T & ioAirlineCode ) [static]`

Extract the airline code from a given JSON-formatted string.

**Parameters**

<i>const</i>	<a href="#">JSONString</a> & JSON-formatted string.
<i>AirlineCode_T</i> &	Airline code extracted from the given string.

**Returns**

bool State whether the extracting has been successful.

Definition at line 98 of file [BomJSONImport.cpp](#).

References [stdair::JSONString::getString\(\)](#).

**32.20.2.3** `bool stdair::BomJSONImport::jsonImportFlightDate ( const JSONString & iBomJSONStr, Date_T & ioDepartureDate ) [static]`

Extract the [FlightDate](#) from a given JSON-formatted string.

**Parameters**

<i>const</i>	<a href="#">JSONString</a> & JSON-formatted string.
<i>Date_T</i> &	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\(\)](#).

**32.20.2.4** `bool stdair::BomJSONImport::jsonImportFlightNumber ( const JSONString & iBomJSONStr, FlightNumber_T & ioFlightNumber ) [static]`

Extract the FlightNumber from a given JSON-formatted string.

**Parameters**

<i>const</i>	<a href="#">JSONString</a> & JSON-formatted string.
<i>FlightNumber_↔ T</i> &	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\(\)](#).

32.20.2.5 `bool stdair::BomJSONImport::jsonImportBreakPoints ( const JSONString & iBomJSONStr, BreakPointList_T & oBreakPointList ) [static]`

Extract the break points from a given JSON-formatted string.

## Parameters

<i>const</i>	<a href="#">JSONString</a> & JSON-formatted string.
<i>BreakPointList</i> ↔ <i>_T&amp;</i>	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\(\)](#).

32.20.2.6 `bool stdair::BomJSONImport::jsonImportEventType ( const JSONString & iBomJSONStr, EventType::EN_EventType & ioEventType ) [static]`

Extract the event type from a given JSON-formatted string.

## Parameters

<i>const</i>	<a href="#">JSONString</a> & JSON-formatted string.
<i>EventType::EN</i> ↔ <i>_EventType&amp;</i>	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\(\)](#).

32.20.2.7 `bool stdair::BomJSONImport::jsonImportConfig ( const JSONString & iBomJSONStr, ConfigHolderStruct & iConfigHolderStruct ) [static]`

Extract the configuration ptree from the given JSON-formatted string and add it to the configuration holder

## Parameters

<i>const</i>	<a href="#">JSONString</a> & JSON-formatted string.
<i>ConfigHolder</i> ↔ <i>Struct&amp;</i>	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](#)

## 32.21 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)

## 32.21.1 Detailed Description

Utility class to extract key structures from strings.

Definition at line 29 of file [BomKeyManager.hpp](#).

## 32.21.2 Member Function Documentation

32.21.2.1 [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\(\)](#).

32.21.2.2 [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

<i>const</i>	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\(\)](#).

32.21.2.3 [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

<i>const</i>	<code>std::string&amp;</code> 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\(\)](#).

**32.21.2.4 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

<i>const</i>	<code>std::string&amp;</code> 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\(\)](#).

**32.21.2.5 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

<i>const</i>	<code>std::string&amp;</code> 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](#)

## 32.22 stdair::BomManager Class Reference

Utility class for StdAir-based objects.

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

### Public Member Functions

- template<typename OBJECT2 , typename OBJECT1 >  
const [BomHolder](#)< OBJECT2 >::BomList\_T & [getList](#) (const OBJECT1 &iObject1)
- template<typename OBJECT2 , typename OBJECT1 >  
const [BomHolder](#)< OBJECT2 >::BomMap\_T & [getMap](#) (const OBJECT1 &iObject1)
- 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](#) &iInventory)
- template<>  
bool [hasMap](#) (const [Inventory](#) &iInventory)
- 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](#)

#### 32.22.1 Detailed Description

Utility class for StdAir-based objects.

Most of those methods work for objects specified and instantiated outside StdAir, as long as those objects inherit from StdAir objects.

Definition at line 34 of file [BomManager.hpp](#).

## 32.22.2 Member Function Documentation

**32.22.2.1** `template<typename OBJECT2 , typename OBJECT1 > static const BomHolder<OBJECT2>::BomList_T& stdair::BomManager::getList ( const OBJECT1 & ) [static]`

Get the container (STL list) of OBJECT2 objects within the OBJECT1 object.

**32.22.2.2** `template<typename OBJECT2 , typename OBJECT1 > static const BomHolder<OBJECT2>::BomMap_T& stdair::BomManager::getMap ( const OBJECT1 & ) [static]`

Get the container (STL map) of OBJECT2 objects within the OBJECT1 object.

**32.22.2.3** `template<typename OBJECT2 , typename OBJECT1 > bool stdair::BomManager::hasList ( const OBJECT1 & iObject1 ) [static]`

Check if the list of object2 has been initialised.

Definition at line 181 of file [BomManager.hpp](#).

References [stdair::BomHolder< BOM >::\\_bomList](#).

**32.22.2.4** `template<typename OBJECT2 , typename OBJECT1 > bool stdair::BomManager::hasMap ( const OBJECT1 & iObject1 ) [static]`

Check if the map of object2 has been initialised.

Definition at line 201 of file [BomManager.hpp](#).

References [stdair::BomHolder< BOM >::\\_bomMap](#).

**32.22.2.5** `template<typename PARENT , typename CHILD > PARENT * stdair::BomManager::getParentPtr ( const CHILD & iChild ) [static]`

Get the PARENT of the given CHILD.

If the types do not match, NULL is returned.

Definition at line 220 of file [BomManager.hpp](#).

**32.22.2.6** `template<typename PARENT , typename CHILD > PARENT & stdair::BomManager::getParent ( const CHILD & iChild ) [static]`

Get the PARENT of the given CHILD.

Definition at line 230 of file [BomManager.hpp](#).

**32.22.2.7** `template<typename OBJECT2 , typename OBJECT1 > OBJECT2 * stdair::BomManager::getObjectPtr ( const OBJECT1 & iObject1, const MapKey_T & iKey ) [static]`

Get the OBJECT2 pointer corresponding to the given string key.

If such a OBJECT2 does not exist, return NULL.

Definition at line 241 of file [BomManager.hpp](#).

References [stdair::BomHolder< BOM >::\\_bomMap](#).

**32.22.2.8** `template<typename OBJECT2 , typename OBJECT1 > OBJECT2 & stdair::BomManager::getObject ( const OBJECT1 & iObject1, const MapKey_T & iKey ) [static]`

Get the OBJECT2 corresponding to the given string key.

Definition at line 283 of file [BomManager.hpp](#).

References [STDAIR\\_LOG\\_ERROR](#).

32.22.2.9 `template<typename OBJECT2 , typename OBJECT1 > const BomHolder<OBJECT2>::BomList_T& stdair::BomManager::getList ( const OBJECT1 & iObject1 )`

Definition at line 140 of file [BomManager.hpp](#).

References [stdair::BomHolder< BOM >::\\_bomList](#).

32.22.2.10 `template<typename OBJECT2 , typename OBJECT1 > const BomHolder<OBJECT2>::BomMap_T& stdair::BomManager::getMap ( const OBJECT1 & iObject1 )`

Definition at line 159 of file [BomManager.hpp](#).

References [stdair::BomHolder< BOM >::\\_bomMap](#).

32.22.2.11 `template<> bool stdair::BomManager::hasList ( const SegmentDate & ioSegmentDate ) [inline]`

Definition at line 329 of file [BomManager.hpp](#).

32.22.2.12 `template<> const BomHolder<SegmentDate>::BomList_T& stdair::BomManager::getList ( const SegmentDate & ioSegmentDate ) [inline]`

Definition at line 345 of file [BomManager.hpp](#).

32.22.2.13 `template<> bool stdair::BomManager::hasMap ( const SegmentDate & ioSegmentDate ) [inline]`

Definition at line 358 of file [BomManager.hpp](#).

32.22.2.14 `template<> bool stdair::BomManager::hasList ( const Inventory & ioInventory ) [inline]`

Definition at line 375 of file [BomManager.hpp](#).

32.22.2.15 `template<> bool stdair::BomManager::hasMap ( const Inventory & ioInventory ) [inline]`

Definition at line 385 of file [BomManager.hpp](#).

32.22.2.16 `template<> AirlineFeature* stdair::BomManager::getObjectPtr ( const Inventory & iInventory, const MapKey_T & iKey ) [inline]`

Definition at line 395 of file [BomManager.hpp](#).

References [stdair::Inventory::getAirlineFeature\(\)](#).

32.22.2.17 `template<> AirlineFeature& stdair::BomManager::getObject ( const Inventory & iInventory, const MapKey_T & iKey ) [inline]`

Definition at line 406 of file [BomManager.hpp](#).

## 32.22.3 Friends And Related Function Documentation

32.22.3.1 `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](#)

## 32.23 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](#) &)

### 32.23.1 Detailed Description

Utility class to retrieve StdAir objects.

Definition at line 36 of file [BomRetriever.hpp](#).

### 32.23.2 Member Function Documentation

#### 32.23.2.1 [Inventory](#) \* [stdair::BomRetriever::retrieveInventoryFromLongKey](#) ( const [BomRoot](#) & *iBomRoot*, const std::string & *iFullKeyStr* ) [static]

Retrieve an [Inventory](#) object from a (full) key string.

The full key string gathers airline code, flight number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

## Parameters

<i>const</i>	<a href="#">BomRoot</a> & The root of the BOM tree.
<i>const</i>	std::string& The full key string.

## Returns

Inventory\* The just retrieved [Inventory](#) object.

Definition at line 31 of file [BomRetriever.cpp](#).

References [stdair::BomKeyManager::extractInventoryKey\(\)](#), and [stdair::BomRoot::getInventory\(\)](#).

Referenced by [retrieveFlightDateFromLongKey\(\)](#), and [retrievePartnerSegmentDateFromLongKey\(\)](#).

**32.23.2.2** `Inventory * stdair::BomRetriever::retrieveInventoryFromLongKey ( const Inventory & iInventory, const std::string & iFullKeyStr ) [static]`

Retrieve an [Inventory](#) object from a (full) key string.

The full key string gathers airline code, flight number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

## Parameters

<i>const</i>	<a href="#">Inventory</a> & The root of the BOM tree.
<i>const</i>	std::string& The full key string.

## Returns

Inventory\* The just retrieved [Inventory](#) object.

Definition at line 46 of file [BomRetriever.cpp](#).

References [stdair::BomKeyManager::extractInventoryKey\(\)](#), and [stdair::InventoryKey::getAirlineCode\(\)](#).

**32.23.2.3** `Inventory * stdair::BomRetriever::retrieveInventoryFromKey ( const BomRoot & iBomRoot, const InventoryKey & iKey ) [static]`

Retrieve an [Inventory](#) object from an [InventoryKey](#) structure.

## Parameters

<i>const</i>	<a href="#">BomRoot</a> & The root of the BOM tree.
<i>const</i>	<a href="#">InventoryKey</a> & 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\(\)](#).

**32.23.2.4** `Inventory * stdair::BomRetriever::retrieveInventoryFromKey ( const BomRoot & iBomRoot, const AirlineCode_T & iAirlineCode ) [static]`

Retrieve an [Inventory](#) object from an [InventoryKey](#) structure.

## Parameters

<i>const</i>	<a href="#">BomRoot</a> & The root of the BOM tree.
<i>const</i>	<a href="#">AirlineCode_T</a> & The key.

## Returns

[Inventory](#)\* The just retrieved [Inventory](#) object.

Definition at line 75 of file [BomRetriever.cpp](#).

References [stdair::BomRoot::getInventory\(\)](#).

**32.23.2.5 [AirlineFeature](#) \* stdair::BomRetriever::retrieveAirlineFeatureFromKey ( [const BomRoot](#) & *iBomRoot*, [const AirlineCode\\_T](#) & *iAirlineCode* ) [static]**

Retrieve an Airline Feature object from an airline code.

## Parameters

<i>const</i>	<a href="#">BomRoot</a> & The root of the BOM tree.
<i>const</i>	<a href="#">AirlineCode_T</a> & 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\(\)](#).

**32.23.2.6 [FlightDate](#) \* stdair::BomRetriever::retrieveFlightDateFromLongKey ( [const BomRoot](#) & *iBomRoot*, [const std::string](#) & *iFullKeyStr* ) [static]**

Retrieve a [FlightDate](#) object from a (full) key string.

The full key string gathers airline code, flight number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

## Parameters

<i>const</i>	<a href="#">BomRoot</a> & The root of the BOM tree.
<i>const</i>	<a href="#">std::string</a> & 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\(\)](#).

**32.23.2.7 [FlightDate](#) \* stdair::BomRetriever::retrieveFlightDateFromKeySet ( [const BomRoot](#) & *iBomRoot*, [const AirlineCode\\_T](#) & *iAirlineCode*, [const FlightNumber\\_T](#) & *iFlightNumber*, [const Date\\_T](#) & *iFlightDateDate* ) [static]**

Retrieve a [FlightDate](#) object from a set of keys.

## Parameters

<i>const</i>	<a href="#">BomRoot</a> & The root of the BOM tree.
<i>const</i>	<a href="#">AirlineCode_T</a> & The key.
<i>const</i>	<a href="#">FlightNumber_T</a> & Part of the key.
<i>const</i>	<a href="#">Date_T</a> & 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\(\)](#).

**32.23.2.8** [FlightDate](#) \* [stdair::BomRetriever::retrieveFlightDateFromLongKey](#) ( *const* [Inventory](#) & *ilinventory*, *const* [std::string](#) & *iFullKeyStr* ) [static]

Retrieve a [FlightDate](#) object from a (full) key string.

The full key string gathers airline code, flight number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

## Parameters

<i>const</i>	<a href="#">Inventory</a> & The root of the BOM tree.
<i>const</i>	<a href="#">std::string</a> & 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\(\)](#).

**32.23.2.9** [FlightDate](#) \* [stdair::BomRetriever::retrieveFlightDateFromKey](#) ( *const* [Inventory](#) & *ilinventory*, *const* [FlightDateKey](#) & *iKey* ) [static]

Retrieve a [FlightDate](#) object from an [FlightDateKey](#) structure.

## Parameters

<i>const</i>	<a href="#">Inventory</a> & The root of the BOM tree.
<i>const</i>	<a href="#">FlightDateKey</a> & 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\(\)](#).

**32.23.2.10** [FlightDate](#) \* [stdair::BomRetriever::retrieveFlightDateFromKey](#) ( *const* [Inventory](#) & *ilinventory*, *const* [FlightNumber\\_T](#) & *iFlightNumber*, *const* [Date\\_T](#) & *iFlightDateDate* ) [static]

Retrieve a [FlightDate](#) object from an [FlightDateKey](#) structure.

## Parameters

<i>const</i>	<a href="#">Inventory</a> & The root of the BOM tree.
<i>const</i>	FlightNumber_T& Part of the key.
<i>const</i>	Date_T& Part of the key.

## Returns

FlightDate\* The just retrieved [FlightDate](#) object.

Definition at line 182 of file [BomRetriever.cpp](#).

References [stdair::Inventory::getFlightDate\(\)](#).

**32.23.2.11** `LegDate * stdair::BomRetriever::retrieveOperatingLegDateFromLongKey ( const FlightDate & iFlightDate, 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

<i>const</i>	<a href="#">FlightDate</a> & The root of the BOM tree.
<i>const</i>	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\(\)](#).

**32.23.2.12** `SegmentDate * stdair::BomRetriever::retrievePartnerSegmentDateFromLongKey ( const Inventory & iInventory, 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

<i>const</i>	<a href="#">Inventory</a> & The root of the BOM tree.
<i>const</i>	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\(\)](#), [retrieve←InventoryFromLongKey\(\)](#), and [retrieveSegmentDateFromLongKey\(\)](#).

**32.23.2.13** `SegmentDate * stdair::BomRetriever::retrieveSegmentDateFromLongKey ( const BomRoot & iBomRoot, const std::string & iFullKeyStr ) [static]`

Retrieve a [SegmentDate](#) object from a (full) key string.

The full key string gathers airline code, segment number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

## Parameters

<i>const</i>	<a href="#">BomRoot</a> & The root of the BOM tree.
<i>const</i>	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\(\)](#).

**32.23.2.14** `SegmentDate * stdair::BomRetriever::retrieveSegmentDateFromLongKey ( const Inventory & iInventory, const std::string & iFullKeyStr ) [static]`

Retrieve a [SegmentDate](#) object from a (full) key string.

The full key string gathers airline code, segment number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

## Parameters

<i>const</i>	<a href="#">Inventory</a> & The root of the BOM tree.
<i>const</i>	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\(\)](#).

**32.23.2.15** `SegmentDate * stdair::BomRetriever::retrieveSegmentDateFromLongKey ( const FlightDate & iFlightDate, const std::string & iFullKeyStr ) [static]`

Retrieve a [SegmentDate](#) object from a (full) key string.

The full key string gathers airline code, segment number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

## Parameters

<i>const</i>	<a href="#">FlightDate</a> & The root of the BOM tree.
<i>const</i>	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\(\)](#).

**32.23.2.16** `SegmentDate * stdair::BomRetriever::retrieveSegmentDateFromKey ( const FlightDate & iFlightDate, const SegmentDateKey & iKey ) [static]`

Retrieve a [SegmentDate](#) object from an [SegmentDateKey](#) structure.

## Parameters

<i>const</i>	<a href="#">FlightDate</a> & The root of the BOM tree.
<i>const</i>	<a href="#">SegmentDateKey</a> & The key.

## Returns

[SegmentDate](#)\* The just retrieved [SegmentDate](#) object.

Definition at line 307 of file [BomRetriever.cpp](#).

References [stdair::FlightDate::getSegmentDate\(\)](#).

Referenced by [retrieveSegmentDateFromLongKey\(\)](#).

**32.23.2.17** [SegmentDate](#) \* [stdair::BomRetriever::retrieveSegmentDateFromKey](#) ( *const* [FlightDate](#) & *iFlightDate*, *const* [AirportCode\\_T](#) & *iOrigin*, *const* [AirportCode\\_T](#) & *iDestination* ) [static]

Retrieve a [SegmentDate](#) object from an [SegmentDateKey](#) structure.

## Parameters

<i>const</i>	<a href="#">FlightDate</a> & The root of the BOM tree.
<i>const</i>	<a href="#">AirportCode_T</a> & Origin, part of the key.
<i>const</i>	<a href="#">AirportCode_T</a> & Destination, part of the key.

## Returns

[SegmentDate](#)\* The just retrieved [SegmentDate](#) object.

Definition at line 319 of file [BomRetriever.cpp](#).

References [stdair::FlightDate::getSegmentDate\(\)](#).

**32.23.2.18** [BookingClass](#) \* [stdair::BomRetriever::retrieveBookingClassFromLongKey](#) ( *const* [Inventory](#) & *iInventory*, *const* *std::string* & *iFullKeyStr*, *const* [ClassCode\\_T](#) & *iClassCode* ) [static]

Retrieve a [BookingClass](#) object from a (full) key string.

The full key string gathers airline code, segment number, origin and destination, cabin and booking class. It corresponds to the output generated by the `toString()` methods of the `XxxKey` structures.

Besides being attached to segment-cabin objects (and fare family objects, when they exist), the booking-class objects must also be attached directly to the segment-date.

Hence, if an assertion fails within that method call, chances are that the booking-class objects have not been attached to the segment-date objects. Check, for instance, the `CmdBomManager::buildSampleBom()` to see how that should be properly done.

## Parameters

<i>const</i>	<a href="#">Inventory</a> & The root of the BOM tree.
<i>const</i>	<i>std::string</i> & Part of the full key string.
<i>const</i>	<a href="#">ClassCode_T</a> & Part of the full key string.

## Returns

[BookingClass](#)\* The just retrieved [BookingClass](#) object.

Definition at line 333 of file [BomRetriever.cpp](#).

References [retrieveSegmentDateFromLongKey\(\)](#).

32.23.2.19 **AirportPair** \* stdair::BomRetriever::retrieveAirportPairFromKeySet ( const BomRoot & *iBomRoot*, const stdair::AirportCode\_T & *iOrigin*, const stdair::AirportCode\_T & *iDestination* ) [static]

Retrieve an [AirportPair](#) object from an [AirportPair](#) structure.



## Parameters

<i>const</i>	<a href="#">BomRoot</a> & The root of the BOM tree.
<i>const</i>	<a href="#">AirportCode_T</a> & Origin, part of the key.
<i>const</i>	<a href="#">AirportCode_T</a> & 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\(\)](#).

```
32.23.2.20 void stdair::BomRetriever::retrieveDatePeriodListFromKey ( const AirportPair & iAirportPair, const
stdair::Date_T & iDepartureDate, stdair::DatePeriodList_T & ioDatePeriodList ) [static]
```

Retrieve a list of date-period corresponding to a flight date.

## Parameters

<i>const</i>	<a href="#">AirportPair</a> & The root of the BOM tree.
<i>const</i>	<a href="#">Date_T</a> & Departure Date of the flight
<i>stdair::Date_T</i> ↔ <i>PeriodList_T</i> &	List of <a href="#">DatePeriod</a> to display.

Definition at line 373 of file [BomRetriever.cpp](#).

References [stdair::DatePeriod::isDepartureDateValid\(\)](#).

Referenced by [retrieveDatePeriodListFromKeySet\(\)](#).

```
32.23.2.21 void stdair::BomRetriever::retrieveDatePeriodListFromKeySet ( const BomRoot & iBomRoot, const
stdair::AirportCode_T & iOrigin, const stdair::AirportCode_T & iDestination, const stdair::Date_T &
iDepartureDate, stdair::DatePeriodList_T & ioDatePeriodList ) [static]
```

Retrieve a list of date-period from a set of keys.

## Parameters

<i>const</i>	<a href="#">BomRoot</a> & The root of the BOM tree.
<i>const</i>	<a href="#">AirportCode_T</a> & Part of the <a href="#">AirportPair</a> key: the origin airport
<i>const</i>	<a href="#">AirportCode_T</a> & Part of the <a href="#">AirportPair</a> key: the destination airport.
<i>const</i>	<a href="#">Date_T</a> & Departure date of the flight
<i>stdair::Date_T</i> ↔ <i>PeriodList_T</i> &	List of <a href="#">DatePeriod</a> 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\(\)](#).

```
32.23.2.22 LegCabin & stdair::BomRetriever::retrieveDummyLegCabin ( stdair::BomRoot & iBomRoot, const bool
isForFareFamilies = false ) [static]
```

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

## Parameters

<i>stdair::BomRoot</i>	The BOM tree.
<i>const</i>	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\\_FLIGHT\\_ORIGIN](#), [stdair::LegDate::getLegCabin\(\)](#), [stdair::FlightDate::getLegDate\(\)](#), [retrieveFlightDateFromKey\(\)](#), and [retrieveInventoryFromKey\(\)](#).

**32.23.2.23 SegmentCabin & stdair::BomRetriever::retrieveDummySegmentCabin ( stdair::BomRoot & iBomRoot, const bool isForFareFamilies = false ) [static]**

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

Parameters

<i>stdair::BomRoot</i>	The BOM tree.
<i>const</i>	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](#), [stdair::FlightDate::getSegmentDate\(\)](#), [retrieveFlightDateFromKey\(\)](#), [retrieveInventoryFromKey\(\)](#), and [stdair::SegmentCabinKey::toString\(\)](#).

**32.23.2.24 std::string stdair::BomRetriever::retrieveFullKeyFromSegmentDate ( const SegmentDate & iSegmentdate ) [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

<i>const</i>	<a href="#">SegmentDate</a> & 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::Inventory::describeKey\(\)](#), and [stdair::SegmentDate::describeKey\(\)](#).

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

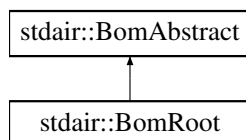
- [stdair/bom/BomRetriever.hpp](#)
- [stdair/bom/BomRetriever.cpp](#)

## 32.24 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

- template<typename BOM >  
class [FacBom](#)
- template<typename BOM >  
class [FacCloneBom](#)
- class [FacBomManager](#)
- class [boost::serialization::access](#)

### 32.24.1 Detailed Description

Class representing the actual attributes for the Bom root.

Definition at line 32 of file [BomRoot.hpp](#).

### 32.24.2 Member Typedef Documentation

#### 32.24.2.1 `typedef BomRootKey stdair::BomRoot::Key_T`

Definition allowing to retrieve the associated BOM key type.

Definition at line 42 of file [BomRoot.hpp](#).

### 32.24.3 Constructor & Destructor Documentation

#### 32.24.3.1 `stdair::BomRoot::BomRoot ( )` `[protected]`

Default constructor.

Definition at line 17 of file [BomRoot.cpp](#).

#### 32.24.3.2 `stdair::BomRoot::BomRoot ( const BomRoot & iBomRoot )` `[protected]`

Copy constructor.

Definition at line 22 of file [BomRoot.cpp](#).

#### 32.24.3.3 `stdair::BomRoot::BomRoot ( const Key_T & iKey )` `[protected]`

Main constructor.

Definition at line 28 of file [BomRoot.cpp](#).

#### 32.24.3.4 `stdair::BomRoot::~~BomRoot ( )` `[protected]`

Destructor.

Definition at line 32 of file [BomRoot.cpp](#).

### 32.24.4 Member Function Documentation

#### 32.24.4.1 `const Key_T& stdair::BomRoot::getKey ( ) const` `[inline]`

Get the inventory key (airline code).

Definition at line 48 of file [BomRoot.hpp](#).

References [\\_key](#).

#### 32.24.4.2 `const HolderMap_T& stdair::BomRoot::getHolderMap ( ) const` `[inline]`

Get the map of children.

Definition at line 53 of file [BomRoot.hpp](#).

References [\\_holderMap](#).

#### 32.24.4.3 `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\(\)](#).

**32.24.4.4** `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\(\)](#).

**32.24.4.5** `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

<i>const</i>	<a href="#">std::string&amp;</a> 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\(\)](#).

**32.24.4.6** `Inventory * stdair::BomRoot::getInventory ( const InventoryKey & iInventoryKey ) const`

Get a pointer on the [Inventory](#) object corresponding to the given key.

#### Note

The [Inventory](#) object can be inherited from, if needed. In that case, a `dynamic_cast<>` may be needed.

#### Parameters

<i>const</i>	<a href="#">InventoryKey&amp;</a> 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\(\)](#).

**32.24.4.7** `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\(\)](#).

**32.24.4.8** `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\(\)](#).

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

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Implements [stdair::BomAbstract](#).

Definition at line 111 of file [BomRoot.hpp](#).

References [toString\(\)](#).

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

Read a Business Object from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Implements [stdair::BomAbstract](#).

Definition at line 120 of file [BomRoot.hpp](#).

**32.24.4.11** 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\(\)](#).

**32.24.4.12** 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\(\)](#).

**32.24.4.13** 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](#).

## 32.24.5 Friends And Related Function Documentation

**32.24.5.1** `template<typename BOM> friend class FacBom` `[friend]`

Definition at line 33 of file [BomRoot.hpp](#).

**32.24.5.2** `template<typename BOM> friend class FacCloneBom` `[friend]`

Definition at line 34 of file [BomRoot.hpp](#).

**32.24.5.3** `friend class FacBomManager` `[friend]`

Definition at line 35 of file [BomRoot.hpp](#).

**32.24.5.4** `friend class boost::serialization::access` `[friend]`

Definition at line 36 of file [BomRoot.hpp](#).

## 32.24.6 Member Data Documentation

**32.24.6.1** `Key_T stdair::BomRoot::_key` `[protected]`

Primary key.

Definition at line 191 of file [BomRoot.hpp](#).

Referenced by [describeKey\(\)](#), [getKey\(\)](#), [serialize\(\)](#), and [toString\(\)](#).

**32.24.6.2** `HolderMap_T stdair::BomRoot::_holderMap` `[protected]`

Map holding the children ([Inventory](#) objects).

Definition at line 196 of file [BomRoot.hpp](#).

Referenced by [getHolderMap\(\)](#).

**32.24.6.3** `FRAT5CurveHolderStruct stdair::BomRoot::_frat5CurveHolder` `[protected]`

Holder of FRAT5 curves.

Definition at line 201 of file [BomRoot.hpp](#).

Referenced by [addFRAT5Curve\(\)](#), and [getFRAT5Curve\(\)](#).

**32.24.6.4** `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:

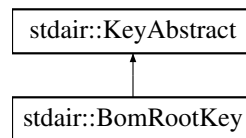
- [stdair/bom/BomRoot.hpp](#)
- [stdair/bom/BomRoot.cpp](#)
- [stdair/command/CmdBomSerialiser.cpp](#)

## 32.25 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 &identification)
- [BomRootKey](#) (const [BomRootKey](#) &)
- [~BomRootKey](#) ()
- const std::string & [getID](#) () const
- void [toStream](#) (std::ostream &ioOut) const
- void [fromStream](#) (std::istream &ioIn)
- const std::string [toString](#) () const
- template<class Archive >  
void [serialize](#) (Archive &ar, const unsigned int iFileVersion)

### Friends

- class [boost::serialization::access](#)

#### 32.25.1 Detailed Description

Key of the BOM structure root.

Definition at line 25 of file [BomRootKey.hpp](#).

#### 32.25.2 Constructor & Destructor Documentation

##### 32.25.2.1 stdair::BomRootKey::BomRootKey ( )

Default constructor.

Definition at line 18 of file [BomRootKey.cpp](#).

##### 32.25.2.2 stdair::BomRootKey::BomRootKey ( const std::string & *identification* )

Constructor.

Definition at line 28 of file [BomRootKey.cpp](#).

##### 32.25.2.3 stdair::BomRootKey::BomRootKey ( const [BomRootKey](#) & *iBomRootKey* )

Copy constructor.

Definition at line 23 of file [BomRootKey.cpp](#).

##### 32.25.2.4 stdair::BomRootKey::~~BomRootKey ( )

Destructor.

Definition at line 33 of file [BomRootKey.cpp](#).



### 32.25.3 Member Function Documentation

#### 32.25.3.1 `const std::string& stdair::BomRootKey::getID ( ) const` `[inline]`

Get the identification.

Definition at line 56 of file [BomRootKey.hpp](#).

#### 32.25.3.2 `void stdair::BomRootKey::toStream ( std::ostream & ioOut ) const` `[virtual]`

Dump a Business Object Key into an output stream.

##### Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 37 of file [BomRootKey.cpp](#).

References [toString\(\)](#).

#### 32.25.3.3 `void stdair::BomRootKey::fromStream ( std::istream & ioIn )` `[virtual]`

Read a Business Object Key from an input stream.

##### Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 42 of file [BomRootKey.cpp](#).

#### 32.25.3.4 `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\(\)](#).

#### 32.25.3.5 `template<class Archive > void stdair::BomRootKey::serialize ( Archive & ar, const unsigned int iFileVersion )`

Serialisation.

Definition at line 68 of file [BomRootKey.cpp](#).

### 32.25.4 Friends And Related Function Documentation

#### 32.25.4.1 `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](#)

## 32.26 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](#)

#### 32.26.1 Detailed Description

##### [BookingClass](#)

Definition at line 16 of file [StdairTestLib.hpp](#).

#### 32.26.2 Constructor & Destructor Documentation

32.26.2.1 stdair\_test::BookingClass::BookingClass ( const std::string & iClassCode ) [\[inline\]](#)

Constructor.

Definition at line 19 of file [StdairTestLib.hpp](#).

#### 32.26.3 Member Function Documentation

32.26.3.1 std::string stdair\_test::BookingClass::toString ( ) const [\[inline\]](#)

Display .

Definition at line 24 of file [StdairTestLib.hpp](#).

References [\\_classCode](#).

#### 32.26.4 Member Data Documentation

32.26.4.1 std::string stdair\_test::BookingClass::\_classCode

Definition at line 17 of file [StdairTestLib.hpp](#).

Referenced by [toString\(\)](#), and [stdair\\_test::Cabin::toString\(\)](#).

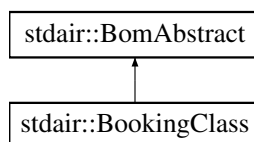
The documentation for this struct was generated from the following file:

- test/stdair/[StdairTestLib.hpp](#)

## 32.27 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 [setCumPriceDemMean](#) (const [MeanValue\\_T](#) &iMean)
- void [setCumPriceDemStdDev](#) (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

- `template<typename BOM >`  
class [FacBom](#)
- `template<typename BOM >`  
class [FacCloneBom](#)
- class [FacBomManager](#)

### 32.27.1 Detailed Description

Class representing the actual attributes for an airline booking class.

Definition at line 24 of file [BookingClass.hpp](#).

### 32.27.2 Member Typedef Documentation

#### 32.27.2.1 `typedef BookingClassKey stdair::BookingClass::Key_T`

Definition allowing to retrieve the associated BOM key type.

Definition at line 32 of file [BookingClass.hpp](#).

### 32.27.3 Constructor & Destructor Documentation

#### 32.27.3.1 `stdair::BookingClass::BookingClass ( const Key_T & iKey ) [protected]`

Constructor.

Definition at line 49 of file [BookingClass.cpp](#).

#### 32.27.3.2 `stdair::BookingClass::~~BookingClass ( ) [protected],[virtual]`

Destructor.

Definition at line 61 of file [BookingClass.cpp](#).

### 32.27.4 Member Function Documentation

#### 32.27.4.1 `const Key_T& stdair::BookingClass::getKey ( ) const [inline]`

Get the booking class key.

Definition at line 37 of file [BookingClass.hpp](#).

References [\\_key](#).

#### 32.27.4.2 `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\(\)](#).

#### 32.27.4.3 `BomAbstract* const stdair::BookingClass::getParent ( ) const [inline]`

Get the parent object.

Definition at line 47 of file [BookingClass.hpp](#).

References [\\_parent](#).

**32.27.4.4** `const HolderMap_T& stdair::BookingClass::getHolderMap ( ) const` `[inline]`

Get the map of children holders.

Definition at line 52 of file [BookingClass.hpp](#).

References [\\_holderMap](#).

**32.27.4.5** `const SubclassCode_T& stdair::BookingClass::getSubclassCode ( ) const` `[inline]`

Get teh sub-class code.

Definition at line 57 of file [BookingClass.hpp](#).

References [\\_subclassCode](#).

**32.27.4.6** `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](#).

**32.27.4.7** `const ProtectionLevel_T& stdair::BookingClass::getProtection ( ) const` `[inline]`

Get the protection.

Definition at line 67 of file [BookingClass.hpp](#).

References [\\_protection](#).

**32.27.4.8** `const ProtectionLevel_T& stdair::BookingClass::getCumulatedProtection ( ) const` `[inline]`

Get the cumulated protection.

Definition at line 72 of file [BookingClass.hpp](#).

References [\\_cumulatedProtection](#).

**32.27.4.9** `const BookingLimit_T& stdair::BookingClass::getCumulatedBookingLimit ( ) const` `[inline]`

Get the cumulated booking limit.

Definition at line 77 of file [BookingClass.hpp](#).

References [\\_cumulatedBookingLimit](#).

**32.27.4.10** `const NbOfSeats_T& stdair::BookingClass::getNegotiatedSpace ( ) const` `[inline]`

Get the negotiated space.

Definition at line 82 of file [BookingClass.hpp](#).

References [\\_nego](#).

**32.27.4.11** `const OverbookingRate_T& stdair::BookingClass::getNoShowPercentage ( ) const` `[inline]`

Get the no-show rate.

Definition at line 87 of file [BookingClass.hpp](#).

References [\\_noShowPercentage](#).

**32.27.4.12** `const OverbookingRate_T& stdair::BookingClass::getCancellationPercentage ( ) const` `[inline]`

Get the cancellation rate.

Definition at line 92 of file [BookingClass.hpp](#).

References [\\_cancellationPercentage](#).

**32.27.4.13** `const NbOfBookings_T& stdair::BookingClass::getNbOfBookings ( ) const` `[inline]`

Get the number of bookings.

Definition at line 97 of file [BookingClass.hpp](#).

References [\\_nbOfBookings](#).

**32.27.4.14** `const NbOfBookings_T& stdair::BookingClass::getNbOfGroupBookings ( ) const` `[inline]`

Get the number of group bookings.

Definition at line 102 of file [BookingClass.hpp](#).

References [\\_groupNbOfBookings](#).

**32.27.4.15** `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](#).

**32.27.4.16** `const NbOfBookings_T& stdair::BookingClass::getNbOfStaffBookings ( ) const` `[inline]`

Get the number of staff bookings.

Definition at line 112 of file [BookingClass.hpp](#).

References [\\_staffNbOfBookings](#).

**32.27.4.17** `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](#).

**32.27.4.18** `const NbOfCancellations_T& stdair::BookingClass::getNbOfCancellations ( ) const` `[inline]`

Get the number of cancellations.

Definition at line 122 of file [BookingClass.hpp](#).

References [\\_nbOfCancellations](#).

**32.27.4.19** `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](#).

**32.27.4.20** `const Availability_T& stdair::BookingClass::getNetClassAvailability ( ) const` `[inline]`

Get the net segment class availability.

Definition at line 132 of file [BookingClass.hpp](#).

References [\\_netClassAvailability](#).

**32.27.4.21** `const Availability_T& stdair::BookingClass::getSegmentAvailability ( ) const` `[inline]`

Get the segment class availability.

Definition at line 137 of file [BookingClass.hpp](#).

References [\\_segmentAvailability](#).

**32.27.4.22** `const Availability_T& stdair::BookingClass::getNetRevenueAvailability ( ) const` `[inline]`

Net revenue availability.

Definition at line 142 of file [BookingClass.hpp](#).

References [\\_netRevenueAvailability](#).

**32.27.4.23** `const Yield_T& stdair::BookingClass::getYield ( ) const` `[inline]`

Yield.

Definition at line 147 of file [BookingClass.hpp](#).

References [\\_yield](#).

**32.27.4.24** `const Yield_T& stdair::BookingClass::getAdjustedYield ( ) const` `[inline]`

Definition at line 148 of file [BookingClass.hpp](#).

References [\\_adjustedYield](#).

**32.27.4.25** `const MeanValue_T& stdair::BookingClass::getMean ( ) const` `[inline]`

Demand distribution.

Definition at line 151 of file [BookingClass.hpp](#).

References [\\_mean](#).

**32.27.4.26** `const StdDevValue_T& stdair::BookingClass::getStdDev ( ) const` `[inline]`

Definition at line 152 of file [BookingClass.hpp](#).

References [\\_stdDev](#).

**32.27.4.27** `const MeanValue_T& stdair::BookingClass::getPriceDemMean ( ) const` `[inline]`

Definition at line 153 of file [BookingClass.hpp](#).

References [\\_priceDemMean](#).

**32.27.4.28** `const StdDevValue_T& stdair::BookingClass::getPriceDemStdDev ( ) const` `[inline]`

Definition at line 154 of file [BookingClass.hpp](#).

References [\\_priceDemStdDev](#).

**32.27.4.29** `const MeanValue_T& stdair::BookingClass::getCumuPriceDemMean ( ) const` `[inline]`

Definition at line 155 of file [BookingClass.hpp](#).

References [\\_cumuPriceDemMean](#).

**32.27.4.30** `const StdDevValue_T& stdair::BookingClass::getCumuPriceDemStdDev ( ) const` `[inline]`

Definition at line 158 of file [BookingClass.hpp](#).

References [\\_cumuPriceDemStdDev](#).



32.27.4.31 `const MeanValue_T& stdair::BookingClass::getProductDemMean ( ) const` [inline]

Definition at line 161 of file [BookingClass.hpp](#).

References [\\_productDemMean](#).

32.27.4.32 `const StdDevValue_T& stdair::BookingClass::getProductDemStdDev ( ) const` [inline]

Definition at line 162 of file [BookingClass.hpp](#).

References [\\_productDemStdDev](#).

32.27.4.33 `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\(\)](#).

32.27.4.34 `void stdair::BookingClass::setCumulatedProtection ( const ProtectionLevel_T & iPL )` [inline]

Cumulated protection.

Definition at line 172 of file [BookingClass.hpp](#).

References [\\_cumulatedProtection](#).

32.27.4.35 `void stdair::BookingClass::setProtection ( const ProtectionLevel_T & iPL )` [inline]

Protection.

Definition at line 177 of file [BookingClass.hpp](#).

References [\\_protection](#).

32.27.4.36 `void stdair::BookingClass::setCumulatedBookingLimit ( const BookingLimit_T & iBL )` [inline]

Cumulated booking limit.

Definition at line 182 of file [BookingClass.hpp](#).

References [\\_cumulatedBookingLimit](#).

32.27.4.37 `void stdair::BookingClass::setAuthorizationLevel ( const AuthorizationLevel_T & iAU )` [inline]

Authorization level.

Definition at line 187 of file [BookingClass.hpp](#).

References [\\_au](#).

32.27.4.38 `void stdair::BookingClass::setSegmentAvailability ( const Availability_T & iAvl )` [inline]

Set availability.

Definition at line 192 of file [BookingClass.hpp](#).

References [\\_segmentAvailability](#).

32.27.4.39 `void stdair::BookingClass::setYield ( const Yield_T & iYield )` [inline]

Yield.

Definition at line 197 of file [BookingClass.hpp](#).

References [\\_adjustedYield](#), and [\\_yield](#).

32.27.4.40 void stdair::BookingClass::setAdjustedYield ( const Yield\_T & iYield ) [inline]

Definition at line 201 of file [BookingClass.hpp](#).

References [\\_adjustedYield](#).

32.27.4.41 void stdair::BookingClass::setMean ( const MeanValue\_T & iMean ) [inline]

Demand distribution.

Definition at line 204 of file [BookingClass.hpp](#).

References [\\_mean](#).

32.27.4.42 void stdair::BookingClass::setStdDev ( const StdDevValue\_T & iStdDev ) [inline]

Definition at line 205 of file [BookingClass.hpp](#).

References [\\_stdDev](#).

32.27.4.43 void stdair::BookingClass::setPriceDemMean ( const MeanValue\_T & iMean ) [inline]

Definition at line 206 of file [BookingClass.hpp](#).

References [\\_priceDemMean](#).

32.27.4.44 void stdair::BookingClass::setPriceDemStdDev ( const StdDevValue\_T & iStdDev ) [inline]

Definition at line 207 of file [BookingClass.hpp](#).

References [\\_priceDemStdDev](#).

32.27.4.45 void stdair::BookingClass::setCumuPriceDemMean ( const MeanValue\_T & iMean ) [inline]

Definition at line 210 of file [BookingClass.hpp](#).

References [\\_cumuPriceDemMean](#).

32.27.4.46 void stdair::BookingClass::setCumuPriceDemStdDev ( const StdDevValue\_T & iStdDev ) [inline]

Definition at line 212 of file [BookingClass.hpp](#).

References [\\_cumuPriceDemStdDev](#).

32.27.4.47 void stdair::BookingClass::setProductDemMean ( const MeanValue\_T & iMean ) [inline]

Definition at line 215 of file [BookingClass.hpp](#).

References [\\_productDemMean](#).

32.27.4.48 void stdair::BookingClass::setProductDemStdDev ( const StdDevValue\_T & iStdDev ) [inline]

Definition at line 218 of file [BookingClass.hpp](#).

References [\\_productDemStdDev](#).

32.27.4.49 void stdair::BookingClass::toStream ( std::ostream & ioOut ) const [inline],[virtual]

Dump a Business Object into an output stream.

## Parameters

<i>ostream</i> &	the output stream.
------------------	--------------------

Implements [stdair::BomAbstract](#).

Definition at line 226 of file [BookingClass.hpp](#).

References [toString\(\)](#).

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

Read a Business Object from an input stream.

## Parameters

<i>istream</i> &	the input stream.
------------------	-------------------

Implements [stdair::BomAbstract](#).

Definition at line 232 of file [BookingClass.hpp](#).

32.27.4.51 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\(\)](#).

32.27.4.52 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\(\)](#).

32.27.4.53 void stdair::BookingClass::sell ( const NbOfBookings\_T & *iNbOfBookings* )

Register a sale.

Definition at line 72 of file [BookingClass.cpp](#).

References [\\_nbOfBookings](#).

32.27.4.54 void stdair::BookingClass::cancel ( const NbOfBookings\_T & *iNbOfCancellations* )

Register a cancellation.

Definition at line 77 of file [BookingClass.cpp](#).

References [\\_nbOfBookings](#), and [\\_nbOfCancellations](#).

32.27.4.55 void stdair::BookingClass::generateDemandSamples ( const NbOfSamples\_T & *K* )

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\(\)](#).

32.27.4.56 void stdair::BookingClass::generateDemandSamples ( const NbOfSamples\_T & K, const RandomSeed\_T & iSeed )

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\(\)](#).

### 32.27.5 Friends And Related Function Documentation

32.27.5.1 template<typename BOM > friend class FacBom [friend]

Definition at line 25 of file [BookingClass.hpp](#).

32.27.5.2 template<typename BOM > friend class FacCloneBom [friend]

Definition at line 26 of file [BookingClass.hpp](#).

32.27.5.3 friend class FacBomManager [friend]

Definition at line 27 of file [BookingClass.hpp](#).

### 32.27.6 Member Data Documentation

32.27.6.1 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\(\)](#).

32.27.6.2 BomAbstract\* stdair::BookingClass::\_parent [protected]

Pointer on the parent class ([SegmentCabin](#)).

Definition at line 279 of file [BookingClass.hpp](#).

Referenced by [getParent\(\)](#).

32.27.6.3 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\(\)](#).

32.27.6.4 SubclassCode\_T stdair::BookingClass::\_subclassCode [protected]

Sub-class code.

Definition at line 285 of file [BookingClass.hpp](#).

Referenced by [getSubclassCode\(\)](#).

32.27.6.5 ProtectionLevel\_T stdair::BookingClass::\_cumulatedProtection [protected]

Cumulated protection.

Definition at line 288 of file [BookingClass.hpp](#).

Referenced by [getCumulatedProtection\(\)](#), and [setCumulatedProtection\(\)](#).

**32.27.6.6 ProtectionLevel\_T** stdair::BookingClass::\_protection [protected]

Protection.

Definition at line 291 of file [BookingClass.hpp](#).

Referenced by [getProtection\(\)](#), and [setProtection\(\)](#).

**32.27.6.7 BookingLimit\_T** stdair::BookingClass::\_cumulatedBookingLimit [protected]

Cumulated booking limit.

Definition at line 294 of file [BookingClass.hpp](#).

Referenced by [getCumulatedBookingLimit\(\)](#), and [setCumulatedBookingLimit\(\)](#).

**32.27.6.8 AuthorizationLevel\_T** stdair::BookingClass::\_au [protected]

Authorization level.

Definition at line 297 of file [BookingClass.hpp](#).

Referenced by [getAuthorizationLevel\(\)](#), and [setAuthorizationLevel\(\)](#).

**32.27.6.9 NbOfSeats\_T** stdair::BookingClass::\_nego [protected]

Negotiated space.

Definition at line 300 of file [BookingClass.hpp](#).

Referenced by [getNegotiatedSpace\(\)](#).

**32.27.6.10 OverbookingRate\_T** stdair::BookingClass::\_noShowPercentage [protected]

Overbooking rate.

Definition at line 303 of file [BookingClass.hpp](#).

Referenced by [getNoShowPercentage\(\)](#).

**32.27.6.11 OverbookingRate\_T** stdair::BookingClass::\_cancellationPercentage [protected]

Cancellation rate.

Definition at line 306 of file [BookingClass.hpp](#).

Referenced by [getCancellationPercentage\(\)](#).

**32.27.6.12 NbOfBookings\_T** stdair::BookingClass::\_nbOfBookings [protected]

Number of bookings.

Definition at line 309 of file [BookingClass.hpp](#).

Referenced by [cancel\(\)](#), [getNbOfBookings\(\)](#), and [sell\(\)](#).

**32.27.6.13 NbOfBookings\_T** stdair::BookingClass::\_groupNbOfBookings [protected]

Number of group bookings.

Definition at line 312 of file [BookingClass.hpp](#).

Referenced by [getNbOfGroupBookings\(\)](#).

**32.27.6.14 NbOfBookings\_T** stdair::BookingClass::\_groupPendingNbOfBookings [protected]

Number of pending group bookings.

Definition at line 315 of file [BookingClass.hpp](#).

Referenced by [getNbOfPendingGroupBookings\(\)](#).

**32.27.6.15 NbOfBookings\_T** stdair::BookingClass::\_staffNbOfBookings [protected]

Number of staff bookings.

Definition at line 318 of file [BookingClass.hpp](#).

Referenced by [getNbOfStaffBookings\(\)](#).

**32.27.6.16 NbOfBookings\_T** stdair::BookingClass::\_wlNbOfBookings [protected]

Number of wait-list bookings.

Definition at line 321 of file [BookingClass.hpp](#).

Referenced by [getNbOfWLBookings\(\)](#).

**32.27.6.17 NbOfCancellations\_T** stdair::BookingClass::\_nbOfCancellations [protected]

Number of cancellations.

Definition at line 324 of file [BookingClass.hpp](#).

Referenced by [cancel\(\)](#), and [getNbOfCancellations\(\)](#).

**32.27.6.18 NbOfBookings\_T** stdair::BookingClass::\_etb [protected]

Expected to board (ETB).

Definition at line 327 of file [BookingClass.hpp](#).

Referenced by [getETB\(\)](#).

**32.27.6.19 Availability\_T** stdair::BookingClass::\_netClassAvailability [protected]

Net segment class availability.

Definition at line 330 of file [BookingClass.hpp](#).

Referenced by [getNetClassAvailability\(\)](#).

**32.27.6.20 Availability\_T** stdair::BookingClass::\_segmentAvailability [protected]

Segment class availability.

Definition at line 333 of file [BookingClass.hpp](#).

Referenced by [getSegmentAvailability\(\)](#), and [setSegmentAvailability\(\)](#).

**32.27.6.21 Availability\_T** stdair::BookingClass::\_netRevenueAvailability [protected]

Net revenue availability.

Definition at line 336 of file [BookingClass.hpp](#).

Referenced by [getNetRevenueAvailability\(\)](#).

**32.27.6.22 Yield\_T** stdair::BookingClass::\_yield [protected]

Yield.

Definition at line 339 of file [BookingClass.hpp](#).

Referenced by [getYield\(\)](#), and [setYield\(\)](#).

**32.27.6.23 Yield\_T** stdair::BookingClass::\_adjustedYield [protected]

Definition at line 340 of file [BookingClass.hpp](#).

Referenced by [getAdjustedYield\(\)](#), [setAdjustedYield\(\)](#), and [setYield\(\)](#).

**32.27.6.24 MeanValue\_T** `stdair::BookingClass::_mean` [protected]

Demand distribution forecast.

Definition at line 343 of file [BookingClass.hpp](#).

Referenced by [generateDemandSamples\(\)](#), [getMean\(\)](#), and [setMean\(\)](#).

**32.27.6.25 StdDevValue\_T** `stdair::BookingClass::_stdDev` [protected]

Definition at line 344 of file [BookingClass.hpp](#).

Referenced by [generateDemandSamples\(\)](#), [getStdDev\(\)](#), and [setStdDev\(\)](#).

**32.27.6.26 MeanValue\_T** `stdair::BookingClass::_priceDemMean` [protected]

Price-oriented demand distribution forecast.

Definition at line 347 of file [BookingClass.hpp](#).

Referenced by [getPriceDemMean\(\)](#), and [setPriceDemMean\(\)](#).

**32.27.6.27 StdDevValue\_T** `stdair::BookingClass::_priceDemStdDev` [protected]

Definition at line 348 of file [BookingClass.hpp](#).

Referenced by [getPriceDemStdDev\(\)](#), and [setPriceDemStdDev\(\)](#).

**32.27.6.28 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\(\)](#).

**32.27.6.29 StdDevValue\_T** `stdair::BookingClass::_cumuPriceDemStdDev` [protected]

Definition at line 352 of file [BookingClass.hpp](#).

Referenced by [getCumuPriceDemStdDev\(\)](#), and [setCumuPriceDemStdDev\(\)](#).

**32.27.6.30 MeanValue\_T** `stdair::BookingClass::_productDemMean` [protected]

Product-oriented demand distribution forecast.

Definition at line 355 of file [BookingClass.hpp](#).

Referenced by [getProductDemMean\(\)](#), and [setProductDemMean\(\)](#).

**32.27.6.31 StdDevValue\_T** `stdair::BookingClass::_productDemStdDev` [protected]

Definition at line 356 of file [BookingClass.hpp](#).

Referenced by [getProductDemStdDev\(\)](#), and [setProductDemStdDev\(\)](#).

**32.27.6.32 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:

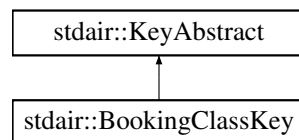
- `stdair/bom/BookingClass.hpp`

- [stdair/bom/BookingClass.cpp](#)

## 32.28 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

### 32.28.1 Detailed Description

Key of a given leg-cabin, made of a cabin code.

Definition at line 16 of file [BookingClassKey.hpp](#).

### 32.28.2 Constructor & Destructor Documentation

#### 32.28.2.1 stdair::BookingClassKey::BookingClassKey ( const [ClassCode\\_T](#) &iClassCode )

Constructor.

Definition at line 24 of file [BookingClassKey.cpp](#).

#### 32.28.2.2 stdair::BookingClassKey::BookingClassKey ( const [BookingClassKey](#) &iKey )

Default copy constructor.

Definition at line 19 of file [BookingClassKey.cpp](#).

#### 32.28.2.3 stdair::BookingClassKey::~~BookingClassKey ( )

Destructor.

Definition at line 29 of file [BookingClassKey.cpp](#).

### 32.28.3 Member Function Documentation

#### 32.28.3.1 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\(\)](#).

**32.28.3.2** `void stdair::BookingClassKey::toStream ( std::ostream & ioOut ) const [virtual]`

Dump a Business Object Key into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 33 of file [BookingClassKey.cpp](#).

References [toString\(\)](#).

**32.28.3.3** `void stdair::BookingClassKey::fromStream ( std::istream & ioIn ) [virtual]`

Read a Business Object Key from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 38 of file [BookingClassKey.cpp](#).

**32.28.3.4** `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\(\)](#).

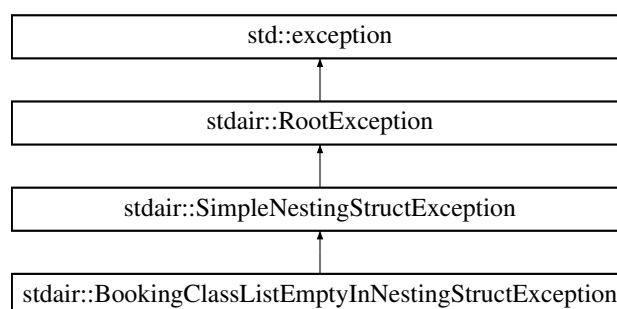
The documentation for this struct was generated from the following files:

- [stdair/bom/BookingClassKey.hpp](#)
- [stdair/bom/BookingClassKey.cpp](#)

## 32.29 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](#)

## 32.29.1 Detailed Description

Empty booking class list in Simple Nesting Structure.

Definition at line 219 of file [stdair\\_exceptions.hpp](#).

## 32.29.2 Constructor &amp; Destructor Documentation

32.29.2.1 `stdair::BookingClassListEmptyInNestingStructException::BookingClassListEmptyInNestingStructException ( const std::string & iWhat ) [inline]`

Constructor.

Definition at line 223 of file [stdair\\_exceptions.hpp](#).

## 32.29.3 Member Function Documentation

32.29.3.1 `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\(\)](#).

## 32.29.4 Member Data Documentation

32.29.4.1 `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:

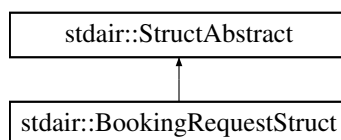
- [stdair/stdair\\_exceptions.hpp](#)

## 32.30 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](#) & [getPreferredDepartureDate](#) () const
- const [Duration\\_T](#) & [getPreferredDepartureTime](#) () const
- const [DateTime\\_T](#) & [getRequestDateTime](#) () const
- const [CabinCode\\_T](#) & [getPreferredCabin](#) () const
- const [NbOfSeats\\_T](#) & [getPartySize](#) () const
- const [ChannelLabel\\_T](#) & [getBookingChannel](#) () const
- const [TripType\\_T](#) & [getTripType](#) () const
- const [DayDuration\\_T](#) & [getStayDuration](#) () const
- const [FrequentFlyer\\_T](#) & [getFrequentFlyerType](#) () const
- const [WTP\\_T](#) & [getWTP](#) () const
- const [PriceValue\\_T](#) & [getValueOfTime](#) () const
- 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](#) ()

### 32.30.1 Detailed Description

Structure holding the elements of a booking request.

Definition at line 21 of file [BookingRequestStruct.hpp](#).

## 32.30.2 Constructor &amp; Destructor Documentation

32.30.2.1 `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](#).

32.30.2.2 `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](#).

32.30.2.3 `stdair::BookingRequestStruct::BookingRequestStruct ( const BookingRequestStruct & iBookingRequest )`

Copy constructor.

Definition at line 39 of file [BookingRequestStruct.cpp](#).

32.30.2.4 `stdair::BookingRequestStruct::~~BookingRequestStruct ( )`

Destructor.

Definition at line 131 of file [BookingRequestStruct.cpp](#).

## 32.30.3 Member Function Documentation

32.30.3.1 `const DemandGeneratorKey_T& stdair::BookingRequestStruct::getDemandGeneratorKey ( ) const` [inline]

Get the demand generator key.

Definition at line 25 of file [BookingRequestStruct.hpp](#).

32.30.3.2 `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\(\)](#).

32.30.3.3 `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\(\)](#).

**32.30.3.4** `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\(\)](#).

**32.30.3.5** `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\(\)](#).

**32.30.3.6** `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\(\)](#).

**32.30.3.7** `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\(\)](#).

**32.30.3.8** `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\(\)](#).

**32.30.3.9** `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\(\)](#).

**32.30.3.10** `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\(\)](#).

**32.30.3.11** `const TripType_T& stdair::BookingRequestStruct::getTripType ( ) const` `[inline]`

Get the trip type.

Definition at line 75 of file [BookingRequestStruct.hpp](#).

**32.30.3.12** `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\(\)](#).

32.30.3.13 `const FrequentFlyer_T& stdair::BookingRequestStruct::getFrequentFlyerType ( ) const` `[inline]`

Get the frequent flyer type.

Definition at line 85 of file [BookingRequestStruct.hpp](#).

32.30.3.14 `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\(\)](#).

32.30.3.15 `const PriceValue_T& stdair::BookingRequestStruct::getValueOfTime ( ) const` `[inline]`

Get the value of time.

Definition at line 95 of file [BookingRequestStruct.hpp](#).

32.30.3.16 `const ChangeFees_T& stdair::BookingRequestStruct::getChangeFees ( ) const` `[inline]`

Get the change fee acceptance.

Definition at line 100 of file [BookingRequestStruct.hpp](#).

32.30.3.17 `const Disutility_T& stdair::BookingRequestStruct::getChangeFeeDisutility ( ) const` `[inline]`

Get the change disutility.

Definition at line 105 of file [BookingRequestStruct.hpp](#).

32.30.3.18 `const NonRefundable_T& stdair::BookingRequestStruct::getNonRefundable ( ) const` `[inline]`

Get the non refundable acceptance.

Definition at line 110 of file [BookingRequestStruct.hpp](#).

32.30.3.19 `const Disutility_T& stdair::BookingRequestStruct::getNonRefundableDisutility ( ) const` `[inline]`

Get the non refundable disutility.

Definition at line 115 of file [BookingRequestStruct.hpp](#).

32.30.3.20 `void stdair::BookingRequestStruct::toStream ( std::ostream & ioOut ) const`

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Definition at line 135 of file [BookingRequestStruct.cpp](#).

References [describe\(\)](#).

32.30.3.21 `void stdair::BookingRequestStruct::fromStream ( std::istream & ioIn )` `[virtual]`

Read a Business Object from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented from [stdair::StructAbstract](#).

Definition at line 140 of file [BookingRequestStruct.cpp](#).

**32.30.3.22** `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\(\)](#).

**32.30.3.23** `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](#).

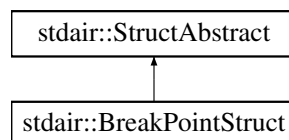
The documentation for this struct was generated from the following files:

- [stdair/bom/BookingRequestStruct.hpp](#)
- [stdair/bom/BookingRequestStruct.cpp](#)

### 32.31 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](#) ()

#### 32.31.1 Detailed Description

Structure holding the elements of a break point.

Definition at line 18 of file [BreakPointStruct.hpp](#).



### 32.31.2 Constructor & Destructor Documentation

#### 32.31.2.1 `stdair::BreakPointStruct::BreakPointStruct ( const DateTime_T & iBreakPointTime )`

Constructor.

Definition at line 26 of file [BreakPointStruct.cpp](#).

#### 32.31.2.2 `stdair::BreakPointStruct::BreakPointStruct ( const Date_T & iBreakPointDate )`

Constructor.

Definition at line 32 of file [BreakPointStruct.cpp](#).

#### 32.31.2.3 `stdair::BreakPointStruct::BreakPointStruct ( const BreakPointStruct & iBreakPoint )`

Copy constructor.

Definition at line 20 of file [BreakPointStruct.cpp](#).

#### 32.31.2.4 `stdair::BreakPointStruct::~~BreakPointStruct ( )`

Destructor.

Definition at line 37 of file [BreakPointStruct.cpp](#).

### 32.31.3 Member Function Documentation

#### 32.31.3.1 `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\(\)](#).

#### 32.31.3.2 `void stdair::BreakPointStruct::toStream ( std::ostream & ioOut ) const`

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Definition at line 41 of file [BreakPointStruct.cpp](#).

References [describe\(\)](#).

#### 32.31.3.3 `void stdair::BreakPointStruct::fromStream ( std::istream & ioIn ) [virtual]`

Read a Business Object from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented from [stdair::StructAbstract](#).

Definition at line 46 of file [BreakPointStruct.cpp](#).

#### 32.31.3.4 `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:

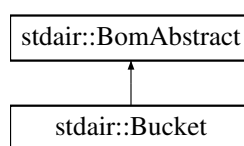
- [stdair/bom/BreakPointStruct.hpp](#)
- [stdair/bom/BreakPointStruct.cpp](#)

## 32.32 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

- `template<typename BOM >`  
class [FacBom](#)
- `template<typename BOM >`  
class [FacCloneBom](#)
- class [FacBomManager](#)
- class [boost::serialization::access](#)

#### 32.32.1 Detailed Description

Class representing the actual attributes for an airline booking class.

Definition at line 29 of file [Bucket.hpp](#).

#### 32.32.2 Member Typedef Documentation

##### 32.32.2.1 `typedef BucketKey stdair::Bucket::Key_T`

Definition allowing to retrieve the associated BOM key type.

Definition at line 40 of file [Bucket.hpp](#).

#### 32.32.3 Constructor & Destructor Documentation

##### 32.32.3.1 `stdair::Bucket::Bucket ( const Key_T & iKey )` [protected]

Default constructor.

Definition at line 34 of file [Bucket.cpp](#).

##### 32.32.3.2 `stdair::Bucket::~~Bucket ( )` [protected], [virtual]

Destructor.

Definition at line 38 of file [Bucket.cpp](#).

#### 32.32.4 Member Function Documentation

##### 32.32.4.1 `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](#).

**32.32.4.2 BomAbstract\*** `const stdair::Bucket::getParent ( ) const` `[inline]`

Get the parent object.

Definition at line 54 of file [Bucket.hpp](#).

References [\\_parent](#).

**32.32.4.3 const HolderMap\_T&** `stdair::Bucket::getHolderMap ( ) const` `[inline]`

Get the map of children holders.

Definition at line 59 of file [Bucket.hpp](#).

References [\\_holderMap](#).

**32.32.4.4 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\(\)](#).

**32.32.4.5 const Yield\_T&** `stdair::Bucket::getYieldRangeUpperValue ( ) const` `[inline]`

Get the upper yield range.

Definition at line 69 of file [Bucket.hpp](#).

References [\\_yieldRangeUpperValue](#).

**32.32.4.6 const CabinCapacity\_T&** `stdair::Bucket::getAvailability ( ) const` `[inline]`

Get the availability.

Definition at line 74 of file [Bucket.hpp](#).

References [\\_availability](#).

**32.32.4.7 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](#).

**32.32.4.8 void** `stdair::Bucket::setYieldRangeUpperValue ( const Yield_T & iYield )` `[inline]`

Set the upper yield range.

Definition at line 86 of file [Bucket.hpp](#).

References [\\_yieldRangeUpperValue](#).

**32.32.4.9 void** `stdair::Bucket::setAvailability ( const CabinCapacity_T & iAvl )` `[inline]`

Set the availability.

Definition at line 91 of file [Bucket.hpp](#).

References [\\_availability](#).

**32.32.4.10 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](#).

**32.32.4.11** `void stdair::Bucket::toStream ( std::ostream & ioOut ) const [inline],[virtual]`

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Implements [stdair::BomAbstract](#).

Definition at line 108 of file [Bucket.hpp](#).

References [toString\(\)](#).

**32.32.4.12** `void stdair::Bucket::fromStream ( std::istream & ioIn ) [inline],[virtual]`

Read a Business Object from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Implements [stdair::BomAbstract](#).

Definition at line 117 of file [Bucket.hpp](#).

**32.32.4.13** `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\(\)](#).

**32.32.4.14** `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\(\)](#).

**32.32.4.15** `template<class Archive > void stdair::Bucket::serialize ( Archive & ar, const unsigned int iFileVersion )`

Serialisation.

Definition at line 64 of file [Bucket.cpp](#).

References [\\_key](#).

## 32.32.5 Friends And Related Function Documentation

**32.32.5.1** `template<typename BOM > friend class FacBom [friend]`

Definition at line 30 of file [Bucket.hpp](#).

**32.32.5.2** `template<typename BOM > friend class FacCloneBom [friend]`

Definition at line 31 of file [Bucket.hpp](#).

### 32.32.5.3 friend class FacBomManager [friend]

Definition at line 32 of file [Bucket.hpp](#).

### 32.32.5.4 friend class boost::serialization::access [friend]

Definition at line 33 of file [Bucket.hpp](#).

## 32.32.6 Member Data Documentation

### 32.32.6.1 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\(\)](#).

### 32.32.6.2 BomAbstract\* stdair::Bucket::\_parent [protected]

Pointer on the parent class ([LegCabin](#)).

Definition at line 184 of file [Bucket.hpp](#).

Referenced by [getParent\(\)](#).

### 32.32.6.3 HolderMap\_T stdair::Bucket::\_holderMap [protected]

Map holding the children (empty for now).

Definition at line 189 of file [Bucket.hpp](#).

Referenced by [getHolderMap\(\)](#).

### 32.32.6.4 Yield\_T stdair::Bucket::\_yieldRangeUpperValue [protected]

Upper yield range.

Definition at line 197 of file [Bucket.hpp](#).

Referenced by [getYieldRangeUpperValue\(\)](#), and [setYieldRangeUpperValue\(\)](#).

### 32.32.6.5 CabinCapacity\_T stdair::Bucket::\_availability [protected]

Availability.

Definition at line 202 of file [Bucket.hpp](#).

Referenced by [getAvailability\(\)](#), and [setAvailability\(\)](#).

### 32.32.6.6 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:

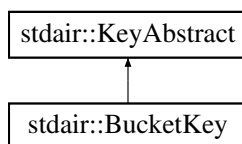
- [stdair/bom/Bucket.hpp](#)
- [stdair/bom/Bucket.cpp](#)

### 32.33 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](#)

#### 32.33.1 Detailed Description

Key of booking-class.

Definition at line 26 of file [BucketKey.hpp](#).

#### 32.33.2 Constructor & Destructor Documentation

##### 32.33.2.1 stdair::BucketKey::BucketKey ( const [SeatIndex\\_T](#) & *iSeatIndex* )

Main constructor.

Definition at line 22 of file [BucketKey.cpp](#).

##### 32.33.2.2 stdair::BucketKey::BucketKey ( const [BucketKey](#) & *iBucketKey* )

Copy constructor.

Definition at line 27 of file [BucketKey.cpp](#).

##### 32.33.2.3 stdair::BucketKey::~~BucketKey ( )

Destructor.

Definition at line 32 of file [BucketKey.cpp](#).

## 32.33.3 Member Function Documentation

## 32.33.3.1 const SeatIndex\_T&amp; stdair::BucketKey::getSeatIndex ( ) const [inline]

Get the seat index.

Definition at line 54 of file [BucketKey.hpp](#).

Referenced by [stdair::Bucket::getSeatIndex\(\)](#).

## 32.33.3.2 void stdair::BucketKey::toStream ( std::ostream &amp; ioOut ) const [virtual]

Dump a Business Object Key into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 36 of file [BucketKey.cpp](#).

References [toString\(\)](#).

## 32.33.3.3 void stdair::BucketKey::fromStream ( std::istream &amp; iIn ) [virtual]

Read a Business Object Key from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 41 of file [BucketKey.cpp](#).

## 32.33.3.4 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\(\)](#).

## 32.33.3.5 template&lt;class Archive &gt; void stdair::BucketKey::serialize ( Archive &amp; ar, const unsigned int iFileVersion )

Serialisation.

Definition at line 67 of file [BucketKey.cpp](#).

## 32.33.4 Friends And Related Function Documentation

## 32.33.4.1 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](#)



### 32.34 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

#### 32.34.1 Detailed Description

##### [Cabin](#)

Definition at line 32 of file [StdairTestLib.hpp](#).

#### 32.34.2 Member Typedef Documentation

##### 32.34.2.1 typedef [BookingClass](#) stdair\_test::Cabin::child

Child type.

Definition at line 46 of file [StdairTestLib.hpp](#).

#### 32.34.3 Constructor & Destructor Documentation

##### 32.34.3.1 stdair\_test::Cabin::Cabin ( const [BookingClass](#) & iBkgClass ) [inline]

Definition at line 34 of file [StdairTestLib.hpp](#).

#### 32.34.4 Member Function Documentation

##### 32.34.4.1 std::string stdair\_test::Cabin::toString ( ) const [inline]

Display .

Definition at line 39 of file [StdairTestLib.hpp](#).

References [stdair\\_test::BookingClass::\\_classCode](#).

#### 32.34.5 Member Data Documentation

##### 32.34.5.1 [BookingClass](#) stdair\_test::Cabin::\_bookingClass

Definition at line 33 of file [StdairTestLib.hpp](#).

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

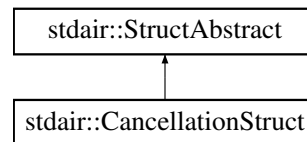
- test/stdair/[StdairTestLib.hpp](#)

### 32.35 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](#) ()

#### 32.35.1 Detailed Description

Structure holding the elements of a travel solution.

Definition at line 23 of file [CancellationStruct.hpp](#).

#### 32.35.2 Constructor & Destructor Documentation

**32.35.2.1** stdair::CancellationStruct::CancellationStruct ( const [SegmentPath\\_T](#) & *iSegPath*, const [ClassList\\_String\\_T](#) & *iList*, const [PartySize\\_T](#) & *iSize*, const [DateTime\\_T](#) & *iDateTime* )

Default constructor without class ID list.

Definition at line 14 of file [CancellationStruct.cpp](#).

**32.35.2.2** stdair::CancellationStruct::CancellationStruct ( const [SegmentPath\\_T](#) & *iSegPath*, const [BookingClassIDList\\_T](#) & *iIDList*, const [PartySize\\_T](#) & *iSize*, const [DateTime\\_T](#) & *iDateTime* )

Default constructor with class ID list.

Definition at line 23 of file [CancellationStruct.cpp](#).

**32.35.2.3** stdair::CancellationStruct::~~CancellationStruct ( )

Destructor.

Definition at line 32 of file [CancellationStruct.cpp](#).

### 32.35.3 Member Function Documentation

#### 32.35.3.1 `const SegmentPath_T& stdair::CancellationStruct::getSegmentPath ( ) const` `[inline]`

Get the segment path.

Definition at line 27 of file [CancellationStruct.hpp](#).

#### 32.35.3.2 `const ClassList_String_T& stdair::CancellationStruct::getClassList ( ) const` `[inline]`

Get the class list.

Definition at line 32 of file [CancellationStruct.hpp](#).

#### 32.35.3.3 `const BookingClassIDList_T& stdair::CancellationStruct::getClassIDList ( ) const` `[inline]`

Get the class ID list.

Definition at line 37 of file [CancellationStruct.hpp](#).

#### 32.35.3.4 `const PartySize_T& stdair::CancellationStruct::getPartySize ( ) const` `[inline]`

Get the party size.

Definition at line 42 of file [CancellationStruct.hpp](#).

#### 32.35.3.5 `const DateTime_T& stdair::CancellationStruct::getCancellationDateTime ( ) const` `[inline]`

Get the datetime.

Definition at line 47 of file [CancellationStruct.hpp](#).

#### 32.35.3.6 `void stdair::CancellationStruct::toStream ( std::ostream & ioOut ) const`

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Definition at line 36 of file [CancellationStruct.cpp](#).

References [describe\(\)](#).

#### 32.35.3.7 `void stdair::CancellationStruct::fromStream ( std::istream & ioIn )` `[virtual]`

Read a Business Object from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented from [stdair::StructAbstract](#).

Definition at line 41 of file [CancellationStruct.cpp](#).

#### 32.35.3.8 `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\(\)](#).

## 32.35.3.9 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\(\)](#).

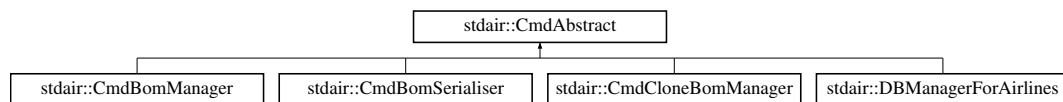
The documentation for this struct was generated from the following files:

- [stdair/bom/CancellationStruct.hpp](#)
- [stdair/bom/CancellationStruct.cpp](#)

## 32.36 stdair::CmdAbstract Class Reference

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

Inheritance diagram for stdair::CmdAbstract:



## 32.36.1 Detailed Description

Base class for the Command layer.

Definition at line 11 of file [CmdAbstract.hpp](#).

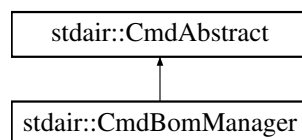
The documentation for this class was generated from the following file:

- [stdair/command/CmdAbstract.hpp](#)

## 32.37 stdair::CmdBomManager Class Reference

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

Inheritance diagram for stdair::CmdBomManager:



## Friends

- class [STDAIR\\_Service](#)

## 32.37.1 Detailed Description

Class wrapping utility functions for handling the BOM tree objects.

Definition at line 25 of file [CmdBomManager.hpp](#).

### 32.37.2 Friends And Related Function Documentation

#### 32.37.2.1 friend class `STDAIR_Service` [friend]

Definition at line 27 of file [CmdBomManager.hpp](#).

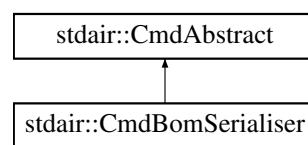
The documentation for this class was generated from the following file:

- [stdair/command/CmdBomManager.hpp](#)

### 32.38 `stdair::CmdBomSerialiser` Class Reference

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

Inheritance diagram for `stdair::CmdBomSerialiser`:



#### 32.38.1 Detailed Description

Class wrapping utility functions for handling the BOM tree objects.

Definition at line 25 of file [CmdBomSerialiser.hpp](#).

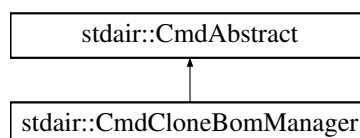
The documentation for this class was generated from the following file:

- [stdair/command/CmdBomSerialiser.hpp](#)

### 32.39 `stdair::CmdCloneBomManager` Class Reference

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

Inheritance diagram for `stdair::CmdCloneBomManager`:



#### Friends

- class [STDAIR\\_Service](#)

#### 32.39.1 Detailed Description

Class wrapping utility functions for handling the BOM tree objects.

Definition at line 40 of file [CmdCloneBomManager.hpp](#).

## 32.39.2 Friends And Related Function Documentation

## 32.39.2.1 friend class STDAIR\_Service [friend]

Definition at line 42 of file [CmdCloneBomManager.hpp](#).

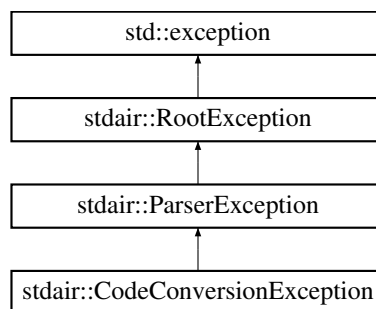
The documentation for this class was generated from the following file:

- [stdair/command/CmdCloneBomManager.hpp](#)

## 32.40 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](#)

## 32.40.1 Detailed Description

Code conversion.

Definition at line 133 of file [stdair\\_exceptions.hpp](#).

## 32.40.2 Constructor &amp; Destructor Documentation

## 32.40.2.1 stdair::CodeConversionException::CodeConversionException ( const std::string &amp;iWhat ) [inline]

Constructor.

Definition at line 136 of file [stdair\\_exceptions.hpp](#).

## 32.40.3 Member Function Documentation

## 32.40.3.1 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\(\)](#).

#### 32.40.4 Member Data Documentation

##### 32.40.4.1 `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\(\)](#).

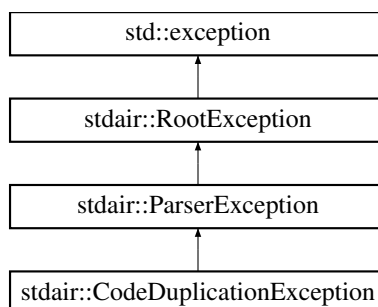
The documentation for this class was generated from the following file:

- [stdair/stdair\\_exceptions.hpp](#)

#### 32.41 `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](#)

##### 32.41.1 Detailed Description

Code duplication.

Definition at line 141 of file [stdair\\_exceptions.hpp](#).

##### 32.41.2 Constructor & Destructor Documentation

###### 32.41.2.1 `stdair::CodeDuplicationException::CodeDuplicationException ( const std::string & iWhat )` `[inline]`

Constructor.

Definition at line 144 of file [stdair\\_exceptions.hpp](#).

### 32.41.3 Member Function Documentation

#### 32.41.3.1 `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\(\)](#).

### 32.41.4 Member Data Documentation

#### 32.41.4.1 `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](#)

## 32.42 COMMAND Struct Reference

```
#include <stdair/ui/cmdline/readline_autocomp.hpp>
```

### Public Attributes

- `char const * name`
- `pt2Func * func`
- `char * doc`

### 32.42.1 Detailed Description

A structure which contains information on the commands this program can understand.

Definition at line 41 of file [readline\\_autocomp.hpp](#).

### 32.42.2 Member Data Documentation

#### 32.42.2.1 `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\(\)](#).

#### 32.42.2.2 `pt2Func* COMMAND::func`

Function to call to do the job.

Definition at line 50 of file [readline\\_autocomp.hpp](#).

Referenced by [execute\\_line\(\)](#).



### 32.42.2.3 char\* COMMAND::doc

Documentation for this function.

Definition at line 55 of file [readline\\_autocomp.hpp](#).

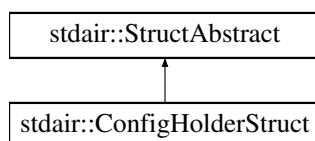
The documentation for this struct was generated from the following file:

- [stdair/ui/cmdline/readline\\_autocomp.hpp](#)

## 32.43 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

### 32.43.1 Detailed Description

Structure holding the configuration of the simulation.

Definition at line 40 of file [ConfigHolderStruct.hpp](#).

### 32.43.2 Constructor & Destructor Documentation

#### 32.43.2.1 stdair::ConfigHolderStruct::ConfigHolderStruct ( )

Constructor.

Definition at line 27 of file [ConfigHolderStruct.cpp](#).

#### 32.43.2.2 stdair::ConfigHolderStruct::ConfigHolderStruct ( const ConfigHolderStruct & iConfigHolderStruct )

Copy constructor.

Definition at line 32 of file [ConfigHolderStruct.cpp](#).

## 32.43.2.3 stdair::ConfigHolderStruct::~~ConfigHolderStruct ( )

Destructor.

Definition at line 37 of file [ConfigHolderStruct.cpp](#).

## 32.43.3 Member Function Documentation

## 32.43.3.1 void stdair::ConfigHolderStruct::add ( const bpt::ptree &amp; iConfigPropertyTree )

Merge the given property tree with the existing configuration property tree gathering all the configuration information.

Parameters

<i>const</i>	<a href="#">bpt::ptree</a> & 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\(\)](#).

## 32.43.3.2 bool stdair::ConfigHolderStruct::addValue ( const std::string &amp; iValue, const std::string &amp; 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

<i>const</i>	std::string& Value to add at the given path.
<i>const</i>	std::string& Path to create (or to look for).

Definition at line 191 of file [ConfigHolderStruct.cpp](#).

Referenced by [stdair::STDAIR\\_Service::importConfigValue\(\)](#).

## 32.43.3.3 template&lt;typename ValueType &gt; bool stdair::ConfigHolderStruct::exportValue ( ValueType &amp; ioValue, const std::string &amp; 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

<i>ValueType</i> &	Value to add in the configuration tree.
<i>const</i>	std::string& Path to look for.

Definition at line 144 of file [ConfigHolderStruct.hpp](#).

Referenced by [stdair::STDAIR\\_Service::exportConfigValue\(\)](#).

## 32.43.3.4 void stdair::ConfigHolderStruct::updateAirlineFeatures ( BomRoot &amp; iBomRoot )

Update the airline features objects thanks to the configuration holder.

Parameters

<i>BomRoot</i> &	Reference on the <a href="#">BomRoot</a> 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\(\)](#).

32.43.3.5 `void stdair::ConfigHolderStruct::toStream ( std::ostream & ioOut ) const`

Dump a Business Object into an output stream.

## Parameters

<i>ostream</i> &	the output stream.
------------------	--------------------

Definition at line 41 of file [ConfigHolderStruct.cpp](#).

References [describe\(\)](#).

32.43.3.6 void stdair::ConfigHolderStruct::fromStream ( std::istream & *ioIn* ) [virtual]

Read a Business Object from an input stream.

## Parameters

<i>istream</i> &	the input stream.
------------------	-------------------

Reimplemented from [stdair::StructAbstract](#).

Definition at line 46 of file [ConfigHolderStruct.cpp](#).

32.43.3.7 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\(\)](#).

32.43.3.8 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\(\)](#).

32.43.3.9 template<> bool stdair::ConfigHolderStruct::exportValue ( Date\_T & *ioValue*, const std::string & *iPath* ) const [inline]

Definition at line 175 of file [ConfigHolderStruct.hpp](#).

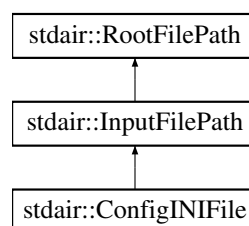
The documentation for this struct was generated from the following files:

- [stdair/bom/ConfigHolderStruct.hpp](#)
- [stdair/bom/ConfigHolderStruct.cpp](#)

## 32.44 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

### 32.44.1 Detailed Description

Config file: INI format

Definition at line 112 of file [stdair\\_file.hpp](#).

### 32.44.2 Constructor & Destructor Documentation

32.44.2.1 [stdair::ConfigINIFile::ConfigINIFile](#) ( const [Filename\\_T](#) &*iFilename* ) [\[inline\]](#), [\[explicit\]](#)

Constructor.

Definition at line 117 of file [stdair\\_file.hpp](#).

### 32.44.3 Member Function Documentation

32.44.3.1 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\(\)](#).

### 32.44.4 Member Data Documentation

32.44.4.1 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](#)

## 32.45 [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](#) ()

## 32.45.1 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](#).

## 32.45.2 Member Typedef Documentation

32.45.2.1 `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](#).

## 32.45.3 Constructor &amp; Destructor Documentation

32.45.3.1 `template<typename T > stdair::ContinuousAttributeLite< T >::ContinuousAttributeLite ( const ContinuousDistribution_T & iValueMap ) [inline]`

Constructor.

Definition at line 204 of file [ContinuousAttributeLite.hpp](#).

32.45.3.2 `template<typename T > stdair::ContinuousAttributeLite< T >::ContinuousAttributeLite ( const ContinuousAttributeLite< T > & iCAL ) [inline]`

Copy constructor.

Definition at line 212 of file [ContinuousAttributeLite.hpp](#).

32.45.3.3 `template<typename T > virtual stdair::ContinuousAttributeLite< T >::~~ContinuousAttributeLite ( ) [inline], [virtual]`

Destructor.

Definition at line 231 of file [ContinuousAttributeLite.hpp](#).

## 32.45.4 Member Function Documentation

32.45.4.1 `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\(\)](#).

**32.45.4.2** `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\(\)](#).

**32.45.4.3** `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\(\)](#).

**32.45.4.4** `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](#).

**32.45.4.5** `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\(\)](#).

**32.45.4.6** `template<typename T> ContinuousAttributeLite& stdair::ContinuousAttributeLite< T>::operator= ( const ContinuousAttributeLite< T> & iCAL ) [inline]`

Copy operator.

Definition at line 221 of file [ContinuousAttributeLite.hpp](#).

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

- [stdair/basic/ContinuousAttributeLite.hpp](#)

## 32.46 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](#)

## 32.46.1 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](#).

## 32.46.2 Constructor &amp; Destructor Documentation

```
32.46.2.1  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](#).

```
32.46.2.2  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](#).

```
32.46.2.3  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](#).

## 32.46.3 Member Function Documentation

```
32.46.3.1  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](#).

## 32.46.4 Member Data Documentation

```
32.46.4.1  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::operator\\*\(\)](#), and [stdair::operator+\(\)](#).

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

- [stdair/basic/BasParserHelperTypes.hpp](#)

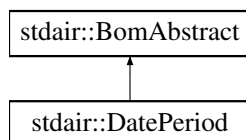
## 32.47 stdair::DatePeriod Class Reference

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

- template<typename BOM >  
class [FacBom](#)
- template<typename BOM >  
class [FacCloneBom](#)
- class [FacBomManager](#)

### 32.47.1 Detailed Description

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

Definition at line 18 of file [DatePeriod.hpp](#).

### 32.47.2 Member Typedef Documentation

#### 32.47.2.1 typedef [DatePeriodKey](#) [stdair::DatePeriod::Key\\_T](#)

Definition allowing to retrieve the associated BOM key type.

Definition at line 28 of file [DatePeriod.hpp](#).

## 32.47.3 Constructor &amp; Destructor Documentation

## 32.47.3.1 stdair::DatePeriod::DatePeriod ( const Key\_T &amp; iKey ) [protected]

Main constructor.

Definition at line 27 of file [DatePeriod.cpp](#).

## 32.47.3.2 stdair::DatePeriod::~~DatePeriod ( ) [protected], [virtual]

Destructor.

Definition at line 32 of file [DatePeriod.cpp](#).

## 32.47.4 Member Function Documentation

## 32.47.4.1 void stdair::DatePeriod::toStream ( std::ostream &amp; ioOut ) const [inline], [virtual]

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Implements [stdair::BomAbstract](#).

Definition at line 37 of file [DatePeriod.hpp](#).

References [toString\(\)](#).

## 32.47.4.2 void stdair::DatePeriod::fromStream ( std::istream &amp; ioIn ) [inline], [virtual]

Read a Business Object from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Implements [stdair::BomAbstract](#).

Definition at line 46 of file [DatePeriod.hpp](#).

## 32.47.4.3 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\(\)](#).

## 32.47.4.4 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\(\)](#).

## 32.47.4.5 const Key\_T&amp; stdair::DatePeriod::getKey ( ) const [inline]

Get the primary key (date period).

Definition at line 66 of file [DatePeriod.hpp](#).

References [\\_key](#).

32.47.4.6 **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](#).

32.47.4.7 **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](#).

32.47.4.8 **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\(\)](#).

32.47.4.9 **bool** `stdair::DatePeriod::isDepartureDateValid ( const Date_T & iFlightDate ) const`

Check if the given departure date is included in the departure period of the segment path.

Definition at line 44 of file [DatePeriod.cpp](#).

References [getDatePeriod\(\)](#).

Referenced by [stdair::BomRetriever::retrieveDatePeriodListFromKey\(\)](#).

## 32.47.5 Friends And Related Function Documentation

32.47.5.1 **template<typename BOM > friend class FacBom** `[friend]`

Definition at line 19 of file [DatePeriod.hpp](#).

32.47.5.2 **template<typename BOM > friend class FacCloneBom** `[friend]`

Definition at line 20 of file [DatePeriod.hpp](#).

32.47.5.3 **friend class FacBomManager** `[friend]`

Definition at line 21 of file [DatePeriod.hpp](#).

## 32.47.6 Member Data Documentation

32.47.6.1 **Key\_T** `stdair::DatePeriod::_key` `[protected]`

Primary key (date period).

Definition at line 126 of file [DatePeriod.hpp](#).

Referenced by [describeKey\(\)](#), [getDatePeriod\(\)](#), and [getKey\(\)](#).

**32.47.6.2 BomAbstract\* stdair::DatePeriod::\_parent** [protected]

Pointer on the parent class.

Definition at line 131 of file [DatePeriod.hpp](#).

Referenced by [getParent\(\)](#).

**32.47.6.3 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:

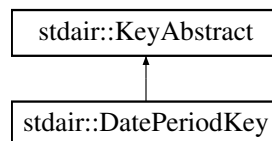
- [stdair/bom/DatePeriod.hpp](#)
- [stdair/bom/DatePeriod.cpp](#)

**32.48 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

**32.48.1 Detailed Description**

Key of date-period.

Definition at line 14 of file [DatePeriodKey.hpp](#).

**32.48.2 Constructor & Destructor Documentation****32.48.2.1 stdair::DatePeriodKey::DatePeriodKey ( const DatePeriod\_T & iDatePeriod )**

Main Constructor.

Definition at line 22 of file [DatePeriodKey.cpp](#).

### 32.48.2.2 `stdair::DatePeriodKey::DatePeriodKey ( const DatePeriodKey & iKey )`

Copy constructor.

Definition at line 27 of file [DatePeriodKey.cpp](#).

### 32.48.2.3 `stdair::DatePeriodKey::~~DatePeriodKey ( )`

Destructor.

Definition at line 32 of file [DatePeriodKey.cpp](#).

## 32.48.3 Member Function Documentation

### 32.48.3.1 `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\(\)](#).

### 32.48.3.2 `void stdair::DatePeriodKey::toStream ( std::ostream & ioOut ) const` [virtual]

Dump a Business Object Key into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 36 of file [DatePeriodKey.cpp](#).

References [toString\(\)](#).

### 32.48.3.3 `void stdair::DatePeriodKey::fromStream ( std::istream & ioIn )` [virtual]

Read a Business Object Key from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 41 of file [DatePeriodKey.cpp](#).

### 32.48.3.4 `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](#)

## 32.49 stdair::DbAbstract Class Reference

```
#include <stdair/dbadaptor/DbAbstract.hpp>
```

### Public Member Functions

- virtual [~DbAbstract](#) ()
- virtual void [toStream](#) (std::ostream &ioOut) const
- virtual void [fromStream](#) (std::istream &ioIn)

### Protected Member Functions

- [DbAbstract](#) ()

### 32.49.1 Detailed Description

Base class for the Database Adaptor (DBA) layer.

Definition at line 13 of file [DbAbstract.hpp](#).

### 32.49.2 Constructor & Destructor Documentation

**32.49.2.1** virtual stdair::DbAbstract::~~DbAbstract ( ) [\[inline\]](#), [\[virtual\]](#)

Destructor.

Definition at line 17 of file [DbAbstract.hpp](#).

**32.49.2.2** stdair::DbAbstract::DbAbstract ( ) [\[inline\]](#), [\[protected\]](#)

Protected Default Constructor to ensure this class is abstract.

Definition at line 29 of file [DbAbstract.hpp](#).

### 32.49.3 Member Function Documentation

**32.49.3.1** virtual void stdair::DbAbstract::toStream ( std::ostream & *ioOut* ) const [\[inline\]](#), [\[virtual\]](#)

Dump a Business Object into an output stream.

#### Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Definition at line 21 of file [DbAbstract.hpp](#).

**32.49.3.2** virtual void stdair::DbAbstract::fromStream ( std::istream & *ioIn* ) [\[inline\]](#), [\[virtual\]](#)

Read a Business Object from an input stream.

#### Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Definition at line 25 of file [DbAbstract.hpp](#).

Referenced by [operator>>\(\)](#).

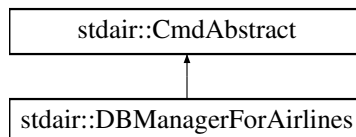
The documentation for this class was generated from the following file:

- stdair/dbadaptor/[DbAbstract.hpp](#)

### 32.50 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](#) &)

#### 32.50.1 Detailed Description

Class building the Business Object Model (BOM) from data retrieved from the database.

Definition at line 18 of file [DBManagerForAirlines.hpp](#).

#### 32.50.2 Member Function Documentation

**32.50.2.1** void stdair::DBManagerForAirlines::updateAirlineInDB ( DBSession\_T & *ioSociSession*, const [AirlineStruct](#) & *iAirline* ) [static]

Update the fields of the database row corresponding to the given BOM object. DBSession\_T& [AirlineStruct](#)& .

Definition at line 99 of file [DBManagerForAirlines.cpp](#).

References [stdair::AirlineStruct::getAirlineCode\(\)](#), and [stdair::AirlineStruct::getAirlineName\(\)](#).

**32.50.2.2** bool stdair::DBManagerForAirlines::retrieveAirline ( DBSession\_T & *ioSociSession*, const [AirlineCode\\_T](#) & *iAirlineCode*, [AirlineStruct](#) & *ioAirline* ) [static]

Retrieve, from the (MySQL) database, the row corresponding to the given BOM code, and fill the given BOM object with that retrieved data. DBSession\_T& const [AirlineCode\\_T](#)& [AirlineStruct](#)& .

Definition at line 134 of file [DBManagerForAirlines.cpp](#).

References [iterateOnStatement\(\)](#).

**32.50.2.3** void stdair::DBManagerForAirlines::prepareSelectStatement ( DBSession\_T & *ioSociSession*, [DBRequestStatement\\_T](#) & *ioSelectStatement*, [AirlineStruct](#) & *ioAirline* ) [static]

Prepare (parse and put in cache) the SQL statement. DBSession\_T& [DBRequestStatement\\_T](#)& [AirlineStruct](#)& .

Definition at line 26 of file [DBManagerForAirlines.cpp](#).

**32.50.2.4** bool stdair::DBManagerForAirlines::iterateOnStatement ( [DBRequestStatement\\_T](#) & *ioStatement*, [AirlineStruct](#) & *ioAirline* ) [static]

Iterate on the SQL statement.

The SQL has to be already prepared. [DBRequestStatement\\_T](#)& [AirlineStruct](#)& .

Definition at line 82 of file [DBManagerForAirlines.cpp](#).

Referenced by [retrieveAirline\(\)](#).

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

- [stdair/command/DBManagerForAirlines.hpp](#)
- [stdair/command/DBManagerForAirlines.cpp](#)

## 32.51 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](#)

#### 32.51.1 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](#).

#### 32.51.2 Member Function Documentation

##### 32.51.2.1 DBSessionManager & stdair::DBSessionManager::instance ( ) [static]

Return the static [DBSessionManager](#) instance.

Definition at line 82 of file [DBSessionManager.cpp](#).

##### 32.51.2.2 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](#).

#### 32.51.3 Friends And Related Function Documentation

##### 32.51.3.1 friend class FacSupervisor [friend]

Definition at line 19 of file [DBSessionManager.hpp](#).



### 32.51.3.2 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](#)

## 32.52 stdair::DefaultDCPList Struct Reference

```
#include <stdair/basic/BasConst_Inventory.hpp>
```

### Static Public Member Functions

- static [DCPList\\_T](#) init ()

### 32.52.1 Detailed Description

Definition at line 126 of file [BasConst\\_Inventory.hpp](#).

### 32.52.2 Member Function Documentation

#### 32.52.2.1 [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](#)

## 32.53 stdair::DefaultDtdFratMap Struct Reference

```
#include <stdair/basic/BasConst_Inventory.hpp>
```

### Static Public Member Functions

- static [DtdFratMap\\_T](#) init ()

### 32.53.1 Detailed Description

Definition at line 130 of file [BasConst\\_Inventory.hpp](#).

### 32.53.2 Member Function Documentation

#### 32.53.2.1 [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](#)

## 32.54 stdair::DefaultDtdProbMap Struct Reference

```
#include <stdair/basic/BasConst_Inventory.hpp>
```

### Static Public Member Functions

- static [DtdProbMap\\_T](#) init ()

#### 32.54.1 Detailed Description

Definition at line 134 of file [BasConst\\_Inventory.hpp](#).

#### 32.54.2 Member Function Documentation

##### 32.54.2.1 [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](#)

## 32.55 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 ()

#### 32.55.1 Detailed Description

FRAT5 curves.

Definition at line 27 of file [BasConst\\_SellUpCurves.hpp](#).

#### 32.55.2 Member Function Documentation

##### 32.55.2.1 [FRAT5Curve\\_T](#) stdair::DefaultMap::createFRAT5CurveA ( ) [static]

Definition at line 533 of file [BasConst.cpp](#).

32.55.2.2 **FRAT5Curve\_T** stdair::DefaultMap::createFRAT5CurveB ( ) [static]

Definition at line 547 of file [BasConst.cpp](#).

32.55.2.3 **FRAT5Curve\_T** stdair::DefaultMap::createFRAT5CurveC ( ) [static]

Definition at line 561 of file [BasConst.cpp](#).

32.55.2.4 **FRAT5Curve\_T** stdair::DefaultMap::createFRAT5CurveD ( ) [static]

Definition at line 575 of file [BasConst.cpp](#).

32.55.2.5 **FFDisutilityCurve\_T** stdair::DefaultMap::createFFDisutilityCurveA ( ) [static]

Definition at line 593 of file [BasConst.cpp](#).

32.55.2.6 **FFDisutilityCurve\_T** stdair::DefaultMap::createFFDisutilityCurveB ( ) [static]

Definition at line 611 of file [BasConst.cpp](#).

32.55.2.7 **FFDisutilityCurve\_T** stdair::DefaultMap::createFFDisutilityCurveC ( ) [static]

Definition at line 629 of file [BasConst.cpp](#).

32.55.2.8 **FFDisutilityCurve\_T** stdair::DefaultMap::createFFDisutilityCurveD ( ) [static]

Definition at line 647 of file [BasConst.cpp](#).

32.55.2.9 **FFDisutilityCurve\_T** stdair::DefaultMap::createFFDisutilityCurveE ( ) [static]

Definition at line 665 of file [BasConst.cpp](#).

32.55.2.10 **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:

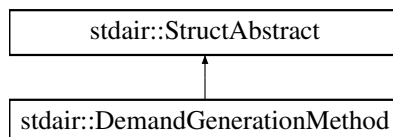
- stdair/basic/[BasConst\\_SellUpCurves.hpp](#)
- stdair/basic/[BasConst.cpp](#)

## 32.56 stdair::DemandGenerationMethod Struct Reference

Enumeration of demand (booking request) generation methods.

```
#include <stdair/basic/DemandGenerationMethod.hpp>
```

Inheritance diagram for stdair::DemandGenerationMethod:



### Public Types

- enum [EN\\_DemandGenerationMethod](#) { [POI\\_PRO](#) = 0, [STA\\_ORD](#), [LAST\\_VALUE](#) }

## Public Member Functions

- [EN\\_DemandGenerationMethod](#) [getMethod](#) () const
- char [getMethodAsChar](#) () const
- std::string [getMethodAsString](#) () const
- const std::string [describe](#) () const
- bool [operator==](#) (const [EN\\_DemandGenerationMethod](#) &) const
- [DemandGenerationMethod](#) (const [EN\\_DemandGenerationMethod](#) &)
- [DemandGenerationMethod](#) (const char iMethod)
- [DemandGenerationMethod](#) (const std::string &iMethod)
- [DemandGenerationMethod](#) (const [DemandGenerationMethod](#) &)
- void [toStream](#) (std::ostream &ioOut) const
- virtual void [fromStream](#) (std::istream &ioIn)

## Static Public Member Functions

- static const std::string & [getLabel](#) (const [EN\\_DemandGenerationMethod](#) &)
- static [EN\\_DemandGenerationMethod](#) [getMethod](#) (const char)
- static char [getMethodLabel](#) (const [EN\\_DemandGenerationMethod](#) &)
- static std::string [getMethodLabelAsString](#) (const [EN\\_DemandGenerationMethod](#) &)
- static std::string [describeLabels](#) ()

## 32.56.1 Detailed Description

Enumeration of demand (booking request) generation methods.

Definition at line 17 of file [DemandGenerationMethod.hpp](#).

## 32.56.2 Member Enumeration Documentation

## 32.56.2.1 enum stdair::DemandGenerationMethod::EN\_DemandGenerationMethod

Enumerator

***POI\_PRO***  
***STA\_ORD***  
***LAST\_VALUE***

Definition at line 19 of file [DemandGenerationMethod.hpp](#).

## 32.56.3 Constructor &amp; Destructor Documentation

32.56.3.1 stdair::DemandGenerationMethod::DemandGenerationMethod ( const [EN\\_DemandGenerationMethod](#) & *iDemandGenerationMethod* )

Main constructor.

Definition at line 34 of file [DemandGenerationMethod.cpp](#).

32.56.3.2 stdair::DemandGenerationMethod::DemandGenerationMethod ( const char *iMethod* )

Alternative constructor.

Definition at line 62 of file [DemandGenerationMethod.cpp](#).

32.56.3.3 `stdair::DemandGenerationMethod::DemandGenerationMethod ( const std::string & iMethod )`

Alternative constructor.

Definition at line 68 of file [DemandGenerationMethod.cpp](#).

References [getMethod\(\)](#).

32.56.3.4 `stdair::DemandGenerationMethod::DemandGenerationMethod ( const DemandGenerationMethod & iDemandGenerationMethod )`

Default copy constructor.

Definition at line 28 of file [DemandGenerationMethod.cpp](#).

## 32.56.4 Member Function Documentation

32.56.4.1 `const std::string & stdair::DemandGenerationMethod::getLabel ( const EN_DemandGenerationMethod & iMethod ) [static]`

Get the label as a string (e.g., "PoissonProcess" or "StatisticsOrder").

Definition at line 78 of file [DemandGenerationMethod.cpp](#).

32.56.4.2 `DemandGenerationMethod::EN_DemandGenerationMethod stdair::DemandGenerationMethod::get↔Method ( const char iMethodChar ) [static]`

Get the method value from parsing a single char (e.g., 'P' or 'S').

Definition at line 40 of file [DemandGenerationMethod.cpp](#).

References [describeLabels\(\)](#), [LAST\\_VALUE](#), [POI\\_PRO](#), and [STA\\_ORD](#).

32.56.4.3 `char stdair::DemandGenerationMethod::getMethodLabel ( const EN_DemandGenerationMethod & iMethod ) [static]`

Get the label as a single char (e.g., 'P' or 'S').

Definition at line 84 of file [DemandGenerationMethod.cpp](#).

32.56.4.4 `std::string stdair::DemandGenerationMethod::getMethodLabelAsString ( const EN_DemandGenerationMethod & iMethod ) [static]`

Get the label as a string of a single char (e.g., "P" or "S").

Definition at line 90 of file [DemandGenerationMethod.cpp](#).

32.56.4.5 `std::string stdair::DemandGenerationMethod::describeLabels ( ) [static]`

List the labels.

Definition at line 97 of file [DemandGenerationMethod.cpp](#).

References [LAST\\_VALUE](#).

Referenced by [getMethod\(\)](#).

32.56.4.6 `DemandGenerationMethod::EN_DemandGenerationMethod stdair::DemandGenerationMethod::get↔Method ( ) const`

Get the enumerated value.

Definition at line 110 of file [DemandGenerationMethod.cpp](#).

Referenced by [DemandGenerationMethod\(\)](#).

32.56.4.7 `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](#).

32.56.4.8 `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](#).

32.56.4.9 `const std::string stdair::DemandGenerationMethod::describe ( ) const` `[virtual]`

Give a description of the structure (e.g., "PoissonProcess" or "StatisticsOrder").

Implements [stdair::StructAbstract](#).

Definition at line 128 of file [DemandGenerationMethod.cpp](#).

32.56.4.10 `bool stdair::DemandGenerationMethod::operator== ( const EN_DemandGenerationMethod & iMethod ) const`

Comparison operator.

Definition at line 136 of file [DemandGenerationMethod.cpp](#).

32.56.4.11 `void stdair::StructAbstract::toStream ( std::ostream & ioOut ) const` `[inline]`, `[inherited]`

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Definition at line 29 of file [StructAbstract.hpp](#).

References [stdair::StructAbstract::describe\(\)](#).

32.56.4.12 `virtual void stdair::StructAbstract::fromStream ( std::istream & ioIn )` `[inline]`, `[virtual]`, `[inherited]`

Read a Business Object from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented in [stdair::BookingRequestStruct](#), [stdair::EventStruct](#), [stdair::TravelSolutionStruct](#), [stdair::VirtualClassStruct](#), [stdair::ConfigHolderStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::ProgressStatusSet](#), [stdair::YieldRange](#), [stdair::CancellationStruct](#), [stdair::AirlineStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::FFDisutilityCurveHolderStruct](#), [stdair::FRAT5CurveHolderStruct](#), and [stdair::BreakPointStruct](#).

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](#)

## 32.57 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](#))

#### 32.57.1 Detailed Description

Class wrapper of dictionary business methods.

Definition at line 22 of file [DictionaryManager.hpp](#).

#### 32.57.2 Member Function Documentation

**32.57.2.1** `const stdair::Probability_T stdair::DictionaryManager::keyToValue ( const DictionaryKey_T iKey )`  
[static]

Convert from key to value.

Definition at line 12 of file [DictionaryManager.cpp](#).

References [stdair::DEFAULT\\_NUMBER\\_OF\\_SUBDIVISIONS](#).

Referenced by [stdair::ContinuousAttributeLite< T >::displayCumulativeDistribution\(\)](#), [stdair::ContinuousAttributeLite< T >::getDerivativeValue\(\)](#), [stdair::ContinuousAttributeLite< T >::getRemainingProportion\(\)](#), and [stdair::ContinuousAttributeLite< T >::getValue\(\)](#).

**32.57.2.2** `const DictionaryKey_T stdair::DictionaryManager::valueToKey ( const stdair::Probability_T iValue )`  
[static]

Convert from value to key.

Definition at line 21 of file [DictionaryManager.cpp](#).

References [stdair::DEFAULT\\_NUMBER\\_OF\\_SUBDIVISIONS](#).

Referenced by [stdair::ContinuousAttributeLite< T >::getValue\(\)](#).

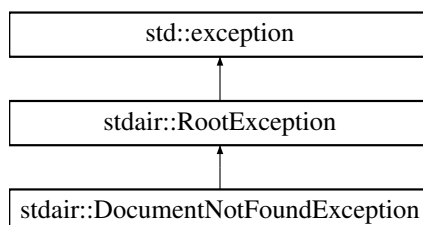
The documentation for this class was generated from the following files:

- [stdair/basic/DictionaryManager.hpp](#)
- [stdair/basic/DictionaryManager.cpp](#)

### 32.58 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`

## 32.58.1 Detailed Description

Document not found.

Definition at line 104 of file [stdair\\_exceptions.hpp](#).

## 32.58.2 Constructor &amp; Destructor Documentation

32.58.2.1 `stdair::DocumentNotFoundException::DocumentNotFoundException ( const std::string & iWhat ) [inline]`

Constructor.

Definition at line 107 of file [stdair\\_exceptions.hpp](#).

## 32.58.3 Member Function Documentation

32.58.3.1 `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\(\)](#).

## 32.58.4 Member Data Documentation

32.58.4.1 `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\(\)](#).

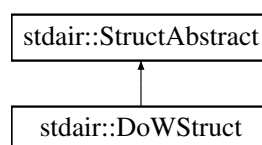
The documentation for this class was generated from the following file:

- [stdair/stdair\\_exceptions.hpp](#)

## 32.59 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)

## 32.59.1 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](#).

## 32.59.2 Member Typedef Documentation

## 32.59.2.1 typedef std::vector&lt;bool&gt; stdair::DoWStruct::BooleanList\_T

Define the bit set representing the DoW.

Definition at line 21 of file [DoWStruct.hpp](#).

## 32.59.3 Constructor &amp; Destructor Documentation

## 32.59.3.1 stdair::DoWStruct::DoWStruct ( const std::string &amp; iDowString )

Constructor from a given bit set (e.g., "0000011" for the week-ends).

Definition at line 21 of file [DoWStruct.cpp](#).

## 32.59.3.2 stdair::DoWStruct::DoWStruct ( )

Default constructors.

Definition at line 14 of file [DoWStruct.cpp](#).

## 32.59.3.3 stdair::DoWStruct::DoWStruct ( const DoWStruct &amp; iDowStruct )

Definition at line 34 of file [DoWStruct.cpp](#).

## 32.59.3.4 stdair::DoWStruct::~~DoWStruct ( ) [inline]

Default destructor.

Definition at line 63 of file [DoWStruct.hpp](#).

## 32.59.4 Member Function Documentation

32.59.4.1 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\(\)](#).

32.59.4.2 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](#).

32.59.4.3 void stdair::DoWStruct::setDayOfWeek ( const unsigned short *i*, const bool *iBool* )

Set the new value for the *i*-th day-of-week.

Definition at line 82 of file [DoWStruct.cpp](#).

Referenced by [intersection\(\)](#), and [shift\(\)](#).

## 32.59.4.4 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\(\)](#).

## 32.59.4.5 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\(\)](#).

32.59.4.6 DoWStruct stdair::DoWStruct::shift ( const long & *iNbOfDays* ) const

Build a new DoW struct by shifting the current DoW by a given number.

Definition at line 88 of file [DoWStruct.cpp](#).

References [stdair::DEFAULT\\_DOW\\_STRING](#), and [setDayOfWeek\(\)](#).

Referenced by [stdair::PeriodStruct::addDateOffset\(\)](#).

32.59.4.7 DoWStruct stdair::DoWStruct::intersection ( const DoWStruct & *iDoW* ) const

Build a new DoW struct by intersecting two DoW structs.

Definition at line 104 of file [DoWStruct.cpp](#).

References [stdair::DEFAULT\\_DOW\\_STRING](#), [getDayOfWeek\(\)](#), and [setDayOfWeek\(\)](#).

Referenced by [stdair::PeriodStruct::intersection\(\)](#).

## 32.59.4.8 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\(\)](#).

**32.59.4.9** `void stdair::StructAbstract::toStream ( std::ostream & ioOut ) const` `[inline],[inherited]`

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Definition at line 29 of file [StructAbstract.hpp](#).

References [stdair::StructAbstract::describe\(\)](#).

**32.59.4.10** `virtual void stdair::StructAbstract::fromStream ( std::istream & ioIn )` `[inline],[virtual],[inherited]`

Read a Business Object from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented in [stdair::BookingRequestStruct](#), [stdair::EventStruct](#), [stdair::TravelSolutionStruct](#), [stdair::VirtualClassStruct](#), [stdair::ConfigHolderStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::ProgressStatusSet](#), [stdair::YieldRange](#), [stdair::CancellationStruct](#), [stdair::AirlineStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::FFDisutilityCurveHolderStruct](#), [stdair::FRAT5CurveHolderStruct](#), and [stdair::BreakPointStruct](#).

Definition at line 38 of file [StructAbstract.hpp](#).

Referenced by [operator>>\(\)](#).

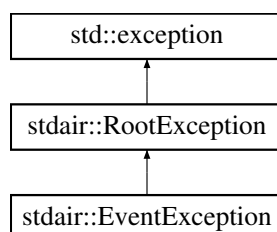
The documentation for this struct was generated from the following files:

- [stdair/bom/DoWStruct.hpp](#)
- [stdair/bom/DoWStruct.cpp](#)

## 32.60 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](#)

## 32.60.1 Detailed Description

Event.

Definition at line 204 of file [stdair\\_exceptions.hpp](#).

## 32.60.2 Constructor &amp; Destructor Documentation

32.60.2.1 stdair::EventException::EventException ( const std::string & *iWhat* ) [inline]

Constructor.

Definition at line 207 of file [stdair\\_exceptions.hpp](#).

## 32.60.3 Member Function Documentation

## 32.60.3.1 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\(\)](#).

## 32.60.4 Member Data Documentation

## 32.60.4.1 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\(\)](#).

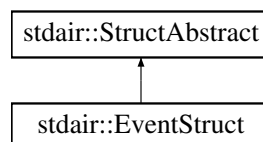
The documentation for this class was generated from the following file:

- [stdair/stdair\\_exceptions.hpp](#)

## 32.61 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 &iIn)
- 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 &iOut) const

### 32.61.1 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 be specified to start before that date-time.

Definition at line 36 of file [EventStruct.hpp](#).

### 32.61.2 Constructor & Destructor Documentation

#### 32.61.2.1 [stdair::EventStruct::EventStruct](#) ( )

Default constructor.

Definition at line 26 of file [EventStruct.cpp](#).

#### 32.61.2.2 [stdair::EventStruct::EventStruct](#) ( const [EventType::EN\\_EventType](#) & *iEventType*, [BookingRequestPtr\\_T](#) *ioRequestPtr* )

Constructor for events corresponding to booking requests.

Definition at line 31 of file [EventStruct.cpp](#).

References [stdair::DEFAULT\\_EVENT\\_OLDEST\\_DATETIME](#).

#### 32.61.2.3 [stdair::EventStruct::EventStruct](#) ( const [EventType::EN\\_EventType](#) & *iEventType*, [CancellationPtr\\_T](#) *ioCancellationPtr* )

Constructor for events corresponding to cancellations.

Definition at line 55 of file [EventStruct.cpp](#).

References [stdair::DEFAULT\\_EVENT\\_OLDEST\\_DATETIME](#).

32.61.2.4 `stdair::EventStruct::EventStruct ( const EventType::EN_EventType & iEventType, const DateTime_T & iDCPDate, OptimisationNotificationPtr_T ioOptimisationNotificationPtr )`

Constructor for events corresponding to optimisation requests.

Definition at line 80 of file [EventStruct.cpp](#).

References [stdair::DEFAULT\\_EVENT\\_OLDEST\\_DATETIME](#).

32.61.2.5 `stdair::EventStruct::EventStruct ( const EventType::EN_EventType & iEventType, SnapshotPtr_T ioSnapshotPtr )`

Constructor for events corresponding to snapshot requests.

Definition at line 105 of file [EventStruct.cpp](#).

References [stdair::DEFAULT\\_EVENT\\_OLDEST\\_DATETIME](#).

32.61.2.6 `stdair::EventStruct::EventStruct ( const EventType::EN_EventType & iEventType, RMEventPtr_T ioRMEventPtr )`

Constructor for events corresponding to RM events.

Definition at line 130 of file [EventStruct.cpp](#).

References [stdair::DEFAULT\\_EVENT\\_OLDEST\\_DATETIME](#).

32.61.2.7 `stdair::EventStruct::EventStruct ( const EventType::EN_EventType & iEventType, BreakPointPtr_T ioBreakPointPtr )`

Constructor for events corresponding to Break Point events.

Definition at line 155 of file [EventStruct.cpp](#).

References [stdair::DEFAULT\\_EVENT\\_OLDEST\\_DATETIME](#).

32.61.2.8 `stdair::EventStruct::EventStruct ( const EventStruct & iEventStruct )`

Copy constructor.

Definition at line 180 of file [EventStruct.cpp](#).

32.61.2.9 `stdair::EventStruct::~EventStruct ( )`

Destructor.

Definition at line 243 of file [EventStruct.cpp](#).

### 32.61.3 Member Function Documentation

32.61.3.1 `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\(\)](#).

32.61.3.2 `const LongDuration_T& stdair::EventStruct::getEventTimeStamp ( ) const` [inline]

Get the event time stamp

Definition at line 46 of file [EventStruct.hpp](#).

### 32.61.3.3 `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](#).

### 32.61.3.4 `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\(\)](#).

### 32.61.3.5 `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](#).

### 32.61.3.6 `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](#).

### 32.61.3.7 `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](#).

### 32.61.3.8 `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](#).

32.61.3.9 `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\(\)](#).

32.61.3.10 `void stdair::EventStruct::fromStream ( std::istream & ioln )` `[virtual]`

Read a Business Object from an input stream.

#### Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented from [stdair::StructAbstract](#).

Definition at line 247 of file [EventStruct.cpp](#).

32.61.3.11 `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::Event↵  
Type::RM](#), and [stdair::EventType::SNAPSHOT](#).

32.61.3.12 `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](#).

32.61.3.13 `void stdair::StructAbstract::toStream ( std::ostream & ioOut ) const` `[inline]`, `[inherited]`

Dump a Business Object into an output stream.

#### Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Definition at line 29 of file [StructAbstract.hpp](#).

References [stdair::StructAbstract::describe\(\)](#).

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

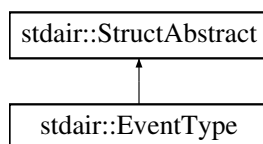
- [stdair/bom/EventStruct.hpp](#)
- [stdair/bom/EventStruct.cpp](#)

## 32.62 stdair::EventType Struct Reference

```
#include <stdair/basic/EventType.hpp>
```

Inheritance diagram for [stdair::EventType](#):





## Public Types

- enum [EN\\_EventType](#) {  
[BKG\\_REQ](#) = 0, [CX](#), [OPT\\_NOT\\_4\\_FD](#), [OPT\\_NOT\\_4\\_NET](#),  
[SKD\\_CHG](#), [SNAPSHOT](#), [RM](#), [BRK\\_PT](#),  
[LAST\\_VALUE](#) }

## Public Member Functions

- [EN\\_EventType](#) [getType](#) () const
- std::string [getTypeAsString](#) () const
- const std::string [describe](#) () const
- bool [operator==](#) (const [EN\\_EventType](#) &) const
- [EventType](#) (const [EN\\_EventType](#) &)
- [EventType](#) (const char iType)
- [EventType](#) (const 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](#) ()

### 32.62.1 Detailed Description

Enumeration of event types.

Definition at line 15 of file [EventType.hpp](#).

### 32.62.2 Member Enumeration Documentation

#### 32.62.2.1 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](#).

### 32.62.3 Constructor & Destructor Documentation

#### 32.62.3.1 stdair::EventType::EventType ( const EN\_EventType & iEventType )

Constructor.

Definition at line 36 of file [EventType.cpp](#).

#### 32.62.3.2 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](#).

#### 32.62.3.3 stdair::EventType::EventType ( const std::string & iTypeStr )

Constructor using a string.

Definition at line 64 of file [EventType.cpp](#).

References [describeLabels\(\)](#), and [LAST\\_VALUE](#).

#### 32.62.3.4 stdair::EventType::EventType ( const EventType & iEventType )

Default copy constructor.

Definition at line 31 of file [EventType.cpp](#).

### 32.62.4 Member Function Documentation

#### 32.62.4.1 const std::string & stdair::EventType::getLabel ( const EN\_EventType & iType ) [static]

Get the label as a string (e.g., "BookingRequest", "Cancellation", "OptimisationNotificationForFlightDate", "↔ OptimisationNotificationForNetwork", "ScheduleChange", "Snapshot", "RevenueManagement", "BreakPoint" or "↔ BookingRequest").

Definition at line 83 of file [EventType.cpp](#).

Referenced by [stdair::EventStruct::describe\(\)](#), [stdair::BomJSONExport::jsonExportBookingRequestObject\(\)](#), and [stdair::BomJSONExport::jsonExportBreakPointObject\(\)](#).

#### 32.62.4.2 char stdair::EventType::getTypeLabel ( const EN\_EventType & iType ) [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](#).

#### 32.62.4.3 std::string stdair::EventType::getTypeLabelAsString ( const EN\_EventType & iType ) [static]

Get the label as a string of a single char (e.g., "B", "X", "F", "N", "C", "S", "R" or "P").

Definition at line 93 of file [EventType.cpp](#).

#### 32.62.4.4 std::string stdair::EventType::describeLabels ( ) [static]

List the labels.

Definition at line 100 of file [EventType.cpp](#).

References [LAST\\_VALUE](#).

Referenced by [EventType\(\)](#).

#### 32.62.4.5 `EventType::EN_EventType stdair::EventType::getType ( ) const`

Get the enumerated value.

Definition at line 112 of file [EventType.cpp](#).

#### 32.62.4.6 `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](#).

#### 32.62.4.7 `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](#).

#### 32.62.4.8 `bool stdair::EventType::operator==( const EN_EventType & iType ) const`

Comparison operator.

Definition at line 131 of file [EventType.cpp](#).

#### 32.62.4.9 `void stdair::StructAbstract::toStream ( std::ostream & ioOut ) const` `[inline],[inherited]`

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Definition at line 29 of file [StructAbstract.hpp](#).

References [stdair::StructAbstract::describe\(\)](#).

#### 32.62.4.10 `virtual void stdair::StructAbstract::fromStream ( std::istream & ioIn )` `[inline],[virtual],[inherited]`

Read a Business Object from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented in [stdair::BookingRequestStruct](#), [stdair::EventStruct](#), [stdair::TravelSolutionStruct](#), [stdair::Virtual↔ ClassStruct](#), [stdair::ConfigHolderStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::↔ ProgressStatusSet](#), [stdair::YieldRange](#), [stdair::CancellationStruct](#), [stdair::AirlineStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::FFDisutilityCurveHolderStruct](#), [stdair::FRAT5CurveHolderStruct](#), and [stdair::Break↔ PointStruct](#).

Definition at line 38 of file [StructAbstract.hpp](#).

Referenced by [operator>>\(\)](#).

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

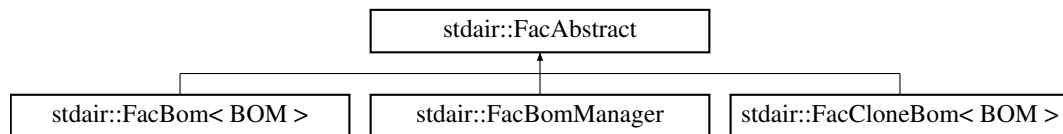
- [stdair/basic/EventType.hpp](#)

- [stdair/basic/EventType.cpp](#)

### 32.63 stdair::FacAbstract Class Reference

```
#include <stdair/factory/FacAbstract.hpp>
```

Inheritance diagram for stdair::FacAbstract:



#### Public Member Functions

- virtual [~FacAbstract](#) ()

#### Protected Member Functions

- [FacAbstract](#) ()

#### 32.63.1 Detailed Description

Base class for Factory layer.

Definition at line 10 of file [FacAbstract.hpp](#).

#### 32.63.2 Constructor & Destructor Documentation

##### 32.63.2.1 stdair::FacAbstract::~~FacAbstract ( ) [virtual]

Destructor.

Definition at line 13 of file [FacAbstract.cpp](#).

##### 32.63.2.2 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:

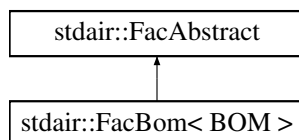
- [stdair/factory/FacAbstract.hpp](#)
- [stdair/factory/FacAbstract.cpp](#)

### 32.64 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](#) ()

#### 32.64.1 Detailed Description

`template<typename BOM>class stdair::FacBom< BOM >`

Base class for Factory layer.

Definition at line 22 of file [FacBom.hpp](#).

#### 32.64.2 Constructor & Destructor Documentation

**32.64.2.1** `template<typename BOM > stdair::FacBom< BOM >::FacBom ( ) [inline],[protected]`

Default Constructor.

Definition at line 50 of file [FacBom.hpp](#).

**32.64.2.2** `template<typename BOM > stdair::FacBom< BOM >::~~FacBom ( ) [inline]`

Destructor.

Definition at line 56 of file [FacBom.hpp](#).

References [stdair::FacBom< BOM >::clean\(\)](#).

#### 32.64.3 Member Function Documentation

**32.64.3.1** `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::FacSupervisor::instance\(\)](#), and [stdair::FacSupervisor::registerPersistentBomFactory\(\)](#).

32.64.3.2 `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](#).

Referenced by [stdair::FacBomManager::addBomHolder\(\)](#).

32.64.3.3 `template<typename BOM > BOM & stdair::FacBom< BOM >::create ( const Key_T & iKey )`

Definition at line 118 of file [FacBom.hpp](#).

32.64.3.4 `template<typename BOM > BOM & stdair::FacBom< BOM >::create ( const BOM & iBom )`

Definition at line 126 of file [FacBom.hpp](#).

32.64.3.5 `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:

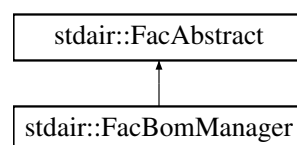
- [stdair/factory/FacBom.hpp](#)

## 32.65 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 ()`

#### 32.65.1 Detailed Description

Utility class for linking StdAir-based objects.

Definition at line 30 of file [FacBomManager.hpp](#).

#### 32.65.2 Constructor & Destructor Documentation

##### 32.65.2.1 `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](#).

##### 32.65.2.2 `stdair::FacBomManager::~~FacBomManager ( )` `[inline]`

Destructor.

Definition at line 231 of file [FacBomManager.hpp](#).

#### 32.65.3 Member Function Documentation

##### 32.65.3.1 `template<typename OBJECT2 , typename OBJECT1 > BomHolder< OBJECT2 > *` `stdair::FacBomManager::getBomHolderPtr ( OBJECT1 & ioObject1 )` `[static]`

Retrieve a pointer on the holder of children (OBJECT2 type) for the parent (OBJECT1 type). If the holder does not exist, return NULL.

## Parameters

<i>typename</i>	OBJECT1& Parent object.
-----------------	-------------------------

## Returns

*typename* BomHolder<OBJECT2>\* [BomHolder](#) for the children objects.

Definition at line 268 of file [FacBomManager.hpp](#).

```
32.65.3.2 template<typename OBJECT2 , typename OBJECT1 > BomHolder< OBJECT2 > &
stdair::FacBomManager::addBomHolder ( OBJECT1 & ioObject1 ) [static]
```

Instantiate a BomHolder<OBJECT2> object, add it to the OBJECT1-typed object, given as parameter, and return a reference on that newly created [BomHolder](#).

## Parameters

<i>typename</i>	OBJECT1& Parent object.
-----------------	-------------------------

## Returns

*typename* BomHolder<OBJECT2>& Just created [BomHolder](#) (e.g., for the children objects).

Definition at line 238 of file [FacBomManager.hpp](#).

References [stdair::FacBom< BOM >::create\(\)](#).

```
32.65.3.3 template<typename OBJECT1 , typename OBJECT2 > void stdair::FacBomManager::addToList ( OBJECT1 &
ioObject1, OBJECT2 & ioObject2 ) [static]
```

Add an OBJECT2-typed object (typically, a child) to the dedicated list held by the OBJECT1-typed object (typically, a parent).

## Note

The underlying list is actually stored within an object of type BomHolder<OBJECT2>.

## Parameters

<i>typename</i>	OBJECT1& Parent object.
<i>typename</i>	OBJECT2& Child object.

Definition at line 354 of file [FacBomManager.hpp](#).

```
32.65.3.4 template<typename OBJECT1 , typename OBJECT2 > void stdair::FacBomManager::addToMap ( OBJECT1 &
ioObject1, OBJECT2 & ioObject2, const MapKey_T & iKey ) [static]
```

Add an OBJECT2-typed object (typically, a child) to the dedicated map held by the OBJECT1-typed object (typically, a parent).

## Note

The underlying map is actually stored within an object of type BomHolder<OBJECT2>.

## Parameters



<i>typename</i>	OBJECT1& Parent object.
<i>typename</i>	OBJECT2& Child object.
<i>const</i>	MapKey_T&

Definition at line 424 of file [FacBomManager.hpp](#).

Referenced by [addToMap\(\)](#).

**32.65.3.5** `template<typename OBJECT1 , typename OBJECT2 > void stdair::FacBomManager::addToMap ( OBJECT1 & ioObject1, OBJECT2 & ioObject2 ) [static]`

Add an OBJECT2-typed object (typically, a child) to the dedicated map held by the OBJECT1-typed object (typically, a parent).

#### Note

The underlying map is actually stored within an object of type BomHolder<OBJECT2>.

#### Parameters

<i>typename</i>	OBJECT1& Parent object.
<i>typename</i>	OBJECT2& Child object.

Definition at line 446 of file [FacBomManager.hpp](#).

References [addToMap\(\)](#).

**32.65.3.6** `template<typename OBJECT1 , typename OBJECT2 > void stdair::FacBomManager::addToListAndMap ( OBJECT1 & ioObject1, OBJECT2 & ioObject2 ) [static]`

Add an OBJECT2-typed object (typically, a child) to the dedicated containers (list and map) held by the OBJECT1-typed object (typically, a parent).

#### Note

The underlying containers are actually stored within an object of type BomHolder<OBJECT2>.

#### Parameters

<i>typename</i>	OBJECT1& Parent object.
<i>typename</i>	OBJECT2& Child object.

Definition at line 490 of file [FacBomManager.hpp](#).

**32.65.3.7** `template<typename OBJECT1 , typename OBJECT2 > void stdair::FacBomManager::addToListAndMap ( OBJECT1 & ioObject1, OBJECT2 & ioObject2, const MapKey_T & iKey ) [static]`

Add an OBJECT2-typed object (typically, a child) to the dedicated containers (list and map) held by the OBJECT1-typed object (typically, a parent).

#### Note

The underlying containers are actually stored within an object of type BomHolder<OBJECT2>.

#### Parameters

<i>typename</i>	OBJECT1& Parent object.
<i>typename</i>	OBJECT2& Child object.

<i>const</i>	MapKey_T&
--------------	-----------

Definition at line 467 of file [FacBomManager.hpp](#).

32.65.3.8 **template**<typename PARENT , typename CHILD > void stdair::FacBomManager::linkWithParent ( PARENT & *ioParent*, CHILD & *ioChild* ) [static]

Allow the CHILD object to store a pointer on its PARENT object.

Parameters

<i>typename</i>	PARENT& Parent object.
<i>typename</i>	CHILD& Child object.

Definition at line 511 of file [FacBomManager.hpp](#).

Referenced by [stdair::serialiseHelper\(\)](#).

32.65.3.9 **template**<typename OBJECT2 , typename OBJECT1 > void stdair::FacBomManager::cloneHolder ( OBJECT1 & *ioDest*, const OBJECT1 & *ioOri* ) [static]

Clone the underlying containers (held by the BomHolder<OBJECT2>-typed holder) of the OBJECT1-typed object.

Note

The underlying containers are actually stored within an object of type BomHolder<OBJECT2>.

Parameters

<i>typename</i>	OBJECT1& Parent object.
<i>typename</i>	OBJECT2& Child object.

Definition at line 519 of file [FacBomManager.hpp](#).

References [stdair::BomHolder< BOM >::\\_bomList](#), and [stdair::BomHolder< BOM >::\\_bomMap](#).

32.65.3.10 void stdair::FacBomManager::resetYieldBasedNestingStructure ( const SegmentCabin & *iSegmentCabin* ) [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::BomHolder< BOM >::\\_bomList](#), [stdair::NestingNode::describeKey\(\)](#), [stdair::NestingNode::getHolderMap\(\)](#), [stdair::NestingNode::setYield\(\)](#), and [stdair::YIELD\\_BASED\\_NESTING\\_STRUCTURE\\_CODE](#).

32.65.3.11 **static void** stdair::FacBomManager::setAirlineFeature ( Inventory & *ilInventory*, AirlineFeature & *iAirlineFeature* ) [inline],[static]

Set the airline feature object of an inventory.

Definition at line 205 of file [FacBomManager.hpp](#).

32.65.3.12 **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](#).

32.65.3.13 **template**<> void stdair::FacBomManager::addToList ( SegmentDate & *ioSegmentDate*, SegmentDate & *ioMarketingSegmentDate* ) [inline]

Definition at line 539 of file [FacBomManager.hpp](#).

References [stdair::SegmentDate::\\_marketingSegmentDateList](#).

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

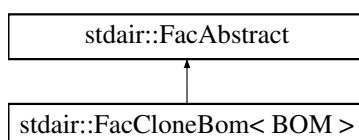
- [stdair/factory/FacBomManager.hpp](#)
- [stdair/factory/FacBomManager.cpp](#)

### 32.66 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](#) ()

#### 32.66.1 Detailed Description

```
template<typename BOM>class stdair::FacCloneBom< BOM >
```

Base class for Factory layer.

Definition at line 22 of file [FacCloneBom.hpp](#).

#### 32.66.2 Constructor & Destructor Documentation

32.66.2.1 `template<typename BOM > stdair::FacCloneBom< BOM >::FacCloneBom ( ) [inline], [protected]`

Default Constructor.

Definition at line 48 of file [FacCloneBom.hpp](#).

32.66.2.2 `template<typename BOM > stdair::FacCloneBom< BOM >::~~FacCloneBom ( ) [inline]`

Destructor.

Definition at line 54 of file [FacCloneBom.hpp](#).

References [stdair::FacCloneBom< BOM >::clean\(\)](#).

## 32.66.3 Member Function Documentation

32.66.3.1 `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::FacSupervisor::instance\(\)](#), and [stdair::FacSupervisor::registerCloneBomFactory\(\)](#).

32.66.3.2 `template<typename BOM > BOM & stdair::FacCloneBom< BOM >::clone ( const BOM & iBom )`

Clone a BOM object.

Definition at line 110 of file [FacCloneBom.hpp](#).

32.66.3.3 `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\(\)](#).

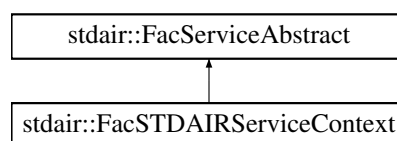
The documentation for this class was generated from the following file:

- [stdair/factory/FacCloneBom.hpp](#)

## 32.67 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](#)

### 32.67.1 Detailed Description

Base class for the (Service) Factory layer.

Definition at line 16 of file [FacServiceAbstract.hpp](#).

### 32.67.2 Member Typedef Documentation

#### 32.67.2.1 `typedef std::vector<ServiceAbstract*> stdair::FacServiceAbstract::ServicePool_T`

Define the list (pool) of Service objects.

Definition at line 20 of file [FacServiceAbstract.hpp](#).

### 32.67.3 Constructor & Destructor Documentation

#### 32.67.3.1 `stdair::FacServiceAbstract::~~FacServiceAbstract ( ) [virtual]`

Destructor.

Definition at line 13 of file [FacServiceAbstract.cpp](#).

References [clean\(\)](#).

#### 32.67.3.2 `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](#).

### 32.67.4 Member Function Documentation

#### 32.67.4.1 `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\(\)](#).

### 32.67.5 Member Data Documentation

#### 32.67.5.1 `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:

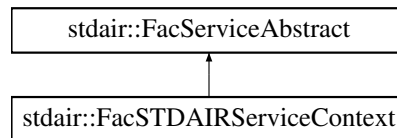
- [stdair/service/FacServiceAbstract.hpp](#)
- [stdair/service/FacServiceAbstract.cpp](#)

## 32.68 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

#### 32.68.1 Detailed Description

Factory for [Bucket](#).

Definition at line 18 of file [FacSTDAIRServiceContext.hpp](#).

#### 32.68.2 Member Typedef Documentation

32.68.2.1 typedef std::vector<[ServiceAbstract](#)\*> stdair::FacServiceAbstract::ServicePool\_T [inherited]

Define the list (pool) of Service objects.

Definition at line 20 of file [FacServiceAbstract.hpp](#).

### 32.68.3 Constructor & Destructor Documentation

#### 32.68.3.1 `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](#).

#### 32.68.3.2 `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\(\)](#).

### 32.68.4 Member Function Documentation

#### 32.68.4.1 `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\(\)](#).

#### 32.68.4.2 `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](#).

#### 32.68.4.3 `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\(\)](#).

### 32.68.5 Member Data Documentation

#### 32.68.5.1 `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](#)

## 32.69 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](#) &)

#### 32.69.1 Detailed Description

Singleton class to register and clean all Factories.

Definition at line 20 of file [FacSupervisor.hpp](#).

#### 32.69.2 Member Typedef Documentation

##### 32.69.2.1 typedef std::list<[FacAbstract](#)\*> stdair::FacSupervisor::PersistentBomFactoryPool\_T

Define the pool (list) of factories.

Definition at line 25 of file [FacSupervisor.hpp](#).



32.69.2.2 `typedef std::list<FacAbstract*> stdair::FacSupervisor::CloneBomFactoryPool_T`

Definition at line 26 of file [FacSupervisor.hpp](#).

32.69.2.3 `typedef std::list<FacServiceAbstract*> stdair::FacSupervisor::ServiceFactoryPool_T`

Definition at line 27 of file [FacSupervisor.hpp](#).

### 32.69.3 Constructor & Destructor Documentation

32.69.3.1 `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\(\)](#).

32.69.3.2 `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\(\)](#).

32.69.3.3 `stdair::FacSupervisor::FacSupervisor ( const FacSupervisor & ) [inline],[protected]`

Definition at line 121 of file [FacSupervisor.hpp](#).

### 32.69.4 Member Function Documentation

32.69.4.1 `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::FacBom< BOM >::instance\(\)](#), and [stdair::FacCloneBom< BOM >::instance\(\)](#).

32.69.4.2 `void stdair::FacSupervisor::registerPersistentBomFactory ( FacAbstract * ioFac_ptr )`

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

<i>FacAbstract*</i>	The concrete Factory to register.
---------------------	-----------------------------------

Definition at line 34 of file [FacSupervisor.cpp](#).

Referenced by [stdair::FacBom< BOM >::instance\(\)](#).

#### 32.69.4.3 void stdair::FacSupervisor::registerCloneBomFactory ( *FacAbstract \* ioFac\_ptr* )

Register a newly instantiated concrete factory for the Bom layer.

When a concrete Factory is firstly instantiated this factory have to register itself to the [FacSupervisor](#)

Parameters

<i>FacAbstract*</i>	The concrete Factory to register.
---------------------	-----------------------------------

Definition at line 39 of file [FacSupervisor.cpp](#).

Referenced by [stdair::FacCloneBom< BOM >::instance\(\)](#).

#### 32.69.4.4 void stdair::FacSupervisor::registerServiceFactory ( *FacServiceAbstract \* ioFac\_ptr* )

Register a newly instantiated concrete factory for the Service layer.

When a concrete Factory is firstly instantiated this factory have to register itself to the [FacSupervisor](#).

Parameters

<i>FacService↔ Abstract&amp;</i>	the concrete Factory to register.
--------------------------------------	-----------------------------------

Definition at line 44 of file [FacSupervisor.cpp](#).

Referenced by [stdair::FacSTDAIRServiceContext::instance\(\)](#).

#### 32.69.4.5 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\(\)](#).

#### 32.69.4.6 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\(\)](#).

#### 32.69.4.7 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\(\)](#).

#### 32.69.4.8 void stdair::FacSupervisor::cleanLoggerService ( ) [static]

Delete the static instance of the [Logger](#) object.

Definition at line 90 of file [FacSupervisor.cpp](#).

Referenced by [cleanAll\(\)](#).

**32.69.4.9** `void stdair::FacSupervisor::cleanDBSessionManager ( ) [static]`

Delete the static instance of the [DBSessionManager](#) object.

Definition at line 96 of file [FacSupervisor.cpp](#).

Referenced by [cleanAll\(\)](#).

**32.69.4.10** `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:

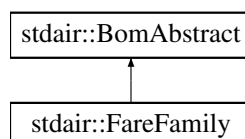
- [stdair/service/FacSupervisor.hpp](#)
- [stdair/service/FacSupervisor.cpp](#)

## 32.70 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

- template<typename BOM >  
class [FacBom](#)
- template<typename BOM >  
class [FacCloneBom](#)
- class [FacBomManager](#)
- class [boost::serialization::access](#)

#### 32.70.1 Detailed Description

Class representing the actual attributes for a family fare.

Definition at line 28 of file [FareFamily.hpp](#).

#### 32.70.2 Member Typedef Documentation

##### 32.70.2.1 typedef FareFamilyKey stdair::FareFamily::Key\_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 39 of file [FareFamily.hpp](#).

#### 32.70.3 Constructor & Destructor Documentation

##### 32.70.3.1 stdair::FareFamily::FareFamily ( const Key\_T & iKey ) [protected]

Constructor.

Definition at line 32 of file [FareFamily.cpp](#).

**32.70.3.2** `stdair::FareFamily::~~FareFamily ( )` `[protected],[virtual]`

Destructor.

Definition at line 36 of file [FareFamily.cpp](#).

#### 32.70.4 Member Function Documentation

**32.70.4.1** `const Key_T& stdair::FareFamily::getKey ( ) const` `[inline]`

Get the family fare key.

Definition at line 45 of file [FareFamily.hpp](#).

References [\\_key](#).

**32.70.4.2** `BomAbstract* const stdair::FareFamily::getParent ( ) const` `[inline]`

Get the parent object.

Definition at line 50 of file [FareFamily.hpp](#).

References [\\_parent](#).

**32.70.4.3** `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\(\)](#).

**32.70.4.4** `const HolderMap_T& stdair::FareFamily::getHolderMap ( ) const` `[inline]`

Get the map of children holders.

Definition at line 60 of file [FareFamily.hpp](#).

References [\\_holderMap](#).

**32.70.4.5** `const FRAT5Curve_T& stdair::FareFamily::getFrat5Curve ( ) const` `[inline]`

Get the FRAT5 Curve.

Definition at line 65 of file [FareFamily.hpp](#).

References [\\_frat5Curve](#).

**32.70.4.6** `const FFDisutilityCurve_T& stdair::FareFamily::getDisutilityCurve ( ) const` `[inline]`

Get the Disutility Curve.

Definition at line 70 of file [FareFamily.hpp](#).

References [\\_disutilityCurve](#).

**32.70.4.7** `const MeanValue_T& stdair::FareFamily::getMean ( ) const` `[inline]`

Demand distribution.

Definition at line 75 of file [FareFamily.hpp](#).

References [\\_mean](#).

**32.70.4.8** `const StdDevValue_T& stdair::FareFamily::getStdDev ( ) const` `[inline]`

Definition at line 76 of file [FareFamily.hpp](#).

References [\\_stdDev](#).

32.70.4.9 `const MeanStdDevPairVector_T& stdair::FareFamily::getMeanStdDev ( ) const` `[inline]`

Demand distribution.

Definition at line 79 of file [FareFamily.hpp](#).

References [\\_meanStdDev](#).

32.70.4.10 `void stdair::FareFamily::setFrat5Curve ( const FRAT5Curve_T & iFRAT5Curve )` `[inline]`

FRAT5 Curve.

Definition at line 85 of file [FareFamily.hpp](#).

References [\\_frat5Curve](#).

32.70.4.11 `void stdair::FareFamily::setDisutilityCurve ( const FFDisutilityCurve_T & iDisutilityCurve )` `[inline]`

Disutility Curve.

Definition at line 90 of file [FareFamily.hpp](#).

References [\\_disutilityCurve](#).

32.70.4.12 `void stdair::FareFamily::setMean ( const MeanValue_T & iMean )` `[inline]`

Demand distribution.

Definition at line 95 of file [FareFamily.hpp](#).

References [\\_mean](#).

32.70.4.13 `void stdair::FareFamily::setStdDev ( const StdDevValue_T & iStdDev )` `[inline]`

Definition at line 96 of file [FareFamily.hpp](#).

References [\\_stdDev](#).

32.70.4.14 `void stdair::FareFamily::setMeanStdDev ( const MeanStdDevPairVector_T & iMeanStdDev )` `[inline]`

Demand distribution.

Definition at line 99 of file [FareFamily.hpp](#).

References [\\_meanStdDev](#).

32.70.4.15 `void stdair::FareFamily::toStream ( std::ostream & ioOut ) const` `[inline], [virtual]`

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Implements [stdair::BomAbstract](#).

Definition at line 111 of file [FareFamily.hpp](#).

References [toString\(\)](#).

32.70.4.16 `void stdair::FareFamily::fromStream ( std::istream & iIn )` `[inline], [virtual]`

Read a Business Object from an input stream.

## Parameters

<i>istream</i> &	the input stream.
------------------	-------------------

Implements [stdair::BomAbstract](#).

Definition at line 120 of file [FareFamily.hpp](#).

**32.70.4.17** `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\(\)](#).

**32.70.4.18** `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\(\)](#).

**32.70.4.19** `template<class Archive > void stdair::FareFamily::serialize ( Archive & ar, const unsigned int iFileVersion )`

Serialisation.

Definition at line 62 of file [FareFamily.cpp](#).

References [\\_key](#).

### 32.70.5 Friends And Related Function Documentation

**32.70.5.1** `template<typename BOM > friend class FacBom` `[friend]`

Definition at line 29 of file [FareFamily.hpp](#).

**32.70.5.2** `template<typename BOM > friend class FacCloneBom` `[friend]`

Definition at line 30 of file [FareFamily.hpp](#).

**32.70.5.3** `friend class FacBomManager` `[friend]`

Definition at line 31 of file [FareFamily.hpp](#).

**32.70.5.4** `friend class boost::serialization::access` `[friend]`

Definition at line 32 of file [FareFamily.hpp](#).

### 32.70.6 Member Data Documentation

**32.70.6.1** `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\(\)](#).

**32.70.6.2 BomAbstract\* stdair::FareFamily::\_parent**

Pointer on the parent class ([SegmentCabin](#)).

Definition at line 189 of file [FareFamily.hpp](#).

Referenced by [getParent\(\)](#).

**32.70.6.3 HolderMap\_T stdair::FareFamily::\_holderMap**

Map holding the children ([BookingClass](#) objects).

Definition at line 194 of file [FareFamily.hpp](#).

Referenced by [getHolderMap\(\)](#).

**32.70.6.4 FRAT5Curve\_T stdair::FareFamily::\_frat5Curve**

The associated FRAT5 curve.

Definition at line 199 of file [FareFamily.hpp](#).

Referenced by [getFrat5Curve\(\)](#), and [setFrat5Curve\(\)](#).

**32.70.6.5 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\(\)](#).

**32.70.6.6 MeanValue\_T stdair::FareFamily::\_mean**

Demand distribution forecast.

Definition at line 207 of file [FareFamily.hpp](#).

Referenced by [getMean\(\)](#), and [setMean\(\)](#).

**32.70.6.7 StdDevValue\_T stdair::FareFamily::\_stdDev**

Definition at line 208 of file [FareFamily.hpp](#).

Referenced by [getStdDev\(\)](#), and [setStdDev\(\)](#).

**32.70.6.8 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](#)

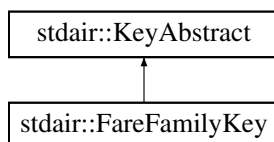
**32.71 stdair::FareFamilyKey Struct Reference**

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](#)

### 32.71.1 Detailed Description

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

Definition at line 26 of file [FareFamilyKey.hpp](#).

### 32.71.2 Constructor & Destructor Documentation

#### 32.71.2.1 stdair::FareFamilyKey::FareFamilyKey ( const [FamilyCode\\_T](#) & *iFamilyCode* )

Constructor.

Definition at line 28 of file [FareFamilyKey.cpp](#).

#### 32.71.2.2 stdair::FareFamilyKey::FareFamilyKey ( const [FareFamilyKey](#) & *iFareFamilyKey* )

Copy constructor.

Definition at line 23 of file [FareFamilyKey.cpp](#).

#### 32.71.2.3 stdair::FareFamilyKey::~~FareFamilyKey ( )

Destructor.

Definition at line 33 of file [FareFamilyKey.cpp](#).

### 32.71.3 Member Function Documentation

#### 32.71.3.1 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\(\)](#).

32.71.3.2 void stdair::FareFamilyKey::toStream ( std::ostream & *ioOut* ) const [virtual]

Dump a Business Object Key into an output stream.

## Parameters

<i>ostream</i> &	the output stream.
------------------	--------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 37 of file [FareFamilyKey.cpp](#).

References [toString\(\)](#).

**32.71.3.3** void stdair::FareFamilyKey::fromStream ( std::istream & *ioIn* ) [virtual]

Read a Business Object Key from an input stream.

## Parameters

<i>istream</i> &	the input stream.
------------------	-------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 42 of file [FareFamilyKey.cpp](#).

**32.71.3.4** 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\(\)](#).

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

Serialisation.

Definition at line 68 of file [FareFamilyKey.cpp](#).

## 32.71.4 Friends And Related Function Documentation

**32.71.4.1** 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:

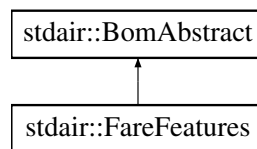
- [stdair/bom/FareFamilyKey.hpp](#)
- [stdair/bom/FareFamilyKey.cpp](#)

## 32.72 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](#) &iFlight←→DateTime) const

### Protected Member Functions

- [FareFeatures](#) (const [Key\\_T](#) &)
- virtual [~FareFeatures](#) ()

### Protected Attributes

- [Key\\_T](#) [\\_key](#)
- [BomAbstract](#) \* [\\_parent](#)
- [HolderMap\\_T](#) [\\_holderMap](#)

### Friends

- template<typename BOM >  
class [FacBom](#)
- template<typename BOM >  
class [FacCloneBom](#)
- class [FacBomManager](#)

### 32.72.1 Detailed Description

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

Definition at line 18 of file [FareFeatures.hpp](#).

### 32.72.2 Member Typedef Documentation

#### 32.72.2.1 typedef FareFeaturesKey stdair::FareFeatures::Key\_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 28 of file [FareFeatures.hpp](#).

### 32.72.3 Constructor & Destructor Documentation

#### 32.72.3.1 stdair::FareFeatures::FareFeatures ( const Key\_T & iKey ) [protected]

Main constructor.

Definition at line 33 of file [FareFeatures.cpp](#).

#### 32.72.3.2 stdair::FareFeatures::~~FareFeatures ( ) [protected],[virtual]

Destructor.

Definition at line 38 of file [FareFeatures.cpp](#).

### 32.72.4 Member Function Documentation

#### 32.72.4.1 void stdair::FareFeatures::toStream ( std::ostream & ioOut ) const [inline],[virtual]

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Implements [stdair::BomAbstract](#).

Definition at line 37 of file [FareFeatures.hpp](#).

References [toString\(\)](#).

#### 32.72.4.2 void stdair::FareFeatures::fromStream ( std::istream & ioIn ) [inline],[virtual]

Read a Business Object from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Implements [stdair::BomAbstract](#).

Definition at line 46 of file [FareFeatures.hpp](#).

#### 32.72.4.3 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\(\)](#).

#### 32.72.4.4 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\(\)](#).

**32.72.4.5** `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](#).

**32.72.4.6** `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](#).

**32.72.4.7** `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](#).

**32.72.4.8** `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\(\)](#).

**32.72.4.9** `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\(\)](#).

**32.72.4.10** `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\(\)](#).

**32.72.4.11** `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\(\)](#).

**32.72.4.12** `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\(\)](#).

**32.72.4.13** `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\(\)](#).

**32.72.4.14** `bool stdair::FareFeatures::isTripTypeValid ( const TripType_T & iBookingRequestTripType ) const`

Check whether the fare rule trip type corresponds to the booking request trip type.

Definition at line 50 of file [FareFeatures.cpp](#).

References [getTripType\(\)](#), [stdair::TRIP\\_TYPE\\_INBOUND](#), [stdair::TRIP\\_TYPE\\_OUTBOUND](#), and [stdair::TRIP\\_TYPE\\_ROUND\\_TRIP](#).

**32.72.4.15** `bool stdair::FareFeatures::isStayDurationValid ( const DayDuration_T & iStayDuration ) const`

Check whether a given stay duration is greater or equal to the minimum stay of the fare rule.

Definition at line 75 of file [FareFeatures.cpp](#).

References [getMinimumStay\(\)](#).

**32.72.4.16** `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\(\)](#).

## 32.72.5 Friends And Related Function Documentation

**32.72.5.1** `template<typename BOM> friend class FacBom` `[friend]`

Definition at line 19 of file [FareFeatures.hpp](#).

**32.72.5.2** `template<typename BOM> friend class FacCloneBom` `[friend]`

Definition at line 20 of file [FareFeatures.hpp](#).

**32.72.5.3** `friend class FacBomManager` `[friend]`

Definition at line 21 of file [FareFeatures.hpp](#).

## 32.72.6 Member Data Documentation

**32.72.6.1** `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\(\)](#).

**32.72.6.2 BomAbstract\*** stdair::FareFeatures::\_parent [protected]

Pointer on the parent class.

Definition at line 181 of file [FareFeatures.hpp](#).

Referenced by [getParent\(\)](#).

**32.72.6.3 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:

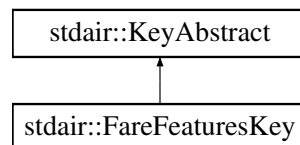
- [stdair/bom/FareFeatures.hpp](#)
- [stdair/bom/FareFeatures.cpp](#)

**32.73 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

**32.73.1 Detailed Description**

Key of date-period.

Definition at line 18 of file [FareFeaturesKey.hpp](#).



### 32.73.2 Constructor & Destructor Documentation

**32.73.2.1** `stdair::FareFeaturesKey::FareFeaturesKey ( const TripType_T & iTripType, const DayDuration_T & iAdvancePurchase, const SaturdayStay_T & iSaturdayStay, const ChangeFees_T & iChangeFees, const NonRefundable_T & iNonRefundable, const DayDuration_T & iMinimumStay )`

Main constructor.

Definition at line 26 of file [FareFeaturesKey.cpp](#).

**32.73.2.2** `stdair::FareFeaturesKey::FareFeaturesKey ( const FareFeaturesKey & iKey )`

Copy constructor.

Definition at line 38 of file [FareFeaturesKey.cpp](#).

**32.73.2.3** `stdair::FareFeaturesKey::~~FareFeaturesKey ( )`

Destructor.

Definition at line 48 of file [FareFeaturesKey.cpp](#).

### 32.73.3 Member Function Documentation

**32.73.3.1** `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\(\)](#).

**32.73.3.2** `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\(\)](#).

**32.73.3.3** `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\(\)](#).

**32.73.3.4** `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\(\)](#).

**32.73.3.5** `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\(\)](#).

32.73.3.6 `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\(\)](#).

32.73.3.7 `void stdair::FareFeaturesKey::toStream ( std::ostream & ioOut ) const` `[virtual]`

Dump a Business Object Key into an output stream.

Parameters

<code>ostream&amp;</code>	the output stream.
---------------------------	--------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 52 of file [FareFeaturesKey.cpp](#).

References [toString\(\)](#).

32.73.3.8 `void stdair::FareFeaturesKey::fromStream ( std::istream & ioIn )` `[virtual]`

Read a Business Object Key from an input stream.

Parameters

<code>istream&amp;</code>	the input stream.
---------------------------	-------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 57 of file [FareFeaturesKey.cpp](#).

32.73.3.9 `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:

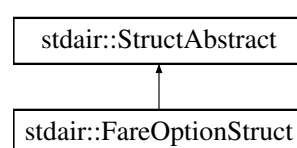
- [stdair/bom/FareFeaturesKey.hpp](#)
- [stdair/bom/FareFeaturesKey.cpp](#)

## 32.74 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](#) ()

## 32.74.1 Detailed Description

Structure holding the elements of a fare option.

Definition at line 20 of file [FareOptionStruct.hpp](#).

## 32.74.2 Constructor &amp; Destructor Documentation

## 32.74.2.1 stdair::FareOptionStruct::FareOptionStruct ( )

Default constructor.

Definition at line 14 of file [FareOptionStruct.cpp](#).

32.74.2.2 stdair::FareOptionStruct::FareOptionStruct ( const std::string & iClassPath, const [Fare\\_T](#) & iFare, const [ChangeFees\\_T](#) & iChangeFee, const [NonRefundable\\_T](#) & iNonRefundable, const [SaturdayStay\\_T](#) & iSaturdayNightStay )

Main constructor.

Definition at line 26 of file [FareOptionStruct.cpp](#).

32.74.2.3 stdair::FareOptionStruct::FareOptionStruct ( const [FareOptionStruct](#) & iFO )

Copy constructor.

Definition at line 19 of file [FareOptionStruct.cpp](#).

## 32.74.2.4 stdair::FareOptionStruct::~~FareOptionStruct ( )

Destructor.

Definition at line 38 of file [FareOptionStruct.cpp](#).

### 32.74.3 Member Function Documentation

**32.74.3.1** `const ClassList_StringList_T& stdair::FareOptionStruct::getClassPath ( ) const` `[inline]`

Get the class-path.

Definition at line 24 of file [FareOptionStruct.hpp](#).

**32.74.3.2** `const Fare_T& stdair::FareOptionStruct::getFare ( ) const` `[inline]`

Get the fare value.

Definition at line 29 of file [FareOptionStruct.hpp](#).

**32.74.3.3** `const Availability_T& stdair::FareOptionStruct::getAvailability ( ) const` `[inline]`

Get the availability.

Definition at line 34 of file [FareOptionStruct.hpp](#).

**32.74.3.4** `const ChangeFees_T stdair::FareOptionStruct::getChangeFees ( ) const` `[inline]`

Get the change fees.

Definition at line 39 of file [FareOptionStruct.hpp](#).

**32.74.3.5** `const NonRefundable_T stdair::FareOptionStruct::getNonRefundable ( ) const` `[inline]`

State whether the ticket is refundable.

Definition at line 44 of file [FareOptionStruct.hpp](#).

**32.74.3.6** `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](#).

**32.74.3.7** `void stdair::FareOptionStruct::addClassList ( const std::string iClassCodeList )`

Set the class-path.

Definition at line 93 of file [FareOptionStruct.cpp](#).

**32.74.3.8** `void stdair::FareOptionStruct::emptyClassList ( )`

Empty the class-path.

Definition at line 98 of file [FareOptionStruct.cpp](#).

**32.74.3.9** `void stdair::FareOptionStruct::setFare ( const Fare_T & iFare )` `[inline]`

Set the fare value.

Definition at line 63 of file [FareOptionStruct.hpp](#).

**32.74.3.10** `void stdair::FareOptionStruct::setAvailability ( const Availability_T & iAv )` `[inline]`

Set the availability.

Definition at line 68 of file [FareOptionStruct.hpp](#).

**32.74.3.11** `void stdair::FareOptionStruct::setChangeFees ( const ChangeFees_T iRes )` `[inline]`

Set the change fees.

Definition at line 73 of file [FareOptionStruct.hpp](#).

**32.74.3.12** `void stdair::FareOptionStruct::setNonRefundable ( const NonRefundable_T iRes ) [inline]`

Set the flag for the ticket refundability.

Definition at line 78 of file [FareOptionStruct.hpp](#).

**32.74.3.13** `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](#).

**32.74.3.14** `void stdair::FareOptionStruct::toStream ( std::ostream & ioOut ) const`

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Definition at line 42 of file [FareOptionStruct.cpp](#).

References [describe\(\)](#).

**32.74.3.15** `void stdair::FareOptionStruct::fromStream ( std::istream & ioin ) [virtual]`

Read a Business Object from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented from [stdair::StructAbstract](#).

Definition at line 47 of file [FareOptionStruct.cpp](#).

**32.74.3.16** `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\(\)](#).

**32.74.3.17** `const std::string stdair::FareOptionStruct::display ( ) const`

Display of the structure.

Definition at line 73 of file [FareOptionStruct.cpp](#).

Referenced by [stdair::TravelSolutionStruct::display\(\)](#).

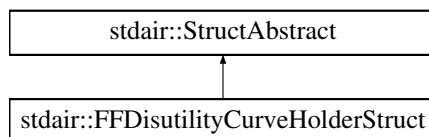
The documentation for this struct was generated from the following files:

- [stdair/bom/FareOptionStruct.hpp](#)
- [stdair/bom/FareOptionStruct.cpp](#)

## 32.75 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](#) ()

### 32.75.1 Detailed Description

Structure holding the elements of a snapshot .

Definition at line 19 of file [FFDisutilityCurveHolderStruct.hpp](#).

### 32.75.2 Constructor & Destructor Documentation

#### 32.75.2.1 stdair::FFDisutilityCurveHolderStruct::FFDisutilityCurveHolderStruct ( )

Constructor.

Definition at line 14 of file [FFDisutilityCurveHolderStruct.cpp](#).

#### 32.75.2.2 stdair::FFDisutilityCurveHolderStruct::FFDisutilityCurveHolderStruct ( const [FFDisutilityCurveHolderStruct](#) & *iHolder* )

Copy constructor.

Definition at line 19 of file [FFDisutilityCurveHolderStruct.cpp](#).

#### 32.75.2.3 stdair::FFDisutilityCurveHolderStruct::~~FFDisutilityCurveHolderStruct ( )

Destructor.

Definition at line 24 of file [FFDisutilityCurveHolderStruct.cpp](#).

### 32.75.3 Member Function Documentation

#### 32.75.3.1 const [FFDisutilityCurve\\_T](#) & stdair::FFDisutilityCurveHolderStruct::getFFDisutilityCurve ( const std::string & *iKey* ) const

Get the [FFDisutilityCurve](#) corresponding to the given key.

Definition at line 29 of file [FFDisutilityCurveHolderStruct.cpp](#).

References [STDAIR\\_LOG\\_DEBUG](#).

Referenced by [stdair::BomRoot::getFFDisutilityCurve\(\)](#).

**32.75.3.2** void stdair::FFDisutilityCurveHolderStruct::addCurve ( const std::string & *iKey*, const FFDisutilityCurve\_T & *iCurve* )

Add a new curve to the holder.

Definition at line 42 of file [FFDisutilityCurveHolderStruct.cpp](#).

References [STDAIR\\_LOG\\_DEBUG](#).

Referenced by [stdair::BomRoot::addFFDisutilityCurve\(\)](#).

**32.75.3.3** void stdair::FFDisutilityCurveHolderStruct::toStream ( std::ostream & *ioOut* ) const

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Definition at line 53 of file [FFDisutilityCurveHolderStruct.cpp](#).

References [describe\(\)](#).

**32.75.3.4** void stdair::FFDisutilityCurveHolderStruct::fromStream ( std::istream & *ioIn* ) [virtual]

Read a Business Object from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented from [stdair::StructAbstract](#).

Definition at line 58 of file [FFDisutilityCurveHolderStruct.cpp](#).

**32.75.3.5** 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\(\)](#).

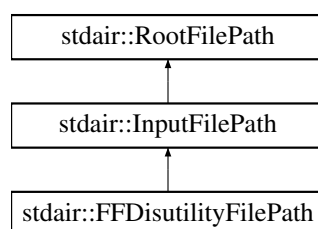
The documentation for this struct was generated from the following files:

- [stdair/bom/FFDisutilityCurveHolderStruct.hpp](#)
- [stdair/bom/FFDisutilityCurveHolderStruct.cpp](#)

## 32.76 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](#)

## 32.76.1 Detailed Description

FFDisutility input file.

Definition at line 100 of file [stdair\\_file.hpp](#).

## 32.76.2 Constructor &amp; Destructor Documentation

32.76.2.1 `stdair::FFDisutilityFilePath::FFDisutilityFilePath ( const Filename\_T &iFilename ) [inline], [explicit]`

Constructor.

Definition at line 105 of file [stdair\\_file.hpp](#).

## 32.76.3 Member Function Documentation

32.76.3.1 `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\(\)](#).

## 32.76.4 Member Data Documentation

32.76.4.1 `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:

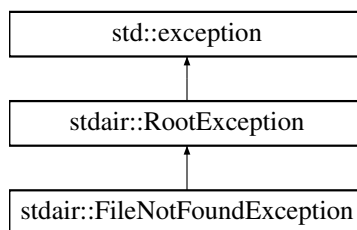
- [stdair/stdair\\_file.hpp](#)

## 32.77 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](#)

#### 32.77.1 Detailed Description

File not found.

Definition at line 50 of file [stdair\\_exceptions.hpp](#).

#### 32.77.2 Constructor & Destructor Documentation

**32.77.2.1** `stdair::FileNotFoundException::FileNotFoundException ( const std::string & iWhat ) [inline]`

Constructor.

Definition at line 53 of file [stdair\\_exceptions.hpp](#).

#### 32.77.3 Member Function Documentation

**32.77.3.1** `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\(\)](#).

#### 32.77.4 Member Data Documentation

**32.77.4.1** `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:

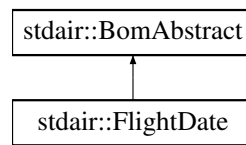
- [stdair/stdair\\_exceptions.hpp](#)

## 32.78 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

- template<typename BOM >  
class [FacBom](#)
- template<typename BOM >  
class [FacCloneBom](#)
- class [FacBomManager](#)
- class [boost::serialization::access](#)

### 32.78.1 Detailed Description

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

Definition at line 35 of file [FlightDate.hpp](#).

### 32.78.2 Member Typedef Documentation

#### 32.78.2.1 `typedef FlightDateKey stdair::FlightDate::Key_T`

Definition allowing to retrieve the associated BOM key type.

Definition at line 46 of file [FlightDate.hpp](#).

### 32.78.3 Constructor & Destructor Documentation

#### 32.78.3.1 `stdair::FlightDate::FlightDate ( const Key_T & iKey )` `[protected]`

Main constructor.

Definition at line 29 of file [FlightDate.cpp](#).

#### 32.78.3.2 `stdair::FlightDate::~~FlightDate ( )` `[protected]`, `[virtual]`

Destructor.

Definition at line 33 of file [FlightDate.cpp](#).

### 32.78.4 Member Function Documentation

#### 32.78.4.1 `const Key_T& stdair::FlightDate::getKey ( ) const` `[inline]`

Get the flight-date key.

Definition at line 52 of file [FlightDate.hpp](#).

References [\\_key](#).

#### 32.78.4.2 `BomAbstract* const stdair::FlightDate::getParent ( ) const` `[inline]`

Get the parent object.

Definition at line 57 of file [FlightDate.hpp](#).

References [\\_parent](#).

Referenced by [getAirlineCode\(\)](#).

#### 32.78.4.3 `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::jsonExportFlightDateList\(\)](#), and [stdair::BomJSONExport::jsonExportFlightDateObjects\(\)](#).

#### 32.78.4.4 `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\(\)](#), [stdair::BomJSONExport::jsonExportFlightDateList\(\)](#), and [stdair::BomJSONExport::jsonExportFlightDateObjects\(\)](#).

#### 32.78.4.5 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\(\)](#).

#### 32.78.4.6 const HolderMap\_T& stdair::FlightDate::getHolderMap ( ) const [inline]

Get the map of children holders.

Definition at line 83 of file [FlightDate.hpp](#).

References [\\_holderMap](#).

#### 32.78.4.7 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

<i>const</i>	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\(\)](#), [stdair::BomRetriever::retrieveDummyLegCabin\(\)](#), and [stdair::BomRetriever::retrieveOperatingLegDateFromLongKey\(\)](#).

#### 32.78.4.8 LegDate \* stdair::FlightDate::getLegDate ( const LegDateKey & iLegDateKey ) const

Get a pointer on the [LegDate](#) object corresponding to the given key.

##### Note

The [LegDate](#) object can be inherited from, if needed. In that case, a `dynamic_cast<>` may be needed.

##### Parameters

<i>const</i>	<a href="#">LegDateKey</a> & 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\(\)](#).

**32.78.4.9** `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**

<i>const</i>	<code>std::string&amp;</code> 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::retrieveDummySegmentCabin\(\)](#), [stdair::BomRetriever::retrieveSegmentDateFromKey\(\)](#), and [stdair::BomRetriever::retrieveSegmentDateFromLongKey\(\)](#).

**32.78.4.10** `SegmentDate * stdair::FlightDate::getSegmentDate ( const SegmentDateKey & iSegmentDateKey ) const`

Get a pointer on the [SegmentDate](#) object corresponding to the given key.

**Note**

The [SegmentDate](#) object can be inherited from, if needed. In that case, a `dynamic_cast<>` may be needed.

**Parameters**

<i>const</i>	<a href="#">SegmentDateKey</a> & 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\(\)](#).

**32.78.4.11** `void stdair::FlightDate::toStream ( std::ostream & ioOut ) const [inline],[virtual]`

Dump a Business Object into an output stream.

**Parameters**

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Implements [stdair::BomAbstract](#).

Definition at line 142 of file [FlightDate.hpp](#).

References [toString\(\)](#).

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

Read a Business Object from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Implements [stdair::BomAbstract](#).

Definition at line 151 of file [FlightDate.hpp](#).

**32.78.4.13** 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\(\)](#).

**32.78.4.14** 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\(\)](#).

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

Serialisation.

Definition at line 187 of file [CmdBomSerialiser.cpp](#).

References [\\_key](#).

## 32.78.5 Friends And Related Function Documentation

**32.78.5.1** template<typename BOM > friend class FacBom [friend]

Definition at line 36 of file [FlightDate.hpp](#).

**32.78.5.2** template<typename BOM > friend class FacCloneBom [friend]

Definition at line 37 of file [FlightDate.hpp](#).

**32.78.5.3** friend class FacBomManager [friend]

Definition at line 38 of file [FlightDate.hpp](#).

**32.78.5.4** friend class boost::serialization::access [friend]

Definition at line 39 of file [FlightDate.hpp](#).

### 32.78.6 Member Data Documentation

#### 32.78.6.1 `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\(\)](#).

#### 32.78.6.2 `BomAbstract* stdair::FlightDate::_parent` [protected]

Pointer on the parent class ([Inventory](#)).

Definition at line 221 of file [FlightDate.hpp](#).

Referenced by [getParent\(\)](#).

#### 32.78.6.3 `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:

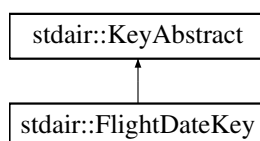
- [stdair/bom/FlightDate.hpp](#)
- [stdair/bom/FlightDate.cpp](#)
- [stdair/command/CmdBomSerialiser.cpp](#)

### 32.79 `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](#)

## 32.79.1 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](#).

## 32.79.2 Constructor &amp; Destructor Documentation

32.79.2.1 `stdair::FlightDateKey::FlightDateKey ( const FlightNumber_T & iFlightNumber, const Date_T & iFlightDate )`

Constructor.

Definition at line 28 of file [FlightDateKey.cpp](#).

32.79.2.2 `stdair::FlightDateKey::FlightDateKey ( const FlightDateKey & iKey )`

Copy constructor.

Definition at line 34 of file [FlightDateKey.cpp](#).

32.79.2.3 `stdair::FlightDateKey::~~FlightDateKey ( )`

Destructor.

Definition at line 39 of file [FlightDateKey.cpp](#).

## 32.79.3 Member Function Documentation

32.79.3.1 `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\(\)](#).

32.79.3.2 `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\(\)](#).

32.79.3.3 `void stdair::FlightDateKey::toStream ( std::ostream & ioOut ) const` `[virtual]`

Dump a Business Object Key into an output stream.

## Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 43 of file [FlightDateKey.cpp](#).

References [toString\(\)](#).



32.79.3.4 void stdair::FlightDateKey::fromStream ( std::istream & *ioln* ) [virtual]

Read a Business Object Key from an input stream.

## Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 48 of file [FlightDateKey.cpp](#).

**32.79.3.5** `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 [toString\(\)](#).

**32.79.3.6** `template<class Archive > void stdair::FlightDateKey::serialize ( Archive & ar, const unsigned int iFileVersion )`

Serialisation.

Definition at line 77 of file [FlightDateKey.cpp](#).

### 32.79.4 Friends And Related Function Documentation

**32.79.4.1** `friend class boost::serialization::access` [friend]

Definition at line 29 of file [FlightDateKey.hpp](#).

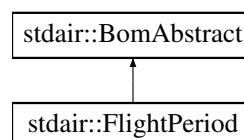
The documentation for this struct was generated from the following files:

- [stdair/bom/FlightDateKey.hpp](#)
- [stdair/bom/FlightDateKey.cpp](#)

## 32.80 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

- template<typename BOM >  
class [FacBom](#)
- template<typename BOM >  
class [FacCloneBom](#)
- class [FacBomManager](#)

### 32.80.1 Detailed Description

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

Definition at line 15 of file [FlightPeriod.hpp](#).

### 32.80.2 Member Typedef Documentation

#### 32.80.2.1 typedef [FlightPeriodKey](#) stdair::FlightPeriod::Key\_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 23 of file [FlightPeriod.hpp](#).

### 32.80.3 Constructor & Destructor Documentation

#### 32.80.3.1 stdair::FlightPeriod::FlightPeriod ( const [Key\\_T](#) & *iKey* ) [protected]

Main constructor.

Definition at line 12 of file [FlightPeriod.cpp](#).

#### 32.80.3.2 stdair::FlightPeriod::~FlightPeriod ( ) [protected]

Destructor.

Definition at line 22 of file [FlightPeriod.cpp](#).

## 32.80.4 Member Function Documentation

## 32.80.4.1 const Key\_T&amp; stdair::FlightPeriod::getKey ( ) const [inline]

Get the flight-period key.

Definition at line 28 of file [FlightPeriod.hpp](#).

References [\\_key](#).

## 32.80.4.2 BomAbstract\* const stdair::FlightPeriod::getParent ( ) const [inline]

Get the parent object.

Definition at line 31 of file [FlightPeriod.hpp](#).

References [\\_parent](#).

## 32.80.4.3 const FlightNumber\_T&amp; 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\(\)](#).

## 32.80.4.4 const PeriodStruct&amp; 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\(\)](#).

## 32.80.4.5 const HolderMap\_T&amp; stdair::FlightPeriod::getHolderMap ( ) const [inline]

Get the map of children holders.

Definition at line 42 of file [FlightPeriod.hpp](#).

References [\\_holderMap](#).

## 32.80.4.6 void stdair::FlightPeriod::toStream ( std::ostream &amp; ioOut ) const [inline], [virtual]

Dump a Business Object into an output stream.

## Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Implements [stdair::BomAbstract](#).

Definition at line 49 of file [FlightPeriod.hpp](#).

References [toString\(\)](#).

## 32.80.4.7 void stdair::FlightPeriod::fromStream ( std::istream &amp; iIn ) [inline], [virtual]

Read a Business Object from an input stream.

## Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Implements [stdair::BomAbstract](#).

Definition at line 53 of file [FlightPeriod.hpp](#).

**32.80.4.8** `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\(\)](#).

**32.80.4.9** `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\(\)](#).

## 32.80.5 Friends And Related Function Documentation

**32.80.5.1** `template<typename BOM > friend class FacBom` `[friend]`

Definition at line 16 of file [FlightPeriod.hpp](#).

**32.80.5.2** `template<typename BOM > friend class FacCloneBom` `[friend]`

Definition at line 17 of file [FlightPeriod.hpp](#).

**32.80.5.3** `friend class FacBomManager` `[friend]`

Definition at line 18 of file [FlightPeriod.hpp](#).

## 32.80.6 Member Data Documentation

**32.80.6.1** `Key_T stdair::FlightPeriod::_key` `[protected]`

Definition at line 86 of file [FlightPeriod.hpp](#).

Referenced by [describeKey\(\)](#), [getFlightNumber\(\)](#), [getKey\(\)](#), and [getPeriod\(\)](#).

**32.80.6.2** `BomAbstract* stdair::FlightPeriod::_parent` `[protected]`

Definition at line 87 of file [FlightPeriod.hpp](#).

Referenced by [getParent\(\)](#).

**32.80.6.3** `HolderMap_T stdair::FlightPeriod::_holderMap` `[protected]`

Definition at line 88 of file [FlightPeriod.hpp](#).

Referenced by [getHolderMap\(\)](#).

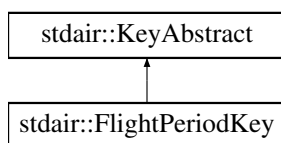
The documentation for this class was generated from the following files:

- [stdair/bom/FlightPeriod.hpp](#)
- [stdair/bom/FlightPeriod.cpp](#)

## 32.81 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

#### 32.81.1 Detailed Description

Key of flight-period.

Definition at line 13 of file [FlightPeriodKey.hpp](#).

#### 32.81.2 Constructor & Destructor Documentation

32.81.2.1 `stdair::FlightPeriodKey::FlightPeriodKey ( const FlightNumber_T & iFlightNumber, const PeriodStruct & iPeriod )`

Constructors.

Definition at line 10 of file [FlightPeriodKey.cpp](#).

32.81.2.2 `stdair::FlightPeriodKey::FlightPeriodKey ( const FlightPeriodKey & iKey )`

Definition at line 16 of file [FlightPeriodKey.cpp](#).

32.81.2.3 `stdair::FlightPeriodKey::~~FlightPeriodKey ( )`

Destructor.

Definition at line 21 of file [FlightPeriodKey.cpp](#).

#### 32.81.3 Member Function Documentation

32.81.3.1 `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\(\)](#).

**32.81.3.2** `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\(\)](#).

**32.81.3.3** `void stdair::FlightPeriodKey::toStream ( std::ostream & ioOut ) const` `[virtual]`

Dump a Business Object Key into an output stream.

Parameters

<code>ostream&amp;</code>	the output stream.
---------------------------	--------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 25 of file [FlightPeriodKey.cpp](#).

References [toString\(\)](#).

**32.81.3.4** `void stdair::FlightPeriodKey::fromStream ( std::istream & ioIn )` `[virtual]`

Read a Business Object Key from an input stream.

Parameters

<code>istream&amp;</code>	the input stream.
---------------------------	-------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 30 of file [FlightPeriodKey.cpp](#).

**32.81.3.5** `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](#)

## 32.82 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 [kMaxUlp](#) = 4

#### 32.82.1 Detailed Description

template<typename RawType>class FloatingPoint< RawType >

Definition at line 117 of file [float\\_utils\\_google.hpp](#).

#### 32.82.2 Member Typedef Documentation

32.82.2.1 template<typename RawType> typedef TypeWithSize<sizeof(RawType)>::UInt FloatingPoint< RawType >::Bits

Definition at line 121 of file [float\\_utils\\_google.hpp](#).

#### 32.82.3 Constructor & Destructor Documentation

32.82.3.1 template<typename RawType> FloatingPoint< RawType >::FloatingPoint ( const RawType & x )  
[inline], [explicit]

Definition at line 165 of file [float\\_utils\\_google.hpp](#).

#### 32.82.4 Member Function Documentation

32.82.4.1 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 >::bits\(\)](#).

Referenced by [FloatingPoint< RawType >::Infinity\(\)](#).



**32.82.4.2** `template<typename RawType> static RawType FloatingPoint< RawType >::Infinity ( ) [inline],  
[static]`

Definition at line 179 of file [float\\_utils\\_google.hpp](#).

References [FloatingPoint< RawType >::ReinterpretBits\(\)](#).

**32.82.4.3** `template<typename RawType> const Bits& FloatingPoint< RawType >::bits ( ) const [inline]`

Definition at line 186 of file [float\\_utils\\_google.hpp](#).

Referenced by [FloatingPoint< RawType >::ReinterpretBits\(\)](#).

**32.82.4.4** `template<typename RawType> Bits FloatingPoint< RawType >::exponent_bits ( ) const [inline]`

Definition at line 189 of file [float\\_utils\\_google.hpp](#).

Referenced by [FloatingPoint< RawType >::is\\_nan\(\)](#).

**32.82.4.5** `template<typename RawType> Bits FloatingPoint< RawType >::fraction_bits ( ) const [inline]`

Definition at line 192 of file [float\\_utils\\_google.hpp](#).

Referenced by [FloatingPoint< RawType >::is\\_nan\(\)](#).

**32.82.4.6** `template<typename RawType> Bits FloatingPoint< RawType >::sign_bit ( ) const [inline]`

Definition at line 195 of file [float\\_utils\\_google.hpp](#).

**32.82.4.7** `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\(\)](#), and [FloatingPoint< RawType >::fraction\\_bits\(\)](#).

Referenced by [FloatingPoint< RawType >::AlmostEquals\(\)](#).

**32.82.4.8** `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\(\)](#), and [FloatingPoint< RawType >::kMaxUlp](#).

## 32.82.5 Member Data Documentation

**32.82.5.1** `template<typename RawType> const size_t FloatingPoint< RawType >::kBitCount = 8*sizeof(RawType)  
[static]`

Definition at line 126 of file [float\\_utils\\_google.hpp](#).

**32.82.5.2** `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](#).

**32.82.5.3** `template<typename RawType> const size_t FloatingPoint< RawType >::kExponentBitCount = kBitCount - 1 -  
kFractionBitCount [static]`

Definition at line 133 of file [float\\_utils\\_google.hpp](#).

32.82.5.4 `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](#).

32.82.5.5 `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](#).

32.82.5.6 `template<typename RawType> const Bits FloatingPoint< RawType >::kExponentBitMask = ~(kSignBitMask | kFractionBitMask) [static]`

Definition at line 143 of file [float\\_utils\\_google.hpp](#).

32.82.5.7 `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\(\)](#).

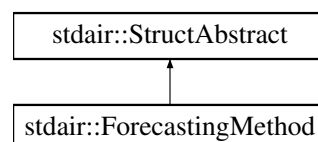
The documentation for this class was generated from the following file:

- [stdair/basic/float\\_utils\\_google.hpp](#)

## 32.83 stdair::ForecastingMethod Struct Reference

```
#include <stdair/basic/ForecastingMethod.hpp>
```

Inheritance diagram for stdair::ForecastingMethod:



### Public Types

- 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](#) ()

### 32.83.1 Detailed Description

Enumeration of forecasting methods.

Definition at line 15 of file [ForecastingMethod.hpp](#).

### 32.83.2 Member Enumeration Documentation

#### 32.83.2.1 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](#).

### 32.83.3 Constructor & Destructor Documentation

#### 32.83.3.1 stdair::ForecastingMethod::ForecastingMethod ( const [EN\\_ForecastingMethod](#) & *iForecastingMethod* )

Constructor.

Definition at line 37 of file [ForecastingMethod.cpp](#).

#### 32.83.3.2 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](#).

#### 32.83.3.3 stdair::ForecastingMethod::ForecastingMethod ( const [ForecastingMethod](#) & *iForecastingMethod* )

Default copy constructor.

Definition at line 31 of file [ForecastingMethod.cpp](#).

### 32.83.4 Member Function Documentation

#### 32.83.4.1 const std::string & stdair::ForecastingMethod::getLabel ( const [EN\\_ForecastingMethod](#) & *iMethod* ) [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](#).

**32.83.4.2** `char stdair::ForecastingMethod::getMethodLabel ( const EN_ForecastingMethod & iMethod ) [static]`

Get the label as a single char (e.g., 'Q', 'H', 'O', 'N' or 'B').

Definition at line 68 of file [ForecastingMethod.cpp](#).

**32.83.4.3** `std::string stdair::ForecastingMethod::getMethodLabelAsString ( const EN_ForecastingMethod & iMethod ) [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](#).

**32.83.4.4** `std::string stdair::ForecastingMethod::describeLabels ( ) [static]`

List the labels.

Definition at line 81 of file [ForecastingMethod.cpp](#).

References [LAST\\_VALUE](#).

Referenced by [ForecastingMethod\(\)](#).

**32.83.4.5** `ForecastingMethod::EN_ForecastingMethod stdair::ForecastingMethod::getMethod ( ) const`

Get the enumerated value.

Definition at line 93 of file [ForecastingMethod.cpp](#).

Referenced by [stdair::AirlineFeature::getForecastingMethod\(\)](#).

**32.83.4.6** `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](#).

**32.83.4.7** `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](#).

**32.83.4.8** `bool stdair::ForecastingMethod::operator== ( const EN_ForecastingMethod & iMethod ) const`

Comparison operator.

Definition at line 113 of file [ForecastingMethod.cpp](#).

**32.83.4.9** `void stdair::StructAbstract::toStream ( std::ostream & ioOut ) const [inline], [inherited]`

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Definition at line 29 of file [StructAbstract.hpp](#).

References [stdair::StructAbstract::describe\(\)](#).

**32.83.4.10** `virtual void stdair::StructAbstract::fromStream ( std::istream & ioIn ) [inline], [virtual], [inherited]`

Read a Business Object from an input stream.

## Parameters

<i>istream</i> &	the input stream.
------------------	-------------------

Reimplemented in [stdair::BookingRequestStruct](#), [stdair::EventStruct](#), [stdair::TravelSolutionStruct](#), [stdair::VirtualClassStruct](#), [stdair::ConfigHolderStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::ProgressStatusSet](#), [stdair::YieldRange](#), [stdair::CancellationStruct](#), [stdair::AirlineStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::FFDisutilityCurveHolderStruct](#), [stdair::FRAT5CurveHolderStruct](#), and [stdair::BreakPointStruct](#).

Definition at line 38 of file [StructAbstract.hpp](#).

Referenced by [operator>>\(\)](#).

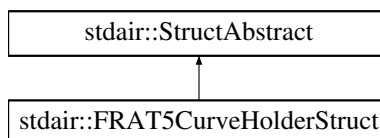
The documentation for this struct was generated from the following files:

- [stdair/basic/ForecastingMethod.hpp](#)
- [stdair/basic/ForecastingMethod.cpp](#)

### 32.84 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](#) ()

#### 32.84.1 Detailed Description

Structure holding the elements of a snapshot .

Definition at line 19 of file [FRAT5CurveHolderStruct.hpp](#).

#### 32.84.2 Constructor & Destructor Documentation

##### 32.84.2.1 stdair::FRAT5CurveHolderStruct::FRAT5CurveHolderStruct ( )

Constructor.

Definition at line 14 of file [FRAT5CurveHolderStruct.cpp](#).

32.84.2.2 `stdair::FRAT5CurveHolderStruct::FRAT5CurveHolderStruct ( const FRAT5CurveHolderStruct & iHolder )`

Copy constructor.

Definition at line 19 of file [FRAT5CurveHolderStruct.cpp](#).

32.84.2.3 `stdair::FRAT5CurveHolderStruct::~FRAT5CurveHolderStruct ( )`

Destructor.

Definition at line 24 of file [FRAT5CurveHolderStruct.cpp](#).

### 32.84.3 Member Function Documentation

32.84.3.1 `const FRAT5Curve_T & stdair::FRAT5CurveHolderStruct::getFRAT5Curve ( const std::string & iKey ) 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\(\)](#).

32.84.3.2 `void stdair::FRAT5CurveHolderStruct::addCurve ( const std::string & iKey, const FRAT5Curve_T & iCurve )`

Add a new curve to the holder.

Definition at line 42 of file [FRAT5CurveHolderStruct.cpp](#).

References [STDAIR\\_LOG\\_DEBUG](#).

Referenced by [stdair::BomRoot::addFRAT5Curve\(\)](#).

32.84.3.3 `void stdair::FRAT5CurveHolderStruct::toStream ( std::ostream & ioOut ) const`

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Definition at line 53 of file [FRAT5CurveHolderStruct.cpp](#).

References [describe\(\)](#).

32.84.3.4 `void stdair::FRAT5CurveHolderStruct::fromStream ( std::istream & ioin ) [virtual]`

Read a Business Object from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented from [stdair::StructAbstract](#).

Definition at line 58 of file [FRAT5CurveHolderStruct.cpp](#).

32.84.3.5 `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\(\)](#).

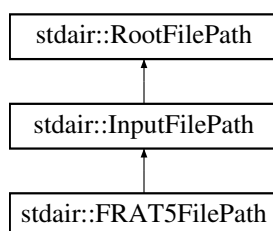
The documentation for this struct was generated from the following files:

- [stdair/bom/FRAT5CurveHolderStruct.hpp](#)
- [stdair/bom/FRAT5CurveHolderStruct.cpp](#)

### 32.85 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

#### 32.85.1 Detailed Description

FRAT5 input file.

Definition at line 88 of file [stdair\\_file.hpp](#).

#### 32.85.2 Constructor & Destructor Documentation

**32.85.2.1** `stdair::FRAT5FilePath::FRAT5FilePath ( const Filename\_T &iFilename )` `[inline], [explicit]`

Constructor.

Definition at line 93 of file [stdair\\_file.hpp](#).

#### 32.85.3 Member Function Documentation

**32.85.3.1** `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\(\)](#).

## 32.85.4 Member Data Documentation

## 32.85.4.1 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\(\)](#).

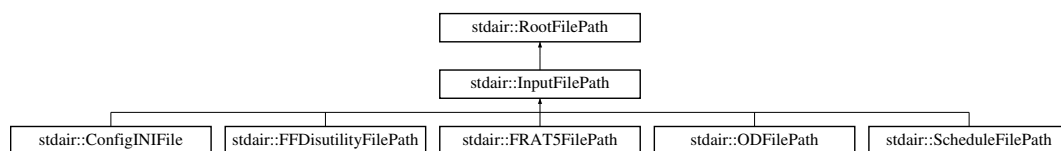
The documentation for this class was generated from the following file:

- [stdair/stdair\\_file.hpp](#)

## 32.86 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

## 32.86.1 Detailed Description

Input File.

Definition at line 54 of file [stdair\\_file.hpp](#).

## 32.86.2 Constructor &amp; Destructor Documentation

## 32.86.2.1 stdair::InputFilePath::InputFilePath ( const Filename\_T &amp;iFilename ) [inline]

Constructor.

Definition at line 57 of file [stdair\\_file.hpp](#).

## 32.86.3 Member Function Documentation

## 32.86.3.1 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\(\)](#).

#### 32.86.4 Member Data Documentation

##### 32.86.4.1 `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:

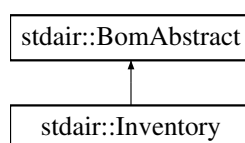
- [stdair/stdair\\_file.hpp](#)

### 32.87 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

- `template<typename BOM >`  
class [FacBom](#)
- `template<typename BOM >`  
class [FacCloneBom](#)
- class [FacBomManager](#)
- class [boost::serialization::access](#)

#### 32.87.1 Detailed Description

Class representing the actual attributes for an airline inventory.

Definition at line 34 of file [Inventory.hpp](#).

#### 32.87.2 Member Typedef Documentation

##### 32.87.2.1 `typedef InventoryKey stdair::Inventory::Key_T`

Definition allowing to retrieve the associated BOM key type.

Definition at line 45 of file [Inventory.hpp](#).

#### 32.87.3 Constructor & Destructor Documentation

##### 32.87.3.1 `stdair::Inventory::Inventory ( const Key_T & iKey )` [protected]

Constructor.

Definition at line 31 of file [Inventory.cpp](#).

##### 32.87.3.2 `stdair::Inventory::~~Inventory ( )` [protected]

Destructor.

Definition at line 38 of file [Inventory.cpp](#).

#### 32.87.4 Member Function Documentation

##### 32.87.4.1 `const Key_T& stdair::Inventory::getKey ( ) const` [inline]

Get the inventory key (airline code).

Definition at line 51 of file [Inventory.hpp](#).

References [\\_key](#).

#### 32.87.4.2 `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\(\)](#), [stdair::BomJSONExport::jsonExportFlightDateList\(\)](#), and [stdair::BomRetriever::retrieveSegmentDateFromLongKey\(\)](#).

#### 32.87.4.3 `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\(\)](#).

#### 32.87.4.4 `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\(\)](#).

#### 32.87.4.5 `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\(\)](#).

#### 32.87.4.6 `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\(\)](#).

#### 32.87.4.7 `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\(\)](#).

#### 32.87.4.8 `BomAbstract* const stdair::Inventory::getParent ( ) const` `[inline]`

Get the parent object.

Definition at line 76 of file [Inventory.hpp](#).

References [\\_parent](#).

#### 32.87.4.9 `const HolderMap_T& stdair::Inventory::getHolderMap ( ) const` `[inline]`

Get the map of children.

Definition at line 81 of file [Inventory.hpp](#).

References [\\_holderMap](#).

**32.87.4.10** `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**

<i>const</i>	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\(\)](#).

**32.87.4.11** `FlightDate* stdair::Inventory::getFlightDate ( const FlightDateKey & iFlightDateKey ) const`

Get a pointer on the [FlightDate](#) object corresponding to the given key.

**Note**

The [FlightDate](#) object can be inherited from, if needed. In that case, a `dynamic_cast<>` may be needed.

**Parameters**

<i>const</i>	<a href="#">FlightDateKey</a> & 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\(\)](#).

**32.87.4.12** `AirlineFeature* stdair::Inventory::getAirlineFeature ( ) const` `[inline]`

Get the airline feature.

Definition at line 112 of file [Inventory.hpp](#).

References [\\_airlineFeature](#).

Referenced by [stdair::BomManager::getObjectPtr\(\)](#).

**32.87.4.13** `void stdair::Inventory::toStream ( std::ostream & ioOut ) const` `[inline],[virtual]`

Dump a Business Object into an output stream.

**Parameters**

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Implements [stdair::BomAbstract](#).

Definition at line 132 of file [Inventory.hpp](#).

References [toString\(\)](#).

**32.87.4.14** `void stdair::Inventory::fromStream ( std::istream & ioln )` `[inline],[virtual]`

Read a Business Object from an input stream.

## Parameters

<i>istream</i> &	the input stream.
------------------	-------------------

Implements [stdair::BomAbstract](#).

Definition at line 141 of file [Inventory.hpp](#).

32.87.4.15 `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\(\)](#).

32.87.4.16 `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\(\)](#).

32.87.4.17 `template<class Archive > void stdair::Inventory::serialize ( Archive & ar, const unsigned int iFileVersion )`

Serialisation.

Definition at line 160 of file [CmdBomSerialiser.cpp](#).

References [\\_key](#).

### 32.87.5 Friends And Related Function Documentation

32.87.5.1 `template<typename BOM > friend class FacBom` `[friend]`

Definition at line 35 of file [Inventory.hpp](#).

32.87.5.2 `template<typename BOM > friend class FacCloneBom` `[friend]`

Definition at line 36 of file [Inventory.hpp](#).

32.87.5.3 `friend class FacBomManager` `[friend]`

Definition at line 37 of file [Inventory.hpp](#).

32.87.5.4 `friend class boost::serialization::access` `[friend]`

Definition at line 38 of file [Inventory.hpp](#).

### 32.87.6 Member Data Documentation

32.87.6.1 `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\(\)](#).

**32.87.6.2 BomAbstract\*** stdair::Inventory::\_parent [protected]

Pointer on the parent class ([BomRoot](#)).

Definition at line 209 of file [Inventory.hpp](#).

Referenced by [getParent\(\)](#).

**32.87.6.3 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\(\)](#).

**32.87.6.4 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:

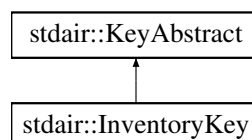
- [stdair/bom/Inventory.hpp](#)
- [stdair/bom/Inventory.cpp](#)
- [stdair/command/CmdBomSerialiser.cpp](#)

**32.88 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](#)

### 32.88.1 Detailed Description

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

Definition at line 26 of file [InventoryKey.hpp](#).

### 32.88.2 Constructor & Destructor Documentation

#### 32.88.2.1 `stdair::InventoryKey::InventoryKey ( const AirlineCode_T & iAirlineCode )`

Constructor.

Definition at line 23 of file [InventoryKey.cpp](#).

#### 32.88.2.2 `stdair::InventoryKey::InventoryKey ( const InventoryKey & iKey )`

Copy constructor.

Definition at line 28 of file [InventoryKey.cpp](#).

#### 32.88.2.3 `stdair::InventoryKey::~~InventoryKey ( )`

Destructor.

Definition at line 33 of file [InventoryKey.cpp](#).

### 32.88.3 Member Function Documentation

#### 32.88.3.1 `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\(\)](#).

#### 32.88.3.2 `void stdair::InventoryKey::toStream ( std::ostream & ioOut ) const` [virtual]

Dump a Business Object Key into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 37 of file [InventoryKey.cpp](#).

References [toString\(\)](#).

#### 32.88.3.3 `void stdair::InventoryKey::fromStream ( std::istream & ioIn )` [virtual]

Read a Business Object Key from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 42 of file [InventoryKey.cpp](#).

32.88.3.4 `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\(\)](#).

32.88.3.5 `template<class Archive > void stdair::InventoryKey::serialize ( Archive & ar, const unsigned int iFileVersion )`

Serialisation.

Definition at line 68 of file [InventoryKey.cpp](#).

## 32.88.4 Friends And Related Function Documentation

32.88.4.1 `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:

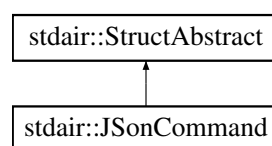
- [stdair/bom/InventoryKey.hpp](#)
- [stdair/bom/InventoryKey.cpp](#)

## 32.89 stdair::JJsonCommand Struct Reference

Enumeration of json commands.

```
#include <stdair/basic/JJsonCommand.hpp>
```

Inheritance diagram for `stdair::JJsonCommand`:



### Public Types

- enum [EN\\_JJsonCommand](#) {  
[LIST](#) = 0, [FLIGHT\\_DATE](#), [EVENT\\_LIST](#), [BREAK\\_POINT](#),  
[RUN](#), [RESET](#), [STATUS](#), [CONFIG](#),  
[LAST\\_VALUE](#) }

### Public Member Functions

- [EN\\_JJsonCommand](#) [getCommand](#) ( ) const
- const std::string [describe](#) ( ) const
- bool [operator==](#) (const [EN\\_JJsonCommand](#) &) const
- [JJsonCommand](#) (const [EN\\_JJsonCommand](#) &)



- [JJsonCommand](#) (const std::string &)
- [JJsonCommand](#) (const [JJsonCommand](#) &)
- void [toStream](#) (std::ostream &ioOut) const
- virtual void [fromStream](#) (std::istream &ioIn)

#### Static Public Member Functions

- static [EN\\_JJsonCommand](#) [getCommand](#) (const std::string &iCommandStr)
- static std::string [getLabel](#) (const [EN\\_JJsonCommand](#) &)
- static std::string [describeLabels](#) ()

#### 32.89.1 Detailed Description

Enumeration of json commands.

Definition at line 17 of file [JJsonCommand.hpp](#).

#### 32.89.2 Member Enumeration Documentation

##### 32.89.2.1 enum stdair::JJsonCommand::EN\_JJsonCommand

Enumerator

***LIST***  
***FLIGHT\_DATE***  
***EVENT\_LIST***  
***BREAK\_POINT***  
***RUN***  
***RESET***  
***STATUS***  
***CONFIG***  
***LAST\_VALUE***

Definition at line 19 of file [JJsonCommand.hpp](#).

#### 32.89.3 Constructor & Destructor Documentation

##### 32.89.3.1 stdair::JJsonCommand::JJsonCommand ( const [EN\\_JJsonCommand](#) & )

Main Constructor.

##### 32.89.3.2 stdair::JJsonCommand::JJsonCommand ( const std::string & *iCommandStr* )

Alternative constructor.

Definition at line 71 of file [JJsonCommand.cpp](#).

References [getCommand\(\)](#).

##### 32.89.3.3 stdair::JJsonCommand::JJsonCommand ( const [JJsonCommand](#) & *iJJsonCommand* )

Default copy constructor.

Definition at line 25 of file [JJsonCommand.cpp](#).

## 32.89.4 Member Function Documentation

## 32.89.4.1 JJsonCommand::EN\_JJsonCommand stdair::JJsonCommand::getCommand ( const std::string &amp; 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 [JJsonCommand.cpp](#).

References [BREAK\\_POINT](#), [CONFIG](#), [describeLabels\(\)](#), [EVENT\\_LIST](#), [FLIGHT\\_DATE](#), [LAST\\_VALUE](#), [LIST](#), [RESET](#), [RUN](#), and [STATUS](#).

Referenced by [stdair::BomJSONImport::jsonImportCommand\(\)](#).

## 32.89.4.2 std::string stdair::JJsonCommand::getLabel ( const EN\_JJsonCommand &amp; iCommand ) [static]

Get a label of a command

Definition at line 66 of file [JJsonCommand.cpp](#).

## 32.89.4.3 std::string stdair::JJsonCommand::describeLabels ( ) [static]

List the labels.

Definition at line 77 of file [JJsonCommand.cpp](#).

References [LAST\\_VALUE](#).

Referenced by [getCommand\(\)](#).

## 32.89.4.4 JJsonCommand::EN\_JJsonCommand stdair::JJsonCommand::getCommand ( ) const

Get the enumerated value.

Definition at line 89 of file [JJsonCommand.cpp](#).

Referenced by [JJsonCommand\(\)](#).

## 32.89.4.5 const std::string stdair::JJsonCommand::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 [JJsonCommand.cpp](#).

## 32.89.4.6 bool stdair::JJsonCommand::operator== ( const EN\_JJsonCommand &amp; iCommand ) const

Comparison operator.

Definition at line 102 of file [JJsonCommand.cpp](#).

## 32.89.4.7 void stdair::StructAbstract::toStream ( std::ostream &amp; ioOut ) const [inline], [inherited]

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Definition at line 29 of file [StructAbstract.hpp](#).

References [stdair::StructAbstract::describe\(\)](#).

**32.89.4.8** `virtual void stdair::StructAbstract::fromStream ( std::istream & ioIn )` `[inline]`, `[virtual]`,  
`[inherited]`

Read a Business Object from an input stream.

## Parameters

<i>istream</i> &	the input stream.
------------------	-------------------

Reimplemented in [stdair::BookingRequestStruct](#), [stdair::EventStruct](#), [stdair::TravelSolutionStruct](#), [stdair::VirtualClassStruct](#), [stdair::ConfigHolderStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::ProgressStatusSet](#), [stdair::YieldRange](#), [stdair::CancellationStruct](#), [stdair::AirlineStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::FFDisutilityCurveHolderStruct](#), [stdair::FRAT5CurveHolderStruct](#), and [stdair::BreakPointStruct](#).

Definition at line 38 of file [StructAbstract.hpp](#).

Referenced by [operator>>\(\)](#).

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

- [stdair/basic/JJsonCommand.hpp](#)
- [stdair/basic/JJsonCommand.cpp](#)

## 32.90 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](#)

#### 32.90.1 Detailed Description

JSON-formatted string.

Definition at line 16 of file [stdair\\_json.hpp](#).

#### 32.90.2 Constructor & Destructor Documentation

**32.90.2.1** [stdair::JSONString::JSONString](#) ( const std::string & *iJsonString* ) [\[inline\]](#), [\[explicit\]](#)

Main Constructor.

Definition at line 21 of file [stdair\\_json.hpp](#).

**32.90.2.2** [stdair::JSONString::JSONString](#) ( ) [\[inline\]](#), [\[explicit\]](#)

Default constructor.

Definition at line 26 of file [stdair\\_json.hpp](#).

**32.90.2.3** [virtual stdair::JSONString::~~JSONString](#) ( ) [\[inline\]](#), [\[virtual\]](#)

Destructor.

Definition at line 31 of file [stdair\\_json.hpp](#).

### 32.90.3 Member Function Documentation

#### 32.90.3.1 `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\(\)](#).

### 32.90.4 Member Data Documentation

#### 32.90.4.1 `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:

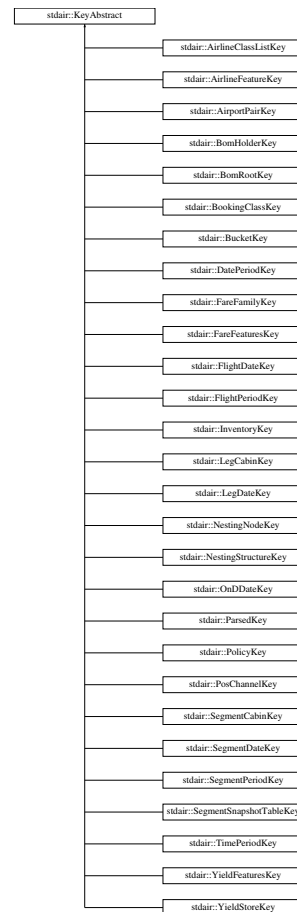
- [stdair/stdair\\_json.hpp](#)

## 32.91 `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.*

### 32.91.1 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](#).

### 32.91.2 Constructor & Destructor Documentation

#### 32.91.2.1 virtual stdair::KeyAbstract::~KeyAbstract ( ) [inline], [virtual]

Default destructor.

Definition at line 61 of file [KeyAbstract.hpp](#).

### 32.91.3 Member Function Documentation

**32.91.3.1** `virtual void stdair::KeyAbstract::toStream ( std::ostream & ioOut ) const [inline],[virtual]`

Dump a Business Object Key into an output stream.

#### Parameters

<i>in, out</i>	<i>ostream&amp;</i>	the output stream.
----------------	---------------------	--------------------

Reimplemented in [stdair::FareFeaturesKey](#), [stdair::OnDDateKey](#), [stdair::FlightDateKey](#), [stdair::AirlineClassListKey](#), [stdair::InventoryKey](#), [stdair::BomRootKey](#), [stdair::FareFamilyKey](#), [stdair::LegCabinKey](#), [stdair::NestingNodeKey](#), [stdair::NestingStructureKey](#), [stdair::PolicyKey](#), [stdair::SegmentCabinKey](#), [stdair::SegmentSnapshotTableKey](#), [stdair::SegmentDateKey](#), [stdair::BucketKey](#), [stdair::YieldFeaturesKey](#), [stdair::PosChannelKey](#), [stdair::AirportPairKey](#), [stdair::TimePeriodKey](#), [stdair::ParsedKey](#), [stdair::DatePeriodKey](#), [stdair::BookingClassKey](#), [stdair::LegDateKey](#), [stdair::SegmentPeriodKey](#), [stdair::FlightPeriodKey](#), [stdair::YieldStoreKey](#), [stdair::AirlineFeatureKey](#), and [stdair::BomHolderKey](#).

Definition at line 36 of file [KeyAbstract.hpp](#).

**32.91.3.2** `virtual void stdair::KeyAbstract::fromStream ( std::istream & ioin ) [inline],[virtual]`

Read a Business Object Key from an input stream.

#### Parameters

<i>in, out</i>	<i>istream&amp;</i>	the input stream.
----------------	---------------------	-------------------

Reimplemented in [stdair::FareFeaturesKey](#), [stdair::OnDDateKey](#), [stdair::FlightDateKey](#), [stdair::AirlineClassListKey](#), [stdair::InventoryKey](#), [stdair::BomRootKey](#), [stdair::FareFamilyKey](#), [stdair::LegCabinKey](#), [stdair::NestingNodeKey](#), [stdair::NestingStructureKey](#), [stdair::PolicyKey](#), [stdair::SegmentCabinKey](#), [stdair::SegmentSnapshotTableKey](#), [stdair::SegmentDateKey](#), [stdair::BucketKey](#), [stdair::YieldFeaturesKey](#), [stdair::PosChannelKey](#), [stdair::AirportPairKey](#), [stdair::TimePeriodKey](#), [stdair::ParsedKey](#), [stdair::DatePeriodKey](#), [stdair::BookingClassKey](#), [stdair::LegDateKey](#), [stdair::SegmentPeriodKey](#), [stdair::FlightPeriodKey](#), [stdair::YieldStoreKey](#), [stdair::AirlineFeatureKey](#), and [stdair::BomHolderKey](#).

Definition at line 43 of file [KeyAbstract.hpp](#).

Referenced by [operator>>\(\)](#).

**32.91.3.3** `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

<i>out</i>	<i>const</i>	std::string The serialised version of the Business Object Key.
------------	--------------	--

Reimplemented in [stdair::FareFeaturesKey](#), [stdair::OnDDateKey](#), [stdair::FlightDateKey](#), [stdair::AirlineClassListKey](#), [stdair::InventoryKey](#), [stdair::BomRootKey](#), [stdair::FareFamilyKey](#), [stdair::LegCabinKey](#), [stdair::NestingNodeKey](#), [stdair::NestingStructureKey](#), [stdair::PolicyKey](#), [stdair::SegmentCabinKey](#), [stdair::SegmentSnapshotTableKey](#), [stdair::SegmentDateKey](#), [stdair::BucketKey](#), [stdair::YieldFeaturesKey](#), [stdair::PosChannelKey](#), [stdair::AirportPairKey](#), [stdair::TimePeriodKey](#), [stdair::ParsedKey](#), [stdair::DatePeriodKey](#), [stdair::BookingClassKey](#), [stdair::LegDateKey](#), [stdair::SegmentPeriodKey](#), [stdair::FlightPeriodKey](#), [stdair::YieldStoreKey](#), [stdair::AirlineFeatureKey](#), and [stdair::BomHolderKey](#).

Definition at line 56 of file [KeyAbstract.hpp](#).

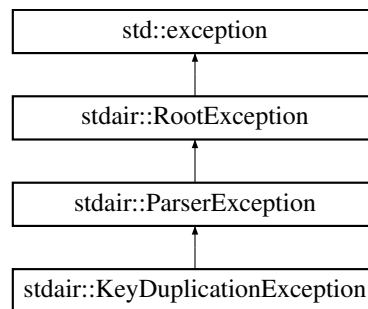
The documentation for this struct was generated from the following file:

- [stdair/bom/KeyAbstract.hpp](#)

## 32.92 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](#)

### 32.92.1 Detailed Description

Key duplication.

Definition at line 149 of file [stdair\\_exceptions.hpp](#).

### 32.92.2 Constructor & Destructor Documentation

32.92.2.1 `stdair::KeyDuplicationException::KeyDuplicationException ( const std::string &iWhat )` `[inline]`

Constructor.

Definition at line 152 of file [stdair\\_exceptions.hpp](#).

### 32.92.3 Member Function Documentation

32.92.3.1 `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\(\)](#).



### 32.92.4 Member Data Documentation

#### 32.92.4.1 `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\(\)](#).

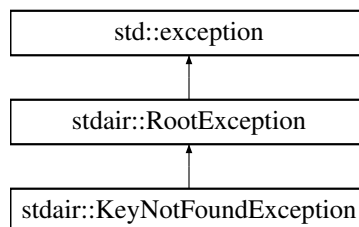
The documentation for this class was generated from the following file:

- [stdair/stdair\\_exceptions.hpp](#)

### 32.93 `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](#)

#### 32.93.1 Detailed Description

Not found key.

Definition at line 126 of file [stdair\\_exceptions.hpp](#).

#### 32.93.2 Constructor & Destructor Documentation

##### 32.93.2.1 `stdair::KeyNotFoundException::KeyNotFoundException ( const std::string & iWhat )` `[inline]`

Constructor.

Definition at line 129 of file [stdair\\_exceptions.hpp](#).

#### 32.93.3 Member Function Documentation

##### 32.93.3.1 `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\(\)](#).

#### 32.93.4 Member Data Documentation

##### 32.93.4.1 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:

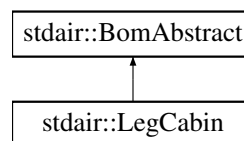
- [stdair/stdair\\_exceptions.hpp](#)

## 32.94 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](#) &iWLSSeats)
- 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

- `template<typename BOM >`  
class [FacBom](#)
- `template<typename BOM >`  
class [FacCloneBom](#)
- class [FacBomManager](#)

## 32.94.1 Detailed Description

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

Definition at line 25 of file [LegCabin.hpp](#).

## 32.94.2 Member Typedef Documentation

32.94.2.1 `typedef LegCabinKey stdair::LegCabin::Key_T`

Definition allowing to retrieve the associated BOM key type.

Definition at line 35 of file [LegCabin.hpp](#).

## 32.94.3 Constructor &amp; Destructor Documentation

32.94.3.1 `stdair::LegCabin::LegCabin ( const Key_T & iKey )` [protected]

Constructor.

Definition at line 46 of file [LegCabin.cpp](#).

32.94.3.2 `stdair::LegCabin::~~LegCabin ( )` [protected]

Destructor.

Definition at line 69 of file [LegCabin.cpp](#).

### 32.94.4 Member Function Documentation

#### 32.94.4.1 `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](#).

#### 32.94.4.2 `BomAbstract* const stdair::LegCabin::getParent ( ) const` `[inline]`

Get the parent object.

Definition at line 49 of file [LegCabin.hpp](#).

References [\\_parent](#).

#### 32.94.4.3 `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\(\)](#).

#### 32.94.4.4 `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\(\)](#).

#### 32.94.4.5 `const HolderMap_T& stdair::LegCabin::getHolderMap ( ) const` `[inline]`

Get the map of children holders.

Definition at line 72 of file [LegCabin.hpp](#).

References [\\_holderMap](#).

#### 32.94.4.6 `const CabinCapacity_T& stdair::LegCabin::getOfferedCapacity ( ) const` `[inline]`

Get the cabin offered capacity.

Definition at line 77 of file [LegCabin.hpp](#).

References [\\_offeredCapacity](#).

#### 32.94.4.7 `const CabinCapacity_T& stdair::LegCabin::getPhysicalCapacity ( ) const` `[inline]`

Get the cabin physical capacity.

Definition at line 82 of file [LegCabin.hpp](#).

References [\\_physicalCapacity](#).

#### 32.94.4.8 `const NbOfSeats_T& stdair::LegCabin::getSoldSeat ( ) const` `[inline]`

Get the number of sold seat.

Definition at line 87 of file [LegCabin.hpp](#).

References [\\_soldSeat](#).

32.94.4.9 **const CommittedSpace\_T&** stdair::LegCabin::getCommittedSpace ( ) const [inline]

Get the value of committed space.

Definition at line 92 of file [LegCabin.hpp](#).

References [\\_committedSpace](#).

32.94.4.10 **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](#).

32.94.4.11 **const Availability\_T&** stdair::LegCabin::getAvailability ( ) const [inline]

Get the value of the availability.

Definition at line 102 of file [LegCabin.hpp](#).

References [\\_availability](#).

32.94.4.12 **const BidPrice\_T&** stdair::LegCabin::getCurrentBidPrice ( ) const [inline]

Get the current Bid-Price.

Definition at line 107 of file [LegCabin.hpp](#).

References [\\_currentBidPrice](#).

32.94.4.13 **const BidPrice\_T&** stdair::LegCabin::getPreviousBidPrice ( ) const [inline]

Get the previous Bid-Price.

Definition at line 112 of file [LegCabin.hpp](#).

References [\\_previousBidPrice](#).

32.94.4.14 **const BidPriceVector\_T&** stdair::LegCabin::getBidPriceVector ( ) const [inline]

Get the Bid-Price Vector.

Definition at line 117 of file [LegCabin.hpp](#).

References [\\_bidPriceVector](#).

32.94.4.15 **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](#).

32.94.4.16 **const AuthorizationLevel\_T&** stdair::LegCabin::getAuthorizationLevel ( ) const [inline]

Authorisation Level (AU).

Definition at line 127 of file [LegCabin.hpp](#).

References [\\_au](#).

**32.94.4.17** `const UPR_T& stdair::LegCabin::getUPR ( ) const [inline]`

Unsold Protection (UPR).

Definition at line 132 of file [LegCabin.hpp](#).

References [\\_upr](#).

**32.94.4.18** `const Availability_T& stdair::LegCabin::getNetAvailability ( ) const [inline]`

Net Availability (NAV).

Definition at line 137 of file [LegCabin.hpp](#).

References [\\_nav](#).

**32.94.4.19** `const Availability_T& stdair::LegCabin::getGrossAvailability ( ) const [inline]`

Gross Availability (GAV).

Definition at line 142 of file [LegCabin.hpp](#).

References [\\_gav](#).

**32.94.4.20** `const OverbookingRate_T& stdair::LegCabin::getAvgCancellationPercentage ( ) const [inline]`

Average Cancellation Percentage (ACP).

Definition at line 147 of file [LegCabin.hpp](#).

References [\\_acp](#).

**32.94.4.21** `const NbOfSeats_T& stdair::LegCabin::getETB ( ) const [inline]`

Expected to Board (ETB).

Definition at line 152 of file [LegCabin.hpp](#).

References [\\_etb](#).

**32.94.4.22** `const NbOfSeats_T& stdair::LegCabin::getStaffNbOfSeats ( ) const [inline]`

Number of staff bookings.

Definition at line 157 of file [LegCabin.hpp](#).

References [\\_staffNbOfBookings](#).

**32.94.4.23** `const NbOfSeats_T& stdair::LegCabin::getWLNbOfSeats ( ) const [inline]`

Number of wait-listed bookings.

Definition at line 162 of file [LegCabin.hpp](#).

References [\\_wlNbOfBookings](#).

**32.94.4.24** `const NbOfSeats_T& stdair::LegCabin::getGroupNbOfSeats ( ) const [inline]`

Number of group bookings.

Definition at line 167 of file [LegCabin.hpp](#).

References [\\_groupNbOfBookings](#).

**32.94.4.25** `VirtualClassList_T& stdair::LegCabin::getVirtualClassList ( ) [inline]`

The virtual class list.

Definition at line 172 of file [LegCabin.hpp](#).

References [\\_virtualClassList](#).

**32.94.4.26** `BidPriceVector_T& stdair::LegCabin::getBidPriceVector ( ) [inline]`

Reset the bid price vector and return it.

Definition at line 177 of file [LegCabin.hpp](#).

References [\\_bidPriceVector](#).

**32.94.4.27** `const YieldLevelDemandMap_T& stdair::LegCabin::getYieldLevelDemandMap ( ) [inline]`

Get the yield-demand map.

Definition at line 183 of file [LegCabin.hpp](#).

References [\\_yieldLevelDemandMap](#).

**32.94.4.28** `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\(\)](#).

**32.94.4.29** `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](#).

**32.94.4.30** `void stdair::LegCabin::setCommittedSpace ( const CommittedSpace_T & iCommittedSpace ) [inline]`

Set the value of committed space.

Definition at line 199 of file [LegCabin.hpp](#).

References [\\_committedSpace](#).

**32.94.4.31** `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\(\)](#).

**32.94.4.32** `void stdair::LegCabin::setAvailability ( const Availability_T & iAvailability ) [inline]`

Set the value of availability.

Definition at line 209 of file [LegCabin.hpp](#).

References [\\_availability](#).

**32.94.4.33** `void stdair::LegCabin::setCurrentBidPrice ( const BidPrice_T & iBidPrice ) [inline]`

Set the current Bid-Price.

Definition at line 214 of file [LegCabin.hpp](#).

References [\\_currentBidPrice](#).



32.94.4.34 void stdair::LegCabin::setPreviousBidPrice ( const BidPrice\_T & iBidPrice ) [inline]

Set the previous Bid-Price.

Definition at line 219 of file [LegCabin.hpp](#).

References [\\_previousBidPrice](#).

32.94.4.35 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](#).

32.94.4.36 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](#).

32.94.4.37 void stdair::LegCabin::setAuthorizationLevel ( const AuthorizationLevel\_T & iAU ) [inline]

Set the Authorisation Level (AU).

Definition at line 234 of file [LegCabin.hpp](#).

References [\\_au](#).

32.94.4.38 void stdair::LegCabin::setUPR ( const UPR\_T & iUPR ) [inline]

Set the Unsold Protection (UPR).

Definition at line 239 of file [LegCabin.hpp](#).

References [\\_upr](#).

32.94.4.39 void stdair::LegCabin::setNetAvailability ( const Availability\_T & iNAV ) [inline]

Set the Net Availability (NAV).

Definition at line 244 of file [LegCabin.hpp](#).

References [\\_nav](#).

32.94.4.40 void stdair::LegCabin::setGrossAvailability ( const Availability\_T & iGAV ) [inline]

Set the Gross Availability (GAV).

Definition at line 249 of file [LegCabin.hpp](#).

References [\\_gav](#).

32.94.4.41 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](#).

32.94.4.42 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](#).

32.94.4.43 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](#).

32.94.4.44 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](#).

32.94.4.45 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](#).

32.94.4.46 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](#).

32.94.4.47 void stdair::LegCabin::toStream ( std::ostream & ioOut ) const [inline],[virtual]

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Implements [stdair::BomAbstract](#).

Definition at line 288 of file [LegCabin.hpp](#).

References [toString\(\)](#).

32.94.4.48 void stdair::LegCabin::fromStream ( std::istream & ioln ) [inline],[virtual]

Read a Business Object from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Implements [stdair::BomAbstract](#).

Definition at line 296 of file [LegCabin.hpp](#).

32.94.4.49 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\(\)](#).

32.94.4.50 `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\(\)](#).

32.94.4.51 `const std::string stdair::LegCabin::displayVirtualClassList ( ) const`

Display the virtual class list content.

Definition at line 96 of file [LegCabin.cpp](#).

References [\\_virtualClassList](#), [stdair::VirtualClassStruct::getCumulatedBookingLimit\(\)](#), [stdair::VirtualClassStruct::getCumulatedProtection\(\)](#), and [stdair::VirtualClassStruct::getYield\(\)](#).

32.94.4.52 `void stdair::LegCabin::updateFromReservation ( const NbOfBookings_T & iNbOfBookings )`

Register a sale.

Definition at line 114 of file [LegCabin.cpp](#).

References [\\_availabilityPool](#), [\\_committedSpace](#), and [\\_offeredCapacity](#).

32.94.4.53 `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](#).

32.94.4.54 `void stdair::LegCabin::emptyVirtualClassList ( ) [inline]`

Empty the virtual class list.

Definition at line 334 of file [LegCabin.hpp](#).

References [\\_virtualClassList](#).

32.94.4.55 `void stdair::LegCabin::emptyBidPriceVector ( ) [inline]`

Empty the bid price vector.

Definition at line 341 of file [LegCabin.hpp](#).

References [\\_bidPriceVector](#).

32.94.4.56 `void stdair::LegCabin::addDemandInformation ( const YieldValue_T & iYield, const MeanValue_T & iMeanValue, const StdDevValue_T & iStdDevValue )`

Add demand information.

Definition at line 133 of file [LegCabin.cpp](#).

References [\\_yieldLevelDemandMap](#).

32.94.4.57 `void stdair::LegCabin::emptyYieldLevelDemandMap ( ) [inline]`

Reset the (yield level,demand) map.

Definition at line 354 of file [LegCabin.hpp](#).

References [\\_yieldLevelDemandMap](#).

### 32.94.5 Friends And Related Function Documentation

#### 32.94.5.1 `template<typename BOM > friend class FacBom` [friend]

Definition at line 26 of file [LegCabin.hpp](#).

#### 32.94.5.2 `template<typename BOM > friend class FacCloneBom` [friend]

Definition at line 27 of file [LegCabin.hpp](#).

#### 32.94.5.3 `friend class FacBomManager` [friend]

Definition at line 28 of file [LegCabin.hpp](#).

### 32.94.6 Member Data Documentation

#### 32.94.6.1 `Key_T stdair::LegCabin::_key` [protected]

Primary key (cabin code).

Definition at line 387 of file [LegCabin.hpp](#).

Referenced by [describeKey\(\)](#), [getCabinCode\(\)](#), and [getKey\(\)](#).

#### 32.94.6.2 `BomAbstract* stdair::LegCabin::_parent` [protected]

Pointer on the parent class ([LegDate](#)).

Definition at line 392 of file [LegCabin.hpp](#).

Referenced by [getParent\(\)](#).

#### 32.94.6.3 `HolderMap_T stdair::LegCabin::_holderMap` [protected]

Map holding the children ([Bucket](#) objects).

Definition at line 397 of file [LegCabin.hpp](#).

Referenced by [getHolderMap\(\)](#).

#### 32.94.6.4 `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\(\)](#).

#### 32.94.6.5 `CabinCapacity_T stdair::LegCabin::_physicalCapacity` [protected]

Physical capacity of the cabin.

Definition at line 403 of file [LegCabin.hpp](#).

Referenced by [getPhysicalCapacity\(\)](#), and [setCapacities\(\)](#).

#### 32.94.6.6 `NbOfSeats_T stdair::LegCabin::_soldSeat` [protected]

Aggregated number of sold seats.

Definition at line 406 of file [LegCabin.hpp](#).

Referenced by [getSoldSeat\(\)](#), and [setSoldSeat\(\)](#).

#### 32.94.6.7 **CommittedSpace\_T** stdair::LegCabin::\_committedSpace [protected]

Definition at line 409 of file [LegCabin.hpp](#).

Referenced by [getCommittedSpace\(\)](#), [setCapacities\(\)](#), [setCommittedSpace\(\)](#), and [updateFromReservation\(\)](#).

#### 32.94.6.8 **Availability\_T** stdair::LegCabin::\_availabilityPool [protected]

Availability pool.

Definition at line 412 of file [LegCabin.hpp](#).

Referenced by [getAvailabilityPool\(\)](#), [setAvailabilityPool\(\)](#), [updateCurrentBidPrice\(\)](#), and [updateFromReservation\(\)](#).

#### 32.94.6.9 **Availability\_T** stdair::LegCabin::\_availability [protected]

Availability.

Definition at line 415 of file [LegCabin.hpp](#).

Referenced by [getAvailability\(\)](#), and [setAvailability\(\)](#).

#### 32.94.6.10 **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\(\)](#).

#### 32.94.6.11 **BidPrice\_T** stdair::LegCabin::\_previousBidPrice [protected]

Previous Bid-Price (BP).

Definition at line 421 of file [LegCabin.hpp](#).

Referenced by [getPreviousBidPrice\(\)](#), [setPreviousBidPrice\(\)](#), and [updatePreviousBidPrice\(\)](#).

#### 32.94.6.12 **BidPriceVector\_T** stdair::LegCabin::\_bidPriceVector [protected]

Bid-Price Vector (BPV).

Definition at line 424 of file [LegCabin.hpp](#).

Referenced by [emptyBidPriceVector\(\)](#), [getBidPriceVector\(\)](#), and [updateCurrentBidPrice\(\)](#).

#### 32.94.6.13 **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\(\)](#).

#### 32.94.6.14 **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\(\)](#).

#### 32.94.6.15 **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\(\)](#).

#### 32.94.6.16 AuthorizationLevel\_T stdair::LegCabin::\_au

Authorisation Level (AU).

Definition at line 438 of file [LegCabin.hpp](#).

Referenced by [getAuthorizationLevel\(\)](#), and [setAuthorizationLevel\(\)](#).

#### 32.94.6.17 UPR\_T stdair::LegCabin::\_upr

Unsold Protection (UPR).

Definition at line 441 of file [LegCabin.hpp](#).

Referenced by [getUPR\(\)](#), and [setUPR\(\)](#).

#### 32.94.6.18 Availability\_T stdair::LegCabin::\_nav

Net Availability (NAV).

Definition at line 444 of file [LegCabin.hpp](#).

Referenced by [getNetAvailability\(\)](#), and [setNetAvailability\(\)](#).

#### 32.94.6.19 Availability\_T stdair::LegCabin::\_gav

Gross Availability (GAV).

Definition at line 447 of file [LegCabin.hpp](#).

Referenced by [getGrossAvailability\(\)](#), and [setGrossAvailability\(\)](#).

#### 32.94.6.20 OverbookingRate\_T stdair::LegCabin::\_acp

Average Cancellation Percentage (ACP).

Definition at line 450 of file [LegCabin.hpp](#).

Referenced by [getAvgCancellationPercentage\(\)](#), and [setAvgCancellationPercentage\(\)](#).

#### 32.94.6.21 NbOfSeats\_T stdair::LegCabin::\_etb

Expected to Board (ETB).

Definition at line 453 of file [LegCabin.hpp](#).

Referenced by [getETB\(\)](#), and [setETB\(\)](#).

#### 32.94.6.22 NbOfSeats\_T stdair::LegCabin::\_staffNbOfBookings

Number of staff bookings.

Definition at line 456 of file [LegCabin.hpp](#).

Referenced by [getStaffNbOfSeats\(\)](#), and [setStaffNbOfSeats\(\)](#).

#### 32.94.6.23 NbOfSeats\_T stdair::LegCabin::\_wlnNbOfBookings

Number of wait-listed bookings.

Definition at line 459 of file [LegCabin.hpp](#).

Referenced by [getWLNbOfSeats\(\)](#), and [setWLNbOfSeats\(\)](#).

#### 32.94.6.24 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:

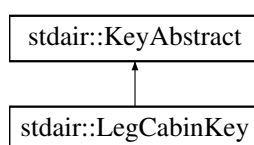
- [stdair/bom/LegCabin.hpp](#)
- [stdair/bom/LegCabin.cpp](#)

### 32.95 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](#)

#### 32.95.1 Detailed Description

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

Definition at line 26 of file [LegCabinKey.hpp](#).

#### 32.95.2 Constructor & Destructor Documentation

##### 32.95.2.1 stdair::LegCabinKey::LegCabinKey ( const [CabinCode\\_T](#) & iCabinCode )

Constructor.

Definition at line 23 of file [LegCabinKey.cpp](#).

##### 32.95.2.2 stdair::LegCabinKey::LegCabinKey ( const [LegCabinKey](#) & iKey )

Copy constructor.

Definition at line 28 of file [LegCabinKey.cpp](#).

### 32.95.2.3 stdair::LegCabinKey::~~LegCabinKey ( )

Destructor.

Definition at line 33 of file [LegCabinKey.cpp](#).

## 32.95.3 Member Function Documentation

### 32.95.3.1 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\(\)](#).

### 32.95.3.2 void stdair::LegCabinKey::toStream ( std::ostream & ioOut ) const [virtual]

Dump a Business Object Key into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 37 of file [LegCabinKey.cpp](#).

References [toString\(\)](#).

### 32.95.3.3 void stdair::LegCabinKey::fromStream ( std::istream & ioIn ) [virtual]

Read a Business Object Key from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 42 of file [LegCabinKey.cpp](#).

### 32.95.3.4 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\(\)](#).

### 32.95.3.5 template<class Archive > void stdair::LegCabinKey::serialize ( Archive & ar, const unsigned int iFileVersion )

Serialisation.

Definition at line 68 of file [LegCabinKey.cpp](#).

## 32.95.4 Friends And Related Function Documentation



#### 32.95.4.1 friend class boost::serialization::access [friend]

Definition at line 27 of file [LegCabinKey.hpp](#).

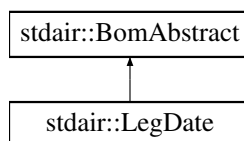
The documentation for this struct was generated from the following files:

- [stdair/bom/LegCabinKey.hpp](#)
- [stdair/bom/LegCabinKey.cpp](#)

### 32.96 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

- `template<typename BOM >`  
class [FacBom](#)
- `template<typename BOM >`  
class [FacCloneBom](#)
- class [FacBomManager](#)

## 32.96.1 Detailed Description

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

Definition at line 25 of file [LegDate.hpp](#).

## 32.96.2 Member Typedef Documentation

32.96.2.1 `typedef LegDateKey stdair::LegDate::Key_T`

Definition allowing to retrieve the associated BOM key type.

Definition at line 33 of file [LegDate.hpp](#).

## 32.96.3 Constructor &amp; Destructor Documentation

32.96.3.1 `stdair::LegDate::LegDate ( const Key_T & iKey )` [protected]

Constructor.

Definition at line 38 of file [LegDate.cpp](#).

32.96.3.2 `stdair::LegDate::~~LegDate ( )` [protected], [virtual]

Destructor.

Definition at line 44 of file [LegDate.cpp](#).

### 32.96.4 Member Function Documentation

#### 32.96.4.1 `const Key_T& stdair::LegDate::getKey ( ) const` [inline]

Get the leg-date key.

Definition at line 39 of file [LegDate.hpp](#).

References [\\_key](#).

#### 32.96.4.2 `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\(\)](#).

#### 32.96.4.3 `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\(\)](#).

#### 32.96.4.4 `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\(\)](#).

#### 32.96.4.5 `const HolderMap_T& stdair::LegDate::getHolderMap ( ) const` [inline]

Get the map of children holders.

Definition at line 65 of file [LegDate.hpp](#).

References [\\_holderMap](#).

#### 32.96.4.6 `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

<i>const</i>	<code>std::string&amp;</code> 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\(\)](#).

**32.96.4.7 LegCabin \* stdair::LegDate::getLegCabin ( const LegCabinKey & iLegCabinKey ) const**

Get a pointer on the [LegCabin](#) object corresponding to the given key.

**Note**

The [LegCabin](#) object can be inherited from, if needed. In that case, a `dynamic_cast<>` may be needed.

**Parameters**

<i>const</i>	<a href="#">LegCabinKey</a> & 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\(\)](#).

**32.96.4.8 const AirportCode\_T& stdair::LegDate::getOffPoint ( ) const [inline]**

Get the off point.

Definition at line 94 of file [LegDate.hpp](#).

References [\\_offPoint](#).

**32.96.4.9 const Date\_T& stdair::LegDate::getBoardingDate ( ) const [inline]**

Get the boarding date.

Definition at line 99 of file [LegDate.hpp](#).

References [\\_boardingDate](#).

**32.96.4.10 const Duration\_T& stdair::LegDate::getBoardingTime ( ) const [inline]**

Get the boarding time.

Definition at line 104 of file [LegDate.hpp](#).

References [\\_boardingTime](#).

**32.96.4.11 const Date\_T& stdair::LegDate::getOffDate ( ) const [inline]**

Get the off date.

Definition at line 109 of file [LegDate.hpp](#).

References [\\_offDate](#).

**32.96.4.12 const Duration\_T& stdair::LegDate::getOffTime ( ) const [inline]**

Get the off time.

Definition at line 114 of file [LegDate.hpp](#).

References [\\_offTime](#).

**32.96.4.13 const Duration\_T& stdair::LegDate::getElapsedTime ( ) const [inline]**

Get the elapsed time.

Definition at line 119 of file [LegDate.hpp](#).

References [\\_elapsedTime](#).

**32.96.4.14** `const Distance_T& stdair::LegDate::getDistance ( ) const` `[inline]`

Get the distance.

Definition at line 124 of file [LegDate.hpp](#).

References [\\_distance](#).

**32.96.4.15** `const CabinCapacity_T& stdair::LegDate::getCapacity ( ) const` `[inline]`

Get the leg capacity.

Definition at line 129 of file [LegDate.hpp](#).

References [\\_capacity](#).

**32.96.4.16** `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\(\)](#).

**32.96.4.17** `const Duration_T stdair::LegDate::getTimeOffset ( ) const`

Get the time off set between boarding and off points.

It is defined as being:  $\text{TimeOffset} = (\text{OffTime} - \text{BoardingTime}) + (\text{OffDate} - \text{BoardingDate}) * 24$

- [ElapsedTime](#).

Definition at line 88 of file [LegDate.cpp](#).

References [\\_boardingTime](#), [\\_elapsedTime](#), [\\_offTime](#), and [getDateOffset\(\)](#).

**32.96.4.18** `void stdair::LegDate::setOffPoint ( const AirportCode_T & iOffPoint )` `[inline]`

Set the off point.

Definition at line 148 of file [LegDate.hpp](#).

References [\\_offPoint](#).

**32.96.4.19** `void stdair::LegDate::setBoardingDate ( const Date_T & iBoardingDate )` `[inline]`

Set the boarding date.

Definition at line 153 of file [LegDate.hpp](#).

References [\\_boardingDate](#).

**32.96.4.20** `void stdair::LegDate::setBoardingTime ( const Duration_T & iBoardingTime )` `[inline]`

Set the boarding time.

Definition at line 158 of file [LegDate.hpp](#).

References [\\_boardingTime](#).

**32.96.4.21** `void stdair::LegDate::setOffDate ( const Date_T & iOffDate )` `[inline]`

Set the off date.

Definition at line 163 of file [LegDate.hpp](#).

References [\\_offDate](#).

32.96.4.22 void stdair::LegDate::setOffTime ( const Duration\_T & iOffTime ) [inline]

Set the off time.

Definition at line 168 of file [LegDate.hpp](#).

References [\\_offTime](#).

32.96.4.23 void stdair::LegDate::setElapsedTime ( const Duration\_T & iElapsedTime )

Set the elapsed time.

Definition at line 103 of file [LegDate.cpp](#).

References [\\_elapsedTime](#).

32.96.4.24 void stdair::LegDate::setOperatingAirlineCode ( const AirlineCode\_T & iAirlineCode ) [inline]

Set the operating airline code.

Definition at line 176 of file [LegDate.hpp](#).

References [\\_operatingAirlineCode](#).

32.96.4.25 void stdair::LegDate::setOperatingFlightNumber ( const FlightNumber\_T & iFlightNumber ) [inline]

Set the operating flight number.

Definition at line 181 of file [LegDate.hpp](#).

References [\\_operatingFlightNumber](#).

32.96.4.26 void stdair::LegDate::toStream ( std::ostream & ioOut ) const [inline],[virtual]

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Implements [stdair::BomAbstract](#).

Definition at line 194 of file [LegDate.hpp](#).

References [toString\(\)](#).

32.96.4.27 void stdair::LegDate::fromStream ( std::istream & ioIn ) [inline],[virtual]

Read a Business Object from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Implements [stdair::BomAbstract](#).

Definition at line 200 of file [LegDate.hpp](#).

32.96.4.28 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\(\)](#).

**32.96.4.29** `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\(\)](#).

**32.96.4.30** `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\(\)](#).

## 32.96.5 Friends And Related Function Documentation

**32.96.5.1** `template<typename BOM> friend class FacBom [friend]`

Definition at line 26 of file [LegDate.hpp](#).

**32.96.5.2** `template<typename BOM> friend class FacCloneBom [friend]`

Definition at line 27 of file [LegDate.hpp](#).

**32.96.5.3** `friend class FacBomManager [friend]`

Definition at line 28 of file [LegDate.hpp](#).

## 32.96.6 Member Data Documentation

**32.96.6.1** `Key_T stdair::LegDate::_key [protected]`

Primary key (origin airport).

Definition at line 231 of file [LegDate.hpp](#).

Referenced by [describeKey\(\)](#), [getBoardingPoint\(\)](#), and [getKey\(\)](#).

**32.96.6.2** `BomAbstract* stdair::LegDate::_parent [protected]`

Pointer on the parent class ([FlightDate](#)).

Definition at line 234 of file [LegDate.hpp](#).

Referenced by [getParent\(\)](#).

**32.96.6.3** `HolderMap_T stdair::LegDate::_holderMap [protected]`

Map holding the children ([LegCabin](#) objects).

Definition at line 237 of file [LegDate.hpp](#).

Referenced by [getHolderMap\(\)](#).

**32.96.6.4** `AirportCode_T stdair::LegDate::_offPoint [protected]`

Landing airport.

Definition at line 240 of file [LegDate.hpp](#).

Referenced by [getOffPoint\(\)](#), and [setOffPoint\(\)](#).

**32.96.6.5 Date\_T stdair::LegDate::\_boardingDate** [protected]

Boarding date.

Definition at line 243 of file [LegDate.hpp](#).

Referenced by [getBoardingDate\(\)](#), [getDateOffset\(\)](#), and [setBoardingDate\(\)](#).

**32.96.6.6 Duration\_T stdair::LegDate::\_boardingTime** [protected]

Boarding time.

Definition at line 246 of file [LegDate.hpp](#).

Referenced by [getBoardingTime\(\)](#), [getTimeOffset\(\)](#), and [setBoardingTime\(\)](#).

**32.96.6.7 Date\_T stdair::LegDate::\_offDate** [protected]

Landing date.

Definition at line 249 of file [LegDate.hpp](#).

Referenced by [getDateOffset\(\)](#), [getOffDate\(\)](#), and [setOffDate\(\)](#).

**32.96.6.8 Duration\_T stdair::LegDate::\_offTime** [protected]

Landing time.

Definition at line 252 of file [LegDate.hpp](#).

Referenced by [getOffTime\(\)](#), [getTimeOffset\(\)](#), and [setOffTime\(\)](#).

**32.96.6.9 Duration\_T stdair::LegDate::\_elapsedTime** [protected]

Trip elapsed time.

Definition at line 255 of file [LegDate.hpp](#).

Referenced by [getElapsedTime\(\)](#), [getTimeOffset\(\)](#), and [setElapsedTime\(\)](#).

**32.96.6.10 Distance\_T stdair::LegDate::\_distance** [protected]

Trip distance.

Definition at line 258 of file [LegDate.hpp](#).

Referenced by [getDistance\(\)](#).

**32.96.6.11 CabinCapacity\_T stdair::LegDate::\_capacity** [protected]

Aggregated capacity for all the leg-cabins.

Definition at line 261 of file [LegDate.hpp](#).

Referenced by [getCapacity\(\)](#).

**32.96.6.12 AirlineCode\_T stdair::LegDate::\_operatingAirlineCode** [protected]

Operating airline code.

Definition at line 264 of file [LegDate.hpp](#).

Referenced by [describeRoutingKey\(\)](#), and [setOperatingAirlineCode\(\)](#).

**32.96.6.13 FlightNumber\_T stdair::LegDate::\_operatingFlightNumber** [protected]

Operating flight number.

Definition at line 267 of file [LegDate.hpp](#).



Referenced by [describeRoutingKey\(\)](#), and [setOperatingFlightNumber\(\)](#).

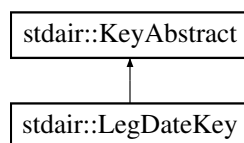
The documentation for this class was generated from the following files:

- [stdair/bom/LegDate.hpp](#)
- [stdair/bom/LegDate.cpp](#)

### 32.97 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

#### 32.97.1 Detailed Description

Key of a given leg-date, made of an origin airport.

Definition at line 16 of file [LegDateKey.hpp](#).

#### 32.97.2 Constructor & Destructor Documentation

##### 32.97.2.1 stdair::LegDateKey::LegDateKey ( const [AirportCode\\_T](#) & iBoardingPoint )

Constructor.

Definition at line 19 of file [LegDateKey.cpp](#).

##### 32.97.2.2 stdair::LegDateKey::LegDateKey ( const [LegDateKey](#) & iKey )

Default copy constructor.

Definition at line 24 of file [LegDateKey.cpp](#).

##### 32.97.2.3 stdair::LegDateKey::~~LegDateKey ( )

Destructor.

Definition at line 29 of file [LegDateKey.cpp](#).

## 32.97.3 Member Function Documentation

## 32.97.3.1 const AirportCode\_T&amp; stdair::LegDateKey::getBoardingPoint ( ) const [inline]

Get the boarding point.

Definition at line 34 of file [LegDateKey.hpp](#).

Referenced by [stdair::LegDate::getBoardingPoint\(\)](#).

## 32.97.3.2 void stdair::LegDateKey::toStream ( std::ostream &amp; ioOut ) const [virtual]

Dump a Business Object Key into an output stream.

## Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 33 of file [LegDateKey.cpp](#).

References [toString\(\)](#).

## 32.97.3.3 void stdair::LegDateKey::fromStream ( std::istream &amp; iIn ) [virtual]

Read a Business Object Key from an input stream.

## Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 38 of file [LegDateKey.cpp](#).

## 32.97.3.4 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](#)

## 32.98 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 &i↵  
ToBeLogged)

### Static Public Member Functions

- static [Logger](#) & [instance](#) ()

### Friends

- class [FacSupervisor](#)  
*Friend classes.*
- class [STDAIR\\_Service](#)

### 32.98.1 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](#).

### 32.98.2 Member Function Documentation

**32.98.2.1** `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](#).

**32.98.2.2** `Logger & stdair::Logger::instance ( ) [static]`

Return the static [Logger](#) instance.

Definition at line 48 of file [Logger.cpp](#).

### 32.98.3 Friends And Related Function Documentation

**32.98.3.1** `friend class FacSupervisor [friend]`

Friend classes.

Definition at line 50 of file [Logger.hpp](#).

**32.98.3.2** `friend class STDAIR_Service [friend]`

Definition at line 51 of file [Logger.hpp](#).

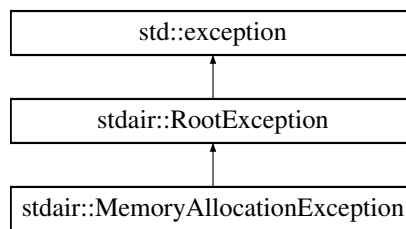
The documentation for this class was generated from the following files:

- [stdair/service/Logger.hpp](#)
- [stdair/service/Logger.cpp](#)

## 32.99 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](#)

#### 32.99.1 Detailed Description

Memory allocation.

Definition at line 89 of file [stdair\\_exceptions.hpp](#).

#### 32.99.2 Constructor & Destructor Documentation

32.99.2.1 `stdair::MemoryAllocationException::MemoryAllocationException ( const std::string &iWhat )` `[inline]`

Constructor.

Definition at line 92 of file [stdair\\_exceptions.hpp](#).

#### 32.99.3 Member Function Documentation

32.99.3.1 `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\(\)](#).

#### 32.99.4 Member Data Documentation

32.99.4.1 `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\(\)](#).

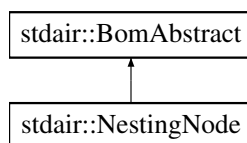
The documentation for this class was generated from the following file:

- [stdair/stdair\\_exceptions.hpp](#)

### 32.100 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

- template<typename BOM >  
class [FacBom](#)
- class [FacBomManager](#)
- class [boost::serialization::access](#)

#### 32.100.1 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](#).

#### 32.100.2 Member Typedef Documentation

##### 32.100.2.1 typedef [NestingNodeKey](#) stdair::NestingNode::Key\_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 39 of file [NestingNode.hpp](#).

## 32.100.3 Constructor &amp; Destructor Documentation

## 32.100.3.1 stdair::NestingNode::NestingNode ( const Key\_T &amp; iKey ) [protected]

Main constructor.

Definition at line 31 of file [NestingNode.cpp](#).

## 32.100.3.2 stdair::NestingNode::~~NestingNode ( ) [protected],[virtual]

Destructor.

Definition at line 35 of file [NestingNode.cpp](#).

## 32.100.4 Member Function Documentation

## 32.100.4.1 const Key\_T&amp; stdair::NestingNode::getKey ( ) const [inline]

Get the policy key.

Definition at line 44 of file [NestingNode.hpp](#).

## 32.100.4.2 BomAbstract\* const stdair::NestingNode::getParent ( ) const [inline]

Get the parent object.

Definition at line 49 of file [NestingNode.hpp](#).

## 32.100.4.3 const HolderMap\_T&amp; stdair::NestingNode::getHolderMap ( ) const [inline]

Get the map of children holders.

Definition at line 56 of file [NestingNode.hpp](#).

Referenced by [stdair::FacBomManager::resetYieldBasedNestingStructure\(\)](#).

## 32.100.4.4 const Yield\_T&amp; stdair::NestingNode::getYield ( ) const [inline]

Getter for the yield.

Definition at line 61 of file [NestingNode.hpp](#).

## 32.100.4.5 void stdair::NestingNode::setYield ( const Yield\_T &amp; iYield ) [inline]

Setter for the yield.

Definition at line 68 of file [NestingNode.hpp](#).

Referenced by [stdair::FacBomManager::resetYieldBasedNestingStructure\(\)](#).

## 32.100.4.6 void stdair::NestingNode::toStream ( std::ostream &amp; ioOut ) const [inline],[virtual]

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Implements [stdair::BomAbstract](#).

Definition at line 80 of file [NestingNode.hpp](#).

References [toString\(\)](#).

## 32.100.4.7 void stdair::NestingNode::fromStream ( std::istream &amp; ioln ) [inline],[virtual]

Read a Business Object from an input stream.

## Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Implements [stdair::BomAbstract](#).

Definition at line 89 of file [NestingNode.hpp](#).

32.100.4.8 `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\(\)](#).

32.100.4.9 `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 [stdair::FacBomManager::resetYieldBasedNestingStructure\(\)](#), and [toString\(\)](#).

32.100.4.10 `template<class Archive > void stdair::NestingNode::serialize ( Archive & ar, const unsigned int iFileVersion )`

Serialisation.

## 32.100.5 Friends And Related Function Documentation

32.100.5.1 `template<typename BOM > friend class FacBom` `[friend]`

Definition at line 30 of file [NestingNode.hpp](#).

32.100.5.2 `friend class FacBomManager` `[friend]`

Definition at line 31 of file [NestingNode.hpp](#).

32.100.5.3 `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:

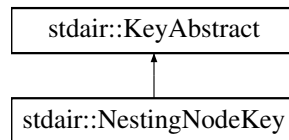
- [stdair/bom/NestingNode.hpp](#)
- [stdair/bom/NestingNode.cpp](#)

## 32.101 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](#)

### 32.101.1 Detailed Description

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

Definition at line 26 of file [NestingNodeKey.hpp](#).

### 32.101.2 Constructor & Destructor Documentation

#### 32.101.2.1 stdair::NestingNodeKey::NestingNodeKey ( const NestingNodeCode\_T & iNestingNodeCode )

Constructor.

Definition at line 28 of file [NestingNodeKey.cpp](#).

#### 32.101.2.2 stdair::NestingNodeKey::NestingNodeKey ( const NestingNodeKey & iNestingNodeKey )

Copy constructor.

Definition at line 23 of file [NestingNodeKey.cpp](#).

#### 32.101.2.3 stdair::NestingNodeKey::~~NestingNodeKey ( )

Destructor.

Definition at line 33 of file [NestingNodeKey.cpp](#).

### 32.101.3 Member Function Documentation

#### 32.101.3.1 const NestingNodeCode\_T& stdair::NestingNodeKey::getNestingNodeCode ( ) const [inline]

Get the policy code.

Definition at line 56 of file [NestingNodeKey.hpp](#).



32.101.3.2 `void stdair::NestingNodeKey::toStream ( std::ostream & ioOut ) const` [virtual]

Dump a Business Object Key into an output stream.

## Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 37 of file [NestingNodeKey.cpp](#).

References [toString\(\)](#).

**32.101.3.3** void stdair::NestingNodeKey::fromStream ( std::istream & *ioIn* ) [virtual]

Read a Business Object Key from an input stream.

## Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 42 of file [NestingNodeKey.cpp](#).

**32.101.3.4** 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\(\)](#).

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

Serialisation.

Definition at line 68 of file [NestingNodeKey.cpp](#).

**32.101.4 Friends And Related Function Documentation**

**32.101.4.1** 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:

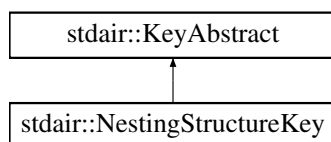
- [stdair/bom/NestingNodeKey.hpp](#)
- [stdair/bom/NestingNodeKey.cpp](#)

**32.102 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](#)

### 32.102.1 Detailed Description

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

Definition at line 26 of file [NestingStructureKey.hpp](#).

### 32.102.2 Constructor & Destructor Documentation

#### 32.102.2.1 stdair::NestingStructureKey::NestingStructureKey ( const NestingStructureCode\_T & iNestingStructureCode )

Constructor.

Definition at line 28 of file [NestingStructureKey.cpp](#).

#### 32.102.2.2 stdair::NestingStructureKey::NestingStructureKey ( const NestingStructureKey & iNestingStructureKey )

Copy constructor.

Definition at line 23 of file [NestingStructureKey.cpp](#).

#### 32.102.2.3 stdair::NestingStructureKey::~~NestingStructureKey ( )

Destructor.

Definition at line 33 of file [NestingStructureKey.cpp](#).

### 32.102.3 Member Function Documentation

#### 32.102.3.1 const NestingStructureCode\_T& stdair::NestingStructureKey::getNestingStructureCode ( ) const [inline]

Get the nesting structure code.

Definition at line 56 of file [NestingStructureKey.hpp](#).

32.102.3.2 void stdair::NestingStructureKey::toStream ( std::ostream & *ioOut* ) const [virtual]

Dump a Business Object Key into an output stream.

## Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 37 of file [NestingStructureKey.cpp](#).

References [toString\(\)](#).

**32.102.3.3** `void stdair::NestingStructureKey::fromStream ( std::istream & ioln ) [virtual]`

Read a Business Object Key from an input stream.

## Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 42 of file [NestingStructureKey.cpp](#).

**32.102.3.4** `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\(\)](#).

**32.102.3.5** `template<class Archive > void stdair::NestingStructureKey::serialize ( Archive & ar, const unsigned int iFileVersion )`

Serialisation.

Definition at line 68 of file [NestingStructureKey.cpp](#).

## 32.102.4 Friends And Related Function Documentation

**32.102.4.1** `friend class boost::serialization::access [friend]`

Definition at line 27 of file [NestingStructureKey.hpp](#).

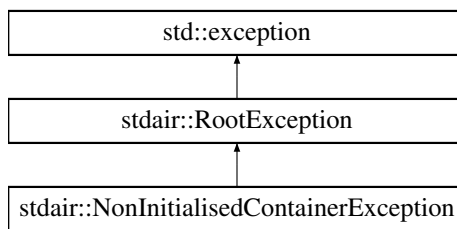
The documentation for this struct was generated from the following files:

- [stdair/bom/NestingStructureKey.hpp](#)
- [stdair/bom/NestingStructureKey.cpp](#)

## 32.103 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](#)

#### 32.103.1 Detailed Description

Non initialised container.

Definition at line 73 of file [stdair\\_exceptions.hpp](#).

#### 32.103.2 Constructor & Destructor Documentation

32.103.2.1 `stdair::NonInitialisedContainerException::NonInitialisedContainerException ( const std::string & iWhat )`  
`[inline]`

Constructor.

Definition at line 76 of file [stdair\\_exceptions.hpp](#).

#### 32.103.3 Member Function Documentation

32.103.3.1 `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\(\)](#).

#### 32.103.4 Member Data Documentation

32.103.4.1 `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\(\)](#).

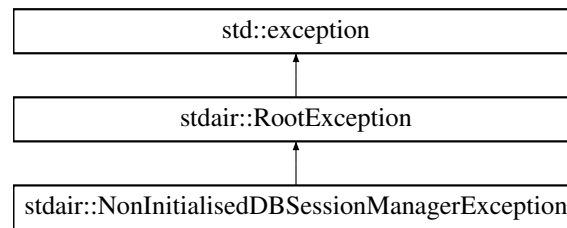
The documentation for this class was generated from the following file:

- [stdair/stdair\\_exceptions.hpp](#)

### 32.104 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](#)

#### 32.104.1 Detailed Description

Non initialised database session.

Definition at line 188 of file [stdair\\_exceptions.hpp](#).

#### 32.104.2 Constructor & Destructor Documentation

**32.104.2.1** `stdair::NonInitialisedDBSessionManagerException::NonInitialisedDBSessionManagerException ( const std::string &iWhat ) [inline]`

Constructor.

Definition at line 191 of file [stdair\\_exceptions.hpp](#).

#### 32.104.3 Member Function Documentation

**32.104.3.1** `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\(\)](#).

#### 32.104.4 Member Data Documentation

**32.104.4.1** `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\(\)](#).

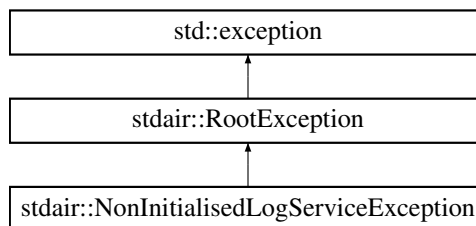
The documentation for this class was generated from the following file:

- [stdair/stdair\\_exceptions.hpp](#)

## 32.105 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](#)

#### 32.105.1 Detailed Description

Non initialised log service.

Definition at line 57 of file [stdair\\_exceptions.hpp](#).

#### 32.105.2 Constructor & Destructor Documentation

**32.105.2.1** `stdair::NonInitialisedLogServiceException::NonInitialisedLogServiceException ( const std::string & iWhat )`  
`[inline]`

Constructor.

Definition at line 60 of file [stdair\\_exceptions.hpp](#).

#### 32.105.3 Member Function Documentation

**32.105.3.1** `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\(\)](#).



### 32.105.4 Member Data Documentation

#### 32.105.4.1 `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\(\)](#).

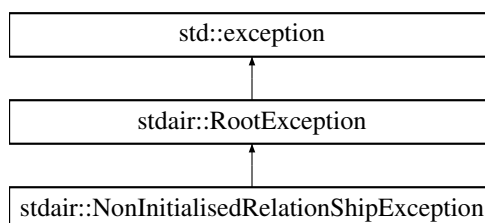
The documentation for this class was generated from the following file:

- [stdair/stdair\\_exceptions.hpp](#)

## 32.106 `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](#)

### 32.106.1 Detailed Description

Non initialised relationship.

Definition at line 81 of file [stdair\\_exceptions.hpp](#).

### 32.106.2 Constructor & Destructor Documentation

#### 32.106.2.1 `stdair::NonInitialisedRelationshipException::NonInitialisedRelationshipException ( const std::string & iWhat )` `[inline]`

Constructor.

Definition at line 84 of file [stdair\\_exceptions.hpp](#).

### 32.106.3 Member Function Documentation

**32.106.3.1** `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\(\)](#).

#### 32.106.4 Member Data Documentation

**32.106.4.1** `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\(\)](#).

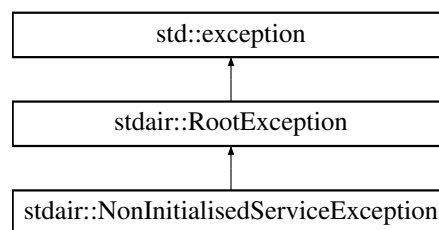
The documentation for this class was generated from the following file:

- [stdair/stdair\\_exceptions.hpp](#)

### 32.107 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`

#### 32.107.1 Detailed Description

Non initialised service.

Definition at line 65 of file [stdair\\_exceptions.hpp](#).

#### 32.107.2 Constructor & Destructor Documentation

32.107.2.1 `stdair::NonInitialisedServiceException::NonInitialisedServiceException ( const std::string & iWhat ) [inline]`

Constructor.

Definition at line 68 of file [stdair\\_exceptions.hpp](#).

### 32.107.3 Member Function Documentation

32.107.3.1 `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\(\)](#).

### 32.107.4 Member Data Documentation

32.107.4.1 `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\(\)](#).

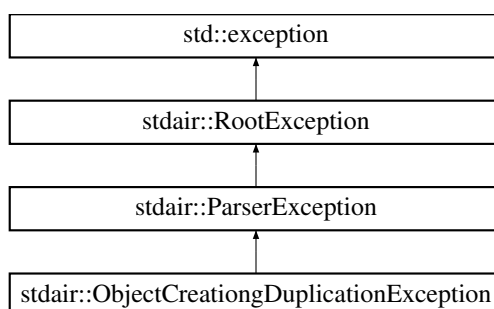
The documentation for this class was generated from the following file:

- [stdair/stdair\\_exceptions.hpp](#)

## 32.108 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](#)

## 32.108.1 Detailed Description

Duplicated object.

Definition at line 157 of file [stdair\\_exceptions.hpp](#).

## 32.108.2 Constructor &amp; Destructor Documentation

32.108.2.1 `stdair::ObjectCreationgDuplicationException::ObjectCreationgDuplicationException ( const std::string & iWhat )`  
`[inline]`

Constructor.

Definition at line 160 of file [stdair\\_exceptions.hpp](#).

## 32.108.3 Member Function Documentation

32.108.3.1 `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\(\)](#).

## 32.108.4 Member Data Documentation

32.108.4.1 `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\(\)](#).

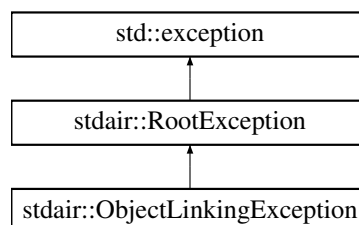
The documentation for this class was generated from the following file:

- [stdair/stdair\\_exceptions.hpp](#)

## 32.109 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`

### 32.109.1 Detailed Description

Object link.

Definition at line 97 of file [stdair\\_exceptions.hpp](#).

### 32.109.2 Constructor & Destructor Documentation

#### 32.109.2.1 `stdair::ObjectLinkingException::ObjectLinkingException ( const std::string & iWhat ) [inline]`

Constructor.

Definition at line 100 of file [stdair\\_exceptions.hpp](#).

### 32.109.3 Member Function Documentation

#### 32.109.3.1 `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\(\)](#).

### 32.109.4 Member Data Documentation

#### 32.109.4.1 `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\(\)](#).

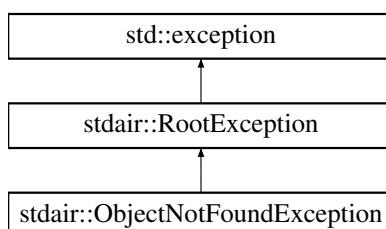
The documentation for this class was generated from the following file:

- [stdair/stdair\\_exceptions.hpp](#)

## 32.110 `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](#)

### 32.110.1 Detailed Description

Not found object.

Definition at line 165 of file [stdair\\_exceptions.hpp](#).

### 32.110.2 Constructor & Destructor Documentation

32.110.2.1 `stdair::ObjectNotFoundException::ObjectNotFoundException ( const std::string & iWhat )` `[inline]`

Constructor.

Definition at line 168 of file [stdair\\_exceptions.hpp](#).

### 32.110.3 Member Function Documentation

32.110.3.1 `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\(\)](#).

### 32.110.4 Member Data Documentation

32.110.4.1 `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\(\)](#).

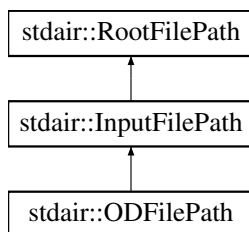
The documentation for this class was generated from the following file:

- [stdair/stdair\\_exceptions.hpp](#)

## 32.111 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

#### 32.111.1 Detailed Description

OD input file.

Definition at line 76 of file [stdair\\_file.hpp](#).

#### 32.111.2 Constructor & Destructor Documentation

32.111.2.1 [stdair::ODFilePath::ODFilePath](#) ( const [Filename\\_T](#) &*iFilename* ) [\[inline\]](#), [\[explicit\]](#)

Constructor.

Definition at line 81 of file [stdair\\_file.hpp](#).

#### 32.111.3 Member Function Documentation

32.111.3.1 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\(\)](#).

#### 32.111.4 Member Data Documentation

32.111.4.1 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:

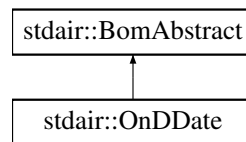
- [stdair/stdair\\_file.hpp](#)

## 32.112 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

- `template<typename BOM >`  
class [FacBom](#)
- `template<typename BOM >`  
class [FacCloneBom](#)
- class [FacBomManager](#)
- class [boost::serialization::access](#)

### 32.112.1 Detailed Description

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

Definition at line 33 of file [OnDDate.hpp](#).

### 32.112.2 Member Typedef Documentation

#### 32.112.2.1 `typedef OnDDateKey stdair::OnDDate::Key_T`

Definition allowing to retrieve the associated BOM key type.

Definition at line 44 of file [OnDDate.hpp](#).

### 32.112.3 Constructor & Destructor Documentation

#### 32.112.3.1 `stdair::OnDDate::OnDDate ( const Key_T & iKey ) [protected]`

Main constructor.

Definition at line 28 of file [OnDDate.cpp](#).

#### 32.112.3.2 `stdair::OnDDate::~~OnDDate ( ) [protected],[virtual]`

Destructor.

Definition at line 33 of file [OnDDate.cpp](#).

### 32.112.4 Member Function Documentation

#### 32.112.4.1 `const Key_T& stdair::OnDDate::getKey ( ) const [inline]`

Get the O&D date key.

Definition at line 50 of file [OnDDate.hpp](#).

References [\\_key](#).

#### 32.112.4.2 `BomAbstract* const stdair::OnDDate::getParent ( ) const [inline]`

Get the parent object.

Definition at line 55 of file [OnDDate.hpp](#).

References [\\_parent](#).

Referenced by [getAirlineCode\(\)](#).

#### 32.112.4.3 `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\(\)](#).

**32.112.4.4** `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\(\)](#).

**32.112.4.5** `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\(\)](#).

**32.112.4.6** `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\(\)](#).

**32.112.4.7** `const HolderMap_T& stdair::OnDDate::getHolderMap ( ) const` `[inline]`

Get the map of children holders.

Definition at line 87 of file [OnDDate.hpp](#).

References [\\_holderMap](#).

**32.112.4.8** `const StringDemandStructMap_T& stdair::OnDDate::getDemandInfoMap ( ) const` `[inline]`

Get the map of demand information.

Definition at line 94 of file [OnDDate.hpp](#).

References [\\_classPathDemandMap](#).

**32.112.4.9** `const CabinForecastMap_T& stdair::OnDDate::getTotalForecastMap ( ) const` `[inline]`

Get the map of total forecast.

Definition at line 101 of file [OnDDate.hpp](#).

References [\\_cabinForecastMap](#).

**32.112.4.10** `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](#).

**32.112.4.11** `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](#).

**32.112.4.12** `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\(\)](#).

**32.112.4.13** `void stdair::OnDDate::setDemandInformation ( const CabinClassPairList_T & iCabinClassPairList, const YieldDemandPair_T & iYieldDemandPair )`

Set demand information.

Definition at line 53 of file [OnDDate.cpp](#).

References [\\_classPathDemandMap](#), and [\\_stringCabinClassPairListMap](#).

**32.112.4.14** `void stdair::OnDDate::setTotalForecast ( const CabinCode_T & iCabinCode, const WTPDemandPair_T & iWTPDemandPair )`

Set forecast information per cabin.

Definition at line 76 of file [OnDDate.cpp](#).

References [\\_cabinForecastMap](#).

**32.112.4.15** `void stdair::OnDDate::toStream ( std::ostream & ioOut ) const` `[inline],[virtual]`

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Implements [stdair::BomAbstract](#).

Definition at line 147 of file [OnDDate.hpp](#).

References [toString\(\)](#).

**32.112.4.16** `void stdair::OnDDate::fromStream ( std::istream & ioIn )` `[inline],[virtual]`

Read a Business Object from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Implements [stdair::BomAbstract](#).

Definition at line 156 of file [OnDDate.hpp](#).

**32.112.4.17** `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\(\)](#).

32.112.4.18 `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\(\)](#).

32.112.4.19 `template<class Archive > void stdair::OnDDate::serialize ( Archive & ar, const unsigned int iFileVersion )`

Serialisation.

### 32.112.5 Friends And Related Function Documentation

32.112.5.1 `template<typename BOM > friend class FacBom [friend]`

Definition at line 34 of file [OnDDate.hpp](#).

32.112.5.2 `template<typename BOM > friend class FacCloneBom [friend]`

Definition at line 35 of file [OnDDate.hpp](#).

32.112.5.3 `friend class FacBomManager [friend]`

Definition at line 36 of file [OnDDate.hpp](#).

32.112.5.4 `friend class boost::serialization::access [friend]`

Definition at line 37 of file [OnDDate.hpp](#).

### 32.112.6 Member Data Documentation

32.112.6.1 `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\(\)](#).

32.112.6.2 `BomAbstract* stdair::OnDDate::_parent [protected]`

Pointer on the parent class ([Inventory](#)).

Definition at line 222 of file [OnDDate.hpp](#).

Referenced by [getParent\(\)](#).

32.112.6.3 `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\(\)](#).

32.112.6.4 `StringDemandStructMap_T stdair::OnDDate::_classPathDemandMap [protected]`

O&D demand information.

Definition at line 232 of file [OnDDate.hpp](#).

Referenced by [getDemandInfoMap\(\)](#), and [setDemandInformation\(\)](#).

#### 32.112.6.5 `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\(\)](#).

#### 32.112.6.6 `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:

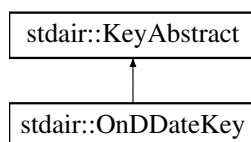
- [stdair/bom/OnDDate.hpp](#)
- [stdair/bom/OnDDate.cpp](#)

### 32.113 `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](#)

### 32.113.1 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](#).

### 32.113.2 Constructor & Destructor Documentation

#### 32.113.2.1 stdair::OnDDateKey::OnDDateKey ( const OnDStringList\_T & iOnDStringList )

Constructor.

Definition at line 33 of file [OnDDateKey.cpp](#).

#### 32.113.2.2 stdair::OnDDateKey::OnDDateKey ( const OnDDateKey & iKey )

Copy constructor.

Definition at line 38 of file [OnDDateKey.cpp](#).

#### 32.113.2.3 stdair::OnDDateKey::~~OnDDateKey ( )

Destructor.

Definition at line 43 of file [OnDDateKey.cpp](#).

### 32.113.3 Member Function Documentation

#### 32.113.3.1 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\(\)](#).

#### 32.113.3.2 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\(\)](#).

#### 32.113.3.3 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\(\)](#).

#### 32.113.3.4 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\(\)](#).

32.113.3.5 `void stdair::OnDDateKey::toStream ( std::ostream & ioOut ) const` [virtual]

Dump a Business Object Key into an output stream.

## Parameters

<i>ostream</i> &	the output stream.
------------------	--------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 68 of file [OnDDateKey.cpp](#).

References [toString\(\)](#).

**32.113.3.6** void stdair::OnDDateKey::fromStream ( std::istream & *ioln* ) [virtual]

Read a Business Object Key from an input stream.

## Parameters

<i>istream</i> &	the input stream.
------------------	-------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 73 of file [OnDDateKey.cpp](#).

**32.113.3.7** 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\(\)](#).

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

Serialisation.

Definition at line 102 of file [OnDDateKey.cpp](#).

**32.113.4 Friends And Related Function Documentation**

**32.113.4.1** friend class boost::serialization::access [friend]

Definition at line 24 of file [OnDDateKey.hpp](#).

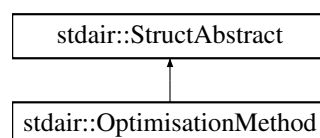
The documentation for this struct was generated from the following files:

- [stdair/bom/OnDDateKey.hpp](#)
- [stdair/bom/OnDDateKey.cpp](#)

**32.114 stdair::OptimisationMethod Struct Reference**

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

Inheritance diagram for stdair::OptimisationMethod:





## Public Types

- 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](#) ()

## 32.114.1 Detailed Description

Enumeration of Optimisation methods.

Definition at line 15 of file [OptimisationMethod.hpp](#).

## 32.114.2 Member Enumeration Documentation

## 32.114.2.1 enum stdair::OptimisationMethod::EN\_OptimisationMethod

Enumerator

**[LEG\\_BASED\\_MC](#)**  
**[LEG\\_BASED\\_EMSR\\_B](#)**  
**[LAST\\_VALUE](#)**

Definition at line 17 of file [OptimisationMethod.hpp](#).

## 32.114.3 Constructor &amp; Destructor Documentation

32.114.3.1 stdair::OptimisationMethod::OptimisationMethod ( const [EN\\_OptimisationMethod](#) & *iOptimisationMethod* )

Constructor.

Definition at line 36 of file [OptimisationMethod.cpp](#).

32.114.3.2 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](#).

32.114.3.3 `stdair::OptimisationMethod::OptimisationMethod ( const OptimisationMethod & iOptimisationMethod )`

Default copy constructor.

Definition at line 30 of file [OptimisationMethod.cpp](#).

#### 32.114.4 Member Function Documentation

32.114.4.1 `const std::string & stdair::OptimisationMethod::getLabel ( const EN_OptimisationMethod & iMethod )`  
[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](#).

32.114.4.2 `char stdair::OptimisationMethod::getMethodLabel ( const EN_OptimisationMethod & iMethod )` [static]

Get the label as a single char (e.g., 'M' or 'E').

Definition at line 64 of file [OptimisationMethod.cpp](#).

32.114.4.3 `std::string stdair::OptimisationMethod::getMethodLabelAsString ( const EN_OptimisationMethod & iMethod )`  
[static]

Get the label as a string of a single char (e.g., "M" or "E").

Definition at line 70 of file [OptimisationMethod.cpp](#).

32.114.4.4 `std::string stdair::OptimisationMethod::describeLabels ( )` [static]

List the labels.

Definition at line 77 of file [OptimisationMethod.cpp](#).

References [LAST\\_VALUE](#).

Referenced by [OptimisationMethod\(\)](#).

32.114.4.5 `OptimisationMethod::EN_OptimisationMethod stdair::OptimisationMethod::getMethod ( ) const`

Get the enumerated value.

Definition at line 89 of file [OptimisationMethod.cpp](#).

Referenced by [stdair::AirlineFeature::getOptimisationMethod\(\)](#).

32.114.4.6 `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](#).

32.114.4.7 `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](#).

32.114.4.8 `bool stdair::OptimisationMethod::operator== ( const EN_OptimisationMethod & iMethod ) const`

Comparison operator.

Definition at line 109 of file [OptimisationMethod.cpp](#).

32.114.4.9 `void stdair::StructAbstract::toStream ( std::ostream & ioOut ) const` `[inline],[inherited]`

Dump a Business Object into an output stream.

## Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Definition at line 29 of file [StructAbstract.hpp](#).

References [stdair::StructAbstract::describe\(\)](#).

32.114.4.10 virtual void stdair::StructAbstract::fromStream ( std::istream & *ioln* ) [inline], [virtual],  
[inherited]

Read a Business Object from an input stream.

## Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented in [stdair::BookingRequestStruct](#), [stdair::EventStruct](#), [stdair::TravelSolutionStruct](#), [stdair::VirtualClassStruct](#), [stdair::ConfigHolderStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::ProgressStatusSet](#), [stdair::YieldRange](#), [stdair::CancellationStruct](#), [stdair::AirlineStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::FFDisutilityCurveHolderStruct](#), [stdair::FRAT5CurveHolderStruct](#), and [stdair::BreakPointStruct](#).

Definition at line 38 of file [StructAbstract.hpp](#).

Referenced by [operator>>\(\)](#).

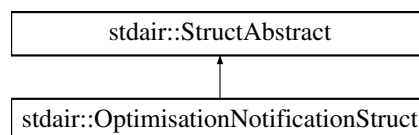
The documentation for this struct was generated from the following files:

- [stdair/basic/OptimisationMethod.hpp](#)
- [stdair/basic/OptimisationMethod.cpp](#)

## 32.115 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](#) & [getPreferredDepartureDate](#) () 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](#) ()

### 32.115.1 Detailed Description

Structure holding the elements of a optimisation notification.

Definition at line 19 of file [OptimisationNotificationStruct.hpp](#).

### 32.115.2 Constructor & Destructor Documentation

32.115.2.1 `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](#).

32.115.2.2 `stdair::OptimisationNotificationStruct::OptimisationNotificationStruct ( const OptimisationNotificationStruct &iOptimisationNotification )`

Copy constructor.

Definition at line 20 of file [OptimisationNotificationStruct.cpp](#).

32.115.2.3 `stdair::OptimisationNotificationStruct::~~OptimisationNotificationStruct ( )`

Destructor.

Definition at line 64 of file [OptimisationNotificationStruct.cpp](#).

### 32.115.3 Member Function Documentation

32.115.3.1 `const AirportCode\_T &stdair::OptimisationNotificationStruct::getOrigin ( ) const` `[inline]`

Get the notificationed origin.

Definition at line 23 of file [OptimisationNotificationStruct.hpp](#).

32.115.3.2 `const AirportCode\_T &stdair::OptimisationNotificationStruct::getDestination ( ) const` `[inline]`

Get the notificationed destination.

Definition at line 28 of file [OptimisationNotificationStruct.hpp](#).

32.115.3.3 `const CityCode\_T &stdair::OptimisationNotificationStruct::getPOS ( ) const` `[inline]`

Get the point-of-sale.

Definition at line 33 of file [OptimisationNotificationStruct.hpp](#).

**32.115.3.4** `const Date_T& stdair::OptimisationNotificationStruct::getPreferedDepartureDate ( ) const [inline]`

Get the notificationed departure date.

Definition at line 38 of file [OptimisationNotificationStruct.hpp](#).

**32.115.3.5** `const DateTime_T& stdair::OptimisationNotificationStruct::getNotificationDateTime ( ) const [inline]`

Get the notification datetime.

Definition at line 43 of file [OptimisationNotificationStruct.hpp](#).

**32.115.3.6** `const CabinCode_T& stdair::OptimisationNotificationStruct::getPreferredCabin ( ) const [inline]`

Get the preferred cabin.

Definition at line 48 of file [OptimisationNotificationStruct.hpp](#).

**32.115.3.7** `const NbOfSeats_T& stdair::OptimisationNotificationStruct::getPartySize ( ) const [inline]`

Get the party size.

Definition at line 53 of file [OptimisationNotificationStruct.hpp](#).

**32.115.3.8** `const ChannelLabel_T& stdair::OptimisationNotificationStruct::getOptimisationChannel ( ) const [inline]`

Get the reservation channel.

Definition at line 58 of file [OptimisationNotificationStruct.hpp](#).

**32.115.3.9** `const TripType_T& stdair::OptimisationNotificationStruct::getTripType ( ) const [inline]`

Get the trip type.

Definition at line 63 of file [OptimisationNotificationStruct.hpp](#).

**32.115.3.10** `const DayDuration_T& stdair::OptimisationNotificationStruct::getStayDuration ( ) const [inline]`

Get the duration of stay.

Definition at line 68 of file [OptimisationNotificationStruct.hpp](#).

**32.115.3.11** `const FrequentFlyer_T& stdair::OptimisationNotificationStruct::getFrequentFlyerType ( ) const [inline]`

Get the frequent flyer type.

Definition at line 73 of file [OptimisationNotificationStruct.hpp](#).

**32.115.3.12** `const Duration_T& stdair::OptimisationNotificationStruct::getPreferredDepartureTime ( ) const [inline]`

Get the preferred departure time.

Definition at line 78 of file [OptimisationNotificationStruct.hpp](#).

**32.115.3.13** `const WTP_T& stdair::OptimisationNotificationStruct::getWTP ( ) const [inline]`

Get the willingness-to-pay.

Definition at line 83 of file [OptimisationNotificationStruct.hpp](#).

**32.115.3.14** `const PriceValue_T& stdair::OptimisationNotificationStruct::getValueOfTime ( ) const [inline]`

Get the value of time.

Definition at line 88 of file [OptimisationNotificationStruct.hpp](#).

32.115.3.15 `void stdair::OptimisationNotificationStruct::toStream ( std::ostream & ioOut ) const`

Dump a Business Object into an output stream.

Parameters

<code>ostream&amp;</code>	the output stream.
---------------------------	--------------------

Definition at line 68 of file [OptimisationNotificationStruct.cpp](#).

References [describe\(\)](#).

32.115.3.16 `void stdair::OptimisationNotificationStruct::fromStream ( std::istream & ioln ) [virtual]`

Read a Business Object from an input stream.

Parameters

<code>istream&amp;</code>	the input stream.
---------------------------	-------------------

Reimplemented from [stdair::StructAbstract](#).

Definition at line 73 of file [OptimisationNotificationStruct.cpp](#).

32.115.3.17 `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\(\)](#).

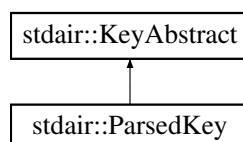
The documentation for this struct was generated from the following files:

- [stdair/bom/OptimisationNotificationStruct.hpp](#)
- [stdair/bom/OptimisationNotificationStruct.cpp](#)

## 32.116 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 &ioln`)

- `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`

#### 32.116.1 Detailed Description

Structure which holds the results/keys after the parsing.

Definition at line 22 of file [ParsedKey.hpp](#).

#### 32.116.2 Constructor & Destructor Documentation

##### 32.116.2.1 `stdair::ParsedKey::ParsedKey ( )`

Definition at line 41 of file [ParsedKey.cpp](#).

##### 32.116.2.2 `stdair::ParsedKey::~~ParsedKey ( )`

Definition at line 47 of file [ParsedKey.cpp](#).

#### 32.116.3 Member Function Documentation

##### 32.116.3.1 `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\(\)](#).

##### 32.116.3.2 `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\(\)](#).

##### 32.116.3.3 `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\(\)](#).

#### 32.116.3.4 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\(\)](#).

#### 32.116.3.5 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\(\)](#).

#### 32.116.3.6 void stdair::ParsedKey::toStream ( std::ostream & ioOut ) const [virtual]

Dump a Business Object Key into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 130 of file [ParsedKey.cpp](#).

References [toString\(\)](#).

#### 32.116.3.7 void stdair::ParsedKey::fromStream ( std::istream & ioIn ) [virtual]

Read a Business Object Key from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 135 of file [ParsedKey.cpp](#).

#### 32.116.3.8 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\(\)](#).

## 32.116.4 Member Data Documentation

## 32.116.4.1 std::string stdair::ParsedKey::\_fullKey

Definition at line 76 of file [ParsedKey.hpp](#).

Referenced by [stdair::BomKeyManager::extractKeys\(\)](#), [getBoardingTime\(\)](#), [getFlightDateKey\(\)](#), [getInventoryKey\(\)](#), [getLegKey\(\)](#), and [getSegmentKey\(\)](#).

## 32.116.4.2 std::string stdair::ParsedKey::\_airlineCode

Definition at line 77 of file [ParsedKey.hpp](#).

Referenced by [stdair::BomKeyManager::extractKeys\(\)](#), [getInventoryKey\(\)](#), [stdair::BomRetriever::retrieveSegment←DateFromLongKey\(\)](#), and [toString\(\)](#).

## 32.116.4.3 std::string stdair::ParsedKey::\_flightNumber

Definition at line 78 of file [ParsedKey.hpp](#).

Referenced by [stdair::BomKeyManager::extractKeys\(\)](#), [getFlightDateKey\(\)](#), and [toString\(\)](#).

## 32.116.4.4 std::string stdair::ParsedKey::\_departureDate

Definition at line 79 of file [ParsedKey.hpp](#).

Referenced by [stdair::BomKeyManager::extractKeys\(\)](#), [getFlightDateKey\(\)](#), and [toString\(\)](#).

## 32.116.4.5 std::string stdair::ParsedKey::\_boardingPoint

Definition at line 80 of file [ParsedKey.hpp](#).

Referenced by [stdair::BomKeyManager::extractKeys\(\)](#), [getLegKey\(\)](#), [getSegmentKey\(\)](#), and [toString\(\)](#).

## 32.116.4.6 std::string stdair::ParsedKey::\_offPoint

Definition at line 81 of file [ParsedKey.hpp](#).

Referenced by [stdair::BomKeyManager::extractKeys\(\)](#), [getSegmentKey\(\)](#), and [toString\(\)](#).

## 32.116.4.7 std::string stdair::ParsedKey::\_boardingTime

Definition at line 82 of file [ParsedKey.hpp](#).

Referenced by [stdair::BomKeyManager::extractKeys\(\)](#), [getBoardingTime\(\)](#), and [toString\(\)](#).

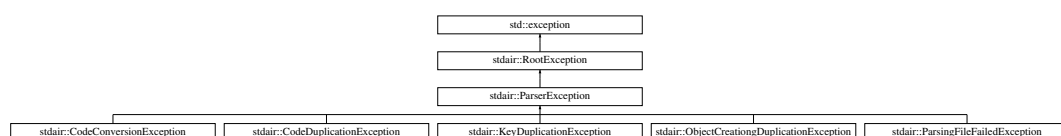
The documentation for this struct was generated from the following files:

- [stdair/bom/ParsedKey.hpp](#)
- [stdair/bom/ParsedKey.cpp](#)

## 32.117 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](#)

### 32.117.1 Detailed Description

Parser.

Definition at line 112 of file [stdair\\_exceptions.hpp](#).

### 32.117.2 Constructor & Destructor Documentation

32.117.2.1 `stdair::ParserException::ParserException ( const std::string & iWhat )` `[inline]`

Constructor.

Definition at line 115 of file [stdair\\_exceptions.hpp](#).

### 32.117.3 Member Function Documentation

32.117.3.1 `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\(\)](#).

### 32.117.4 Member Data Documentation

32.117.4.1 `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\(\)](#).

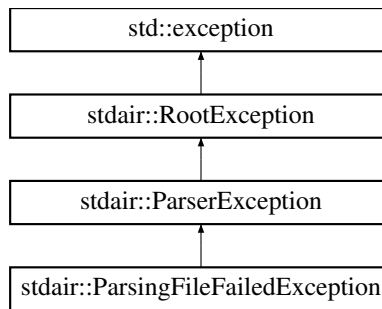
The documentation for this class was generated from the following file:

- [stdair/stdair\\_exceptions.hpp](#)

## 32.118 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](#)

#### 32.118.1 Detailed Description

Input file parsing failure.

Definition at line 173 of file [stdair\\_exceptions.hpp](#).

#### 32.118.2 Constructor & Destructor Documentation

32.118.2.1 `stdair::ParsingFileFailedException::ParsingFileFailedException ( const std::string & iWhat ) [inline]`

Constructor.

Definition at line 176 of file [stdair\\_exceptions.hpp](#).

#### 32.118.3 Member Function Documentation

32.118.3.1 `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\(\)](#).

#### 32.118.4 Member Data Documentation

32.118.4.1 `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:

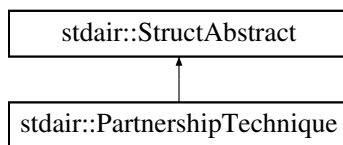
- [stdair/stdair\\_exceptions.hpp](#)

### 32.119 stdair::PartnershipTechnique Struct Reference

Enumeration of partnership techniques.

```
#include <stdair/basic/PartnershipTechnique.hpp>
```

Inheritance diagram for stdair::PartnershipTechnique:



#### Public Types

- enum [EN\\_PartnershipTechnique](#) {  
[NONE](#) = 0, [RAE\\_DA](#), [RAE\\_YP](#), [IBP\\_DA](#),  
[IBP\\_YP](#), [IBP\\_YP\\_U](#), [RMC](#), [A\\_RMC](#),  
[LAST\\_VALUE](#) }

#### Public Member Functions

- [EN\\_PartnershipTechnique](#) [getTechnique](#) () const
- char [getTechniqueAsChar](#) () const
- std::string [getTechniqueAsString](#) () const
- const std::string [describe](#) () const
- bool [operator==](#) (const [EN\\_PartnershipTechnique](#) &) const
- [PartnershipTechnique](#) (const [EN\\_PartnershipTechnique](#) &)
- [PartnershipTechnique](#) (const char iTechnique)
- [PartnershipTechnique](#) (const std::string &iTechnique)
- [PartnershipTechnique](#) (const [PartnershipTechnique](#) &)
- void [toStream](#) (std::ostream &ioOut) const
- virtual void [fromStream](#) (std::istream &ioIn)

#### Static Public Member Functions

- static const std::string & [getLabel](#) (const [EN\\_PartnershipTechnique](#) &)
- static [EN\\_PartnershipTechnique](#) [getTechnique](#) (const char)
- static char [getTechniqueLabel](#) (const [EN\\_PartnershipTechnique](#) &)
- static std::string [getTechniqueLabelAsString](#) (const [EN\\_PartnershipTechnique](#) &)
- static std::string [describeLabels](#) ()

#### 32.119.1 Detailed Description

Enumeration of partnership techniques.

Definition at line 17 of file [PartnershipTechnique.hpp](#).

#### 32.119.2 Member Enumeration Documentation

##### 32.119.2.1 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](#).

### 32.119.3 Constructor & Destructor Documentation

32.119.3.1 **stdair::PartnershipTechnique::PartnershipTechnique ( const EN\_PartnershipTechnique & *iPartnershipTechnique* )**

Main constructor.

Definition at line 48 of file [PartnershipTechnique.cpp](#).

32.119.3.2 **stdair::PartnershipTechnique::PartnershipTechnique ( const char *iTechnique* )**

Alternative constructor.

Definition at line 82 of file [PartnershipTechnique.cpp](#).

32.119.3.3 **stdair::PartnershipTechnique::PartnershipTechnique ( const std::string & *iTechnique* )**

Alternative constructor.

Definition at line 88 of file [PartnershipTechnique.cpp](#).

References [getTechnique\(\)](#).

32.119.3.4 **stdair::PartnershipTechnique::PartnershipTechnique ( const PartnershipTechnique & *iPartnershipTechnique* )**

Default copy constructor.

Definition at line 42 of file [PartnershipTechnique.cpp](#).

### 32.119.4 Member Function Documentation

32.119.4.1 **const std::string & stdair::PartnershipTechnique::getLabel ( const EN\_PartnershipTechnique & *iTechnique* )**  
**[static]**

Get the label as a string (e.g., "RevenueManagementCooperation").

Definition at line 98 of file [PartnershipTechnique.cpp](#).

32.119.4.2 **PartnershipTechnique::EN\_PartnershipTechnique stdair::PartnershipTechnique::getTechnique ( const char *iTechniqueChar* )**  
**[static]**

Get the technique value from parsing a single char (e.g., 'r' or 'C').

Definition at line 54 of file [PartnershipTechnique.cpp](#).

References [A\\_RMC](#), [describeLabels\(\)](#), [IBP\\_DA](#), [IBP\\_YP](#), [IBP\\_YP\\_U](#), [LAST\\_VALUE](#), [NONE](#), [RAE\\_DA](#), [RAE\\_YP](#), and [RMC](#).

Referenced by [stdair::AirlineFeature::getPartnershipTechnique\(\)](#).

**32.119.4.3** `char stdair::PartnershipTechnique::getTechniqueLabel ( const EN_PartnershipTechnique & iTechnique )`  
`[static]`

Get the label as a single char (e.g., 'r' or 'C').

Definition at line 104 of file [PartnershipTechnique.cpp](#).

**32.119.4.4** `std::string stdair::PartnershipTechnique::getTechniqueLabelAsString ( const EN_PartnershipTechnique & iTechnique )`  
`[static]`

Get the label as a string of a single char (e.g., "r" or "C").

Definition at line 110 of file [PartnershipTechnique.cpp](#).

**32.119.4.5** `std::string stdair::PartnershipTechnique::describeLabels ( )` `[static]`

List the labels.

Definition at line 117 of file [PartnershipTechnique.cpp](#).

References [LAST\\_VALUE](#).

Referenced by [getTechnique\(\)](#).

**32.119.4.6** `PartnershipTechnique::EN_PartnershipTechnique stdair::PartnershipTechnique::getTechnique ( ) const`

Get the enumerated value.

Definition at line 130 of file [PartnershipTechnique.cpp](#).

Referenced by [PartnershipTechnique\(\)](#).

**32.119.4.7** `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](#).

**32.119.4.8** `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](#).

**32.119.4.9** `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](#).

**32.119.4.10** `bool stdair::PartnershipTechnique::operator== ( const EN_PartnershipTechnique & iTechnique ) const`

Comparison operator.

Definition at line 156 of file [PartnershipTechnique.cpp](#).

**32.119.4.11** `void stdair::StructAbstract::toStream ( std::ostream & ioOut ) const` `[inline], [inherited]`

Dump a Business Object into an output stream.

**Parameters**

---

<i>ostream</i> &	the output stream.
------------------	--------------------

Definition at line 29 of file [StructAbstract.hpp](#).

References [stdair::StructAbstract::describe\(\)](#).

32.119.4.12 virtual void stdair::StructAbstract::fromStream ( std::istream & *ioIn* ) [inline], [virtual], [inherited]

Read a Business Object from an input stream.

Parameters

<i>istream</i> &	the input stream.
------------------	-------------------

Reimplemented in [stdair::BookingRequestStruct](#), [stdair::EventStruct](#), [stdair::TravelSolutionStruct](#), [stdair::VirtualClassStruct](#), [stdair::ConfigHolderStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::ProgressStatusSet](#), [stdair::YieldRange](#), [stdair::CancellationStruct](#), [stdair::AirlineStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::FFDisutilityCurveHolderStruct](#), [stdair::FRAT5CurveHolderStruct](#), and [stdair::BreakPointStruct](#).

Definition at line 38 of file [StructAbstract.hpp](#).

Referenced by [operator>>\(\)](#).

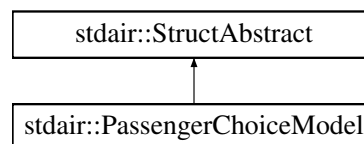
The documentation for this struct was generated from the following files:

- [stdair/basic/PartnershipTechnique.hpp](#)
- [stdair/basic/PartnershipTechnique.cpp](#)

## 32.120 stdair::PassengerChoiceModel Struct Reference

```
#include <stdair/basic/PassengerChoiceModel.hpp>
```

Inheritance diagram for stdair::PassengerChoiceModel:



### Public Types

- 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](#) ()

### 32.120.1 Detailed Description

Enumeration of passenger choice models.

Definition at line 15 of file [PassengerChoiceModel.hpp](#).

### 32.120.2 Member Enumeration Documentation

#### 32.120.2.1 enum stdair::PassengerChoiceModel::EN\_PassengerChoiceModel

Enumerator

***HARD\_RESTRICTION***  
***PRICE\_ORIENTED***  
***HYBRID***  
***LAST\_VALUE***

Definition at line 17 of file [PassengerChoiceModel.hpp](#).

### 32.120.3 Constructor & Destructor Documentation

#### 32.120.3.1 stdair::PassengerChoiceModel::PassengerChoiceModel ( const [EN\\_PassengerChoiceModel](#) & *iPassengerChoiceModel* )

Constructor.

Definition at line 36 of file [PassengerChoiceModel.cpp](#).

#### 32.120.3.2 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](#).

#### 32.120.3.3 stdair::PassengerChoiceModel::PassengerChoiceModel ( const [PassengerChoiceModel](#) & *iPassengerChoiceModel* )

Default copy constructor.

Definition at line 30 of file [PassengerChoiceModel.cpp](#).

### 32.120.4 Member Function Documentation

#### 32.120.4.1 const std::string & stdair::PassengerChoiceModel::getLabel ( const [EN\\_PassengerChoiceModel](#) & *iModel* ) [static]

Get the label as a string (e.g., `HardRestrictionModel`, `PriceOrientedModel` or `HybridModel`).

Definition at line 60 of file [PassengerChoiceModel.cpp](#).

**32.120.4.2** `char stdair::PassengerChoiceModel::getModelLabel ( const EN_PassengerChoiceModel & iModel )`  
`[static]`

Get the label as a single char (e.g., 'R', 'P' or 'H').

Definition at line 65 of file [PassengerChoiceModel.cpp](#).

**32.120.4.3** `std::string stdair::PassengerChoiceModel::getModelLabelAsString ( const EN_PassengerChoiceModel & iModel )`  
`[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](#).

**32.120.4.4** `std::string stdair::PassengerChoiceModel::describeLabels ( )` `[static]`

List the labels.

Definition at line 78 of file [PassengerChoiceModel.cpp](#).

References [LAST\\_VALUE](#).

Referenced by [PassengerChoiceModel\(\)](#).

**32.120.4.5** `PassengerChoiceModel::EN_PassengerChoiceModel stdair::PassengerChoiceModel::getModel ( ) const`

Get the enumerated value.

Definition at line 90 of file [PassengerChoiceModel.cpp](#).

**32.120.4.6** `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](#).

**32.120.4.7** `const std::string stdair::PassengerChoiceModel::describe ( ) const` `[virtual]`

Give a description of the structure (e.g., HardRestrictionModel", "PriceOrientedModel" or "HybridModel").

Implements [stdair::StructAbstract](#).

Definition at line 102 of file [PassengerChoiceModel.cpp](#).

**32.120.4.8** `bool stdair::PassengerChoiceModel::operator== ( const EN_PassengerChoiceModel & iModel ) const`

Comparison operator.

Definition at line 110 of file [PassengerChoiceModel.cpp](#).

**32.120.4.9** `void stdair::StructAbstract::toStream ( std::ostream & ioOut ) const` `[inline],[inherited]`

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Definition at line 29 of file [StructAbstract.hpp](#).

References [stdair::StructAbstract::describe\(\)](#).

**32.120.4.10** `virtual void stdair::StructAbstract::fromStream ( std::istream & ioIn )` `[inline],[virtual],[inherited]`

Read a Business Object from an input stream.

## Parameters

<i>istream</i> &	the input stream.
------------------	-------------------

Reimplemented in [stdair::BookingRequestStruct](#), [stdair::EventStruct](#), [stdair::TravelSolutionStruct](#), [stdair::VirtualClassStruct](#), [stdair::ConfigHolderStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::ProgressStatusSet](#), [stdair::YieldRange](#), [stdair::CancellationStruct](#), [stdair::AirlineStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::FFDisutilityCurveHolderStruct](#), [stdair::FRAT5CurveHolderStruct](#), and [stdair::BreakPointStruct](#).

Definition at line 38 of file [StructAbstract.hpp](#).

Referenced by [operator>>\(\)](#).

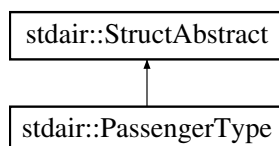
The documentation for this struct was generated from the following files:

- [stdair/basic/PassengerChoiceModel.hpp](#)
- [stdair/basic/PassengerChoiceModel.cpp](#)

### 32.121 stdair::PassengerType Struct Reference

```
#include <stdair/basic/PassengerType.hpp>
```

Inheritance diagram for [stdair::PassengerType](#):



## Public Types

- enum [EN\\_PassengerType](#) { [LEISURE](#) = 0, [BUSINESS](#), [FIRST](#), [LAST\\_VALUE](#) }

## Public Member Functions

- [EN\\_PassengerType](#) [getType](#) () const
- [std::string](#) [getTypeAsString](#) () const
- [const std::string](#) [describe](#) () const
- [bool](#) [operator==](#) (const [EN\\_PassengerType](#) &) const
- [PassengerType](#) (const [EN\\_PassengerType](#) &)
- [PassengerType](#) (const [char](#) iType)
- [void](#) [toStream](#) ([std::ostream](#) &ioOut) const
- [virtual void](#) [fromStream](#) ([std::istream](#) &ioIn)

## Static Public Member Functions

- [static const std::string](#) & [getLabel](#) (const [EN\\_PassengerType](#) &)
- [static char](#) [getTypeLabel](#) (const [EN\\_PassengerType](#) &)
- [static std::string](#) [getTypeLabelAsString](#) (const [EN\\_PassengerType](#) &)
- [static std::string](#) [describeLabels](#) ()

#### 32.121.1 Detailed Description

Enumeration of Frequent Flyer types.

Definition at line 15 of file [PassengerType.hpp](#).

## 32.121.2 Member Enumeration Documentation

## 32.121.2.1 enum stdair::PassengerType::EN\_PassengerType

Enumerator

**LEISURE**  
**BUSINESS**  
**FIRST**  
**LAST\_VALUE**

Definition at line 17 of file [PassengerType.hpp](#).

## 32.121.3 Constructor &amp; Destructor Documentation

## 32.121.3.1 stdair::PassengerType::PassengerType ( const EN\_PassengerType &amp; iPassengerType )

Constructor.

Definition at line 21 of file [PassengerType.cpp](#).

## 32.121.3.2 stdair::PassengerType::PassengerType ( const char iType )

Constructor.

Definition at line 26 of file [PassengerType.cpp](#).

References [BUSINESS](#), [describeLabels\(\)](#), [FIRST](#), [LAST\\_VALUE](#), and [LEISURE](#).

## 32.121.4 Member Function Documentation

## 32.121.4.1 const std::string &amp; stdair::PassengerType::getLabel ( const EN\_PassengerType &amp; iType ) [static]

Get the label as a string (e.g., "Leisure" or "Business").

Definition at line 44 of file [PassengerType.cpp](#).

## 32.121.4.2 char stdair::PassengerType::getTypeLabel ( const EN\_PassengerType &amp; iType ) [static]

Get the label as a single char (e.g., 'L' or 'B').

Definition at line 49 of file [PassengerType.cpp](#).

## 32.121.4.3 std::string stdair::PassengerType::getTypeLabelAsString ( const EN\_PassengerType &amp; iType ) [static]

Get the label as a single char (e.g., 'L' or 'B').

Definition at line 55 of file [PassengerType.cpp](#).

## 32.121.4.4 std::string stdair::PassengerType::describeLabels ( ) [static]

List the labels.

Definition at line 62 of file [PassengerType.cpp](#).

References [LAST\\_VALUE](#).

Referenced by [PassengerType\(\)](#).

## 32.121.4.5 PassengerType::EN\_PassengerType stdair::PassengerType::getType ( ) const

Get the enumerated value.

Definition at line 74 of file [PassengerType.cpp](#).

**32.121.4.6** `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](#).

**32.121.4.7** `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](#).

**32.121.4.8** `bool stdair::PassengerType::operator== ( const EN_PassengerType & iType ) const`

Comparison operator.

Definition at line 93 of file [PassengerType.cpp](#).

**32.121.4.9** `void stdair::StructAbstract::toStream ( std::ostream & ioOut ) const` `[inline], [inherited]`

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Definition at line 29 of file [StructAbstract.hpp](#).

References [stdair::StructAbstract::describe\(\)](#).

**32.121.4.10** `virtual void stdair::StructAbstract::fromStream ( std::istream & ioIn )` `[inline], [virtual], [inherited]`

Read a Business Object from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented in [stdair::BookingRequestStruct](#), [stdair::EventStruct](#), [stdair::TravelSolutionStruct](#), [stdair::VirtualClassStruct](#), [stdair::ConfigHolderStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::ProgressStatusSet](#), [stdair::YieldRange](#), [stdair::CancellationStruct](#), [stdair::AirlineStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::FFDisutilityCurveHolderStruct](#), [stdair::FRAT5CurveHolderStruct](#), and [stdair::BreakPointStruct](#).

Definition at line 38 of file [StructAbstract.hpp](#).

Referenced by [operator>>\(\)](#).

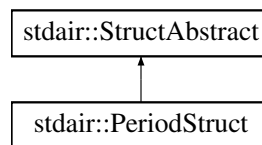
The documentation for this struct was generated from the following files:

- [stdair/basic/PassengerType.hpp](#)
- [stdair/basic/PassengerType.cpp](#)

**32.122** `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)

#### 32.122.1 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](#).

#### 32.122.2 Constructor & Destructor Documentation

##### 32.122.2.1 stdair::PeriodStruct::PeriodStruct ( const [DatePeriod\\_T](#) & iDateRange, const [DoWStruct](#) & iDoW )

Constructor.

Definition at line 19 of file [PeriodStruct.cpp](#).

##### 32.122.2.2 stdair::PeriodStruct::PeriodStruct ( )

Default constructors.

Definition at line 14 of file [PeriodStruct.cpp](#).

Referenced by [addDateOffset\(\)](#), and [intersection\(\)](#).

##### 32.122.2.3 stdair::PeriodStruct::PeriodStruct ( const [PeriodStruct](#) & iPeriodStruct )

Definition at line 25 of file [PeriodStruct.cpp](#).

##### 32.122.2.4 stdair::PeriodStruct::~~PeriodStruct ( ) [inline]

Default destructor.

Definition at line 64 of file [PeriodStruct.hpp](#).

### 32.122.3 Member Function Documentation

#### 32.122.3.1 `const DatePeriod_T& stdair::PeriodStruct::getDateRange ( ) const` `[inline]`

Retrieve the attributes.

Definition at line 23 of file [PeriodStruct.hpp](#).

Referenced by [addDateOffset\(\)](#).

#### 32.122.3.2 `const DoWStruct& stdair::PeriodStruct::getDoW ( ) const` `[inline]`

Definition at line 26 of file [PeriodStruct.hpp](#).

Referenced by [addDateOffset\(\)](#).

#### 32.122.3.3 `void stdair::PeriodStruct::setDateRange ( const DatePeriod_T & iDateRange )` `[inline]`

Set the new value for the attributes.

Definition at line 33 of file [PeriodStruct.hpp](#).

#### 32.122.3.4 `void stdair::PeriodStruct::setDoW ( const DoWStruct & iDoW )` `[inline]`

Definition at line 36 of file [PeriodStruct.hpp](#).

#### 32.122.3.5 `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\(\)](#).

#### 32.122.3.6 `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\(\)](#).

#### 32.122.3.7 `PeriodStruct stdair::PeriodStruct::addDateOffset ( const DateOffset_T & iDateOffset ) const`

Build a period struct from this period struct by adding a date offset.

Definition at line 46 of file [PeriodStruct.cpp](#).

References [getDateRange\(\)](#), [getDoW\(\)](#), [PeriodStruct\(\)](#), and [stdair::DoWStruct::shift\(\)](#).

#### 32.122.3.8 `PeriodStruct stdair::PeriodStruct::intersection ( const PeriodStruct & iPeriodStruct ) const`

Build a new period struct which is the intersection of two period structs.

Definition at line 63 of file [PeriodStruct.cpp](#).

References [stdair::DoWStruct::intersection\(\)](#), and [PeriodStruct\(\)](#).

#### 32.122.3.9 `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\(\)](#).

32.122.3.10 `void stdair::StructAbstract::toStream ( std::ostream & ioOut ) const` `[inline]`, `[inherited]`

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Definition at line 29 of file [StructAbstract.hpp](#).

References [stdair::StructAbstract::describe\(\)](#).

32.122.3.11 `virtual void stdair::StructAbstract::fromStream ( std::istream & ioIn )` `[inline]`, `[virtual]`, `[inherited]`

Read a Business Object from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented in [stdair::BookingRequestStruct](#), [stdair::EventStruct](#), [stdair::TravelSolutionStruct](#), [stdair::VirtualClassStruct](#), [stdair::ConfigHolderStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::ProgressStatusSet](#), [stdair::YieldRange](#), [stdair::CancellationStruct](#), [stdair::AirlineStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::FFDisutilityCurveHolderStruct](#), [stdair::FRAT5CurveHolderStruct](#), and [stdair::BreakPointStruct](#).

Definition at line 38 of file [StructAbstract.hpp](#).

Referenced by [operator>>\(\)](#).

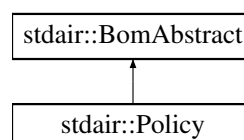
The documentation for this struct was generated from the following files:

- [stdair/bom/PeriodStruct.hpp](#)
- [stdair/bom/PeriodStruct.cpp](#)

## 32.123 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

- template<typename BOM >  
class [FacBom](#)
- class [FacBomManager](#)
- class [boost::serialization::access](#)

### 32.123.1 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](#).

### 32.123.2 Member Typedef Documentation

#### 32.123.2.1 typedef PolicyKey stdair::Policy::Key\_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 40 of file [Policy.hpp](#).

### 32.123.3 Constructor & Destructor Documentation

#### 32.123.3.1 stdair::Policy::Policy ( const Key\_T & iKey ) [protected]

Main constructor.

Definition at line 31 of file [Policy.cpp](#).

#### 32.123.3.2 stdair::Policy::~~Policy ( ) [protected],[virtual]

Destructor.

Definition at line 35 of file [Policy.cpp](#).

#### 32.123.4 Member Function Documentation

**32.123.4.1** `const Key_T& stdair::Policy::getKey ( ) const` `[inline]`

Get the policy key.

Definition at line 45 of file [Policy.hpp](#).

**32.123.4.2** `BomAbstract* const stdair::Policy::getParent ( ) const` `[inline]`

Get the parent object.

Definition at line 50 of file [Policy.hpp](#).

**32.123.4.3** `const HolderMap_T& stdair::Policy::getHolderMap ( ) const` `[inline]`

Get the map of children holders.

Definition at line 57 of file [Policy.hpp](#).

**32.123.4.4** `const BookingClassList_T & stdair::Policy::getBookingClassList ( ) const`

Getter for the booking classes.

Definition at line 52 of file [Policy.cpp](#).

**32.123.4.5** `const NbOfBookings_T& stdair::Policy::getDemand ( ) const` `[inline]`

Getter for the demand.

Definition at line 65 of file [Policy.hpp](#).

**32.123.4.6** `const StdDevValue_T& stdair::Policy::getStdDev ( ) const` `[inline]`

Getter for the standard deviation demand.

Definition at line 70 of file [Policy.hpp](#).

**32.123.4.7** `const Yield_T& stdair::Policy::getYield ( ) const` `[inline]`

Getter for the yield.

Definition at line 75 of file [Policy.hpp](#).

**32.123.4.8** `const Revenue_T stdair::Policy::getTotalRevenue ( ) const`

Get the total revenue of the policy.

Definition at line 57 of file [Policy.cpp](#).

**32.123.4.9** `void stdair::Policy::setDemand ( const NbOfBookings_T & iDemand )` `[inline]`

Setter for the unconstraining demand.

Definition at line 85 of file [Policy.hpp](#).

**32.123.4.10** `void stdair::Policy::setStdDev ( const StdDevValue_T & iStdDev )` `[inline]`

Setter for standard deviation demand.

Definition at line 90 of file [Policy.hpp](#).

**32.123.4.11** `void stdair::Policy::setYield ( const Yield_T & iYield )` `[inline]`

Setter for the yield.

Definition at line 95 of file [Policy.hpp](#).

**32.123.4.12** `void stdair::Policy::resetDemandForecast ( ) [inline]`

Reset demand forecast.

Definition at line 100 of file [Policy.hpp](#).

**32.123.4.13** `void stdair::Policy::addYieldDemand ( const Yield_T & iYield, const NbOfBookings_T & iDemand )`

Add the new pair (yield, demand) to the map.

Definition at line 70 of file [Policy.cpp](#).

**32.123.4.14** `void stdair::Policy::toStream ( std::ostream & ioOut ) const [inline],[virtual]`

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Implements [stdair::BomAbstract](#).

Definition at line 116 of file [Policy.hpp](#).

References [toString\(\)](#).

**32.123.4.15** `void stdair::Policy::fromStream ( std::istream & ioin ) [inline],[virtual]`

Read a Business Object from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Implements [stdair::BomAbstract](#).

Definition at line 125 of file [Policy.hpp](#).

**32.123.4.16** `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 [stdair::SegmentCabin::describeConvexHull\(\)](#), and [toStream\(\)](#).

**32.123.4.17** `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\(\)](#).

**32.123.4.18** `template<class Archive > void stdair::Policy::serialize ( Archive & ar, const unsigned int iFileVersion )`

Serialisation.

**32.123.5** Friends And Related Function Documentation

32.123.5.1 `template<typename BOM > friend class FacBom` `[friend]`

Definition at line 31 of file [Policy.hpp](#).

32.123.5.2 `friend class FacBomManager` `[friend]`

Definition at line 32 of file [Policy.hpp](#).

32.123.5.3 `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:

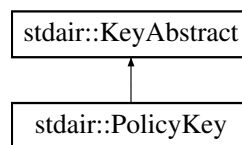
- [stdair/bom/Policy.hpp](#)
- [stdair/bom/Policy.cpp](#)

## 32.124 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](#)

### 32.124.1 Detailed Description

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

Definition at line 26 of file [PolicyKey.hpp](#).

### 32.124.2 Constructor & Destructor Documentation

#### 32.124.2.1 `stdair::PolicyKey::PolicyKey ( const PolicyCode_T & iPolicyCode )`

Constructor.

Definition at line 28 of file [PolicyKey.cpp](#).

#### 32.124.2.2 `stdair::PolicyKey::PolicyKey ( const PolicyKey & iPolicyKey )`

Copy constructor.

Definition at line 23 of file [PolicyKey.cpp](#).

#### 32.124.2.3 `stdair::PolicyKey::~~PolicyKey ( )`

Destructor.

Definition at line 33 of file [PolicyKey.cpp](#).

### 32.124.3 Member Function Documentation

#### 32.124.3.1 `const PolicyCode_T& stdair::PolicyKey::getPolicyCode ( ) const` `[inline]`

Get the policy code.

Definition at line 56 of file [PolicyKey.hpp](#).

#### 32.124.3.2 `void stdair::PolicyKey::toStream ( std::ostream & ioOut ) const` `[virtual]`

Dump a Business Object Key into an output stream.

Parameters

<code>ostream&amp;</code>	the output stream.
---------------------------	--------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 37 of file [PolicyKey.cpp](#).

References [toString\(\)](#).

#### 32.124.3.3 `void stdair::PolicyKey::fromStream ( std::istream & ioln )` `[virtual]`

Read a Business Object Key from an input stream.

Parameters

<code>istream&amp;</code>	the input stream.
---------------------------	-------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 42 of file [PolicyKey.cpp](#).

#### 32.124.3.4 `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\(\)](#).

32.124.3.5 `template<class Archive > void stdair::PolicyKey::serialize ( Archive & ar, const unsigned int iFileVersion )`

Serialisation.

Definition at line 68 of file [PolicyKey.cpp](#).

#### 32.124.4 Friends And Related Function Documentation

32.124.4.1 `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:

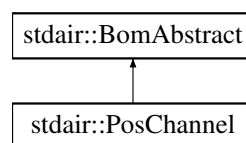
- [stdair/bom/PolicyKey.hpp](#)
- [stdair/bom/PolicyKey.cpp](#)

## 32.125 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

- `template<typename BOM >`  
class [FacBom](#)
- `template<typename BOM >`  
class [FacCloneBom](#)
- class [FacBomManager](#)

### 32.125.1 Detailed Description

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

Definition at line 19 of file [PosChannel.hpp](#).

### 32.125.2 Member Typedef Documentation

#### 32.125.2.1 `typedef PosChannelKey stdair::PosChannel::Key_T`

Definition allowing to retrieve the associated BOM key type.

Definition at line 29 of file [PosChannel.hpp](#).

### 32.125.3 Constructor & Destructor Documentation

#### 32.125.3.1 `stdair::PosChannel::PosChannel ( const Key_T & iKey ) [protected]`

Main constructor.

Definition at line 28 of file [PosChannel.cpp](#).

#### 32.125.3.2 `stdair::PosChannel::~~PosChannel ( ) [protected],[virtual]`

Destructor.

Definition at line 33 of file [PosChannel.cpp](#).

### 32.125.4 Member Function Documentation

#### 32.125.4.1 `void stdair::PosChannel::toStream ( std::ostream & ioOut ) const [inline],[virtual]`

Dump a Business Object into an output stream.

#### Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Implements [stdair::BomAbstract](#).

Definition at line 38 of file [PosChannel.hpp](#).

References [toString\(\)](#).

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

Read a Business Object from an input stream.



## Parameters

<i>istream</i> &	the input stream.
------------------	-------------------

Implements [stdair::BomAbstract](#).

Definition at line 47 of file [PosChannel.hpp](#).

32.125.4.3 `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\(\)](#).

32.125.4.4 `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\(\)](#).

32.125.4.5 `const Key_T& stdair::PosChannel::getKey ( ) const` `[inline]`

Get the primary key (pos, channel).

Definition at line 67 of file [PosChannel.hpp](#).

References [\\_key](#).

32.125.4.6 `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](#).

32.125.4.7 `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](#).

32.125.4.8 `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\(\)](#).

32.125.4.9 `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\(\)](#).

## 32.125.5 Friends And Related Function Documentation

32.125.5.1 `template<typename BOM > friend class FacBom` `[friend]`Definition at line 20 of file [PosChannel.hpp](#).32.125.5.2 `template<typename BOM > friend class FacCloneBom` `[friend]`Definition at line 21 of file [PosChannel.hpp](#).32.125.5.3 `friend class FacBomManager` `[friend]`Definition at line 22 of file [PosChannel.hpp](#).

## 32.125.6 Member Data Documentation

32.125.6.1 `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\(\)](#).32.125.6.2 `BomAbstract* stdair::PosChannel::_parent` `[protected]`

Pointer on the parent class.

Definition at line 132 of file [PosChannel.hpp](#).Referenced by [getParent\(\)](#).32.125.6.3 `HolderMap_T stdair::PosChannel::_holderMap` `[protected]`

Map holding the children.

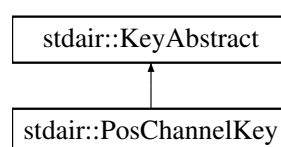
Definition at line 137 of file [PosChannel.hpp](#).Referenced by [getHolderMap\(\)](#).

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

- [stdair/bom/PosChannel.hpp](#)
- [stdair/bom/PosChannel.cpp](#)

## 32.126 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

### 32.126.1 Detailed Description

Key of point of sale and channel.

Definition at line 15 of file [PosChannelKey.hpp](#).

### 32.126.2 Constructor & Destructor Documentation

#### 32.126.2.1 `stdair::PosChannelKey::PosChannelKey ( const stdair::CityCode_T & iPos, const stdair::ChannelLabel_T & iChannel )`

Main constructor.

Definition at line 22 of file [PosChannelKey.cpp](#).

#### 32.126.2.2 `stdair::PosChannelKey::PosChannelKey ( const PosChannelKey & iKey )`

Copy constructor.

Definition at line 28 of file [PosChannelKey.cpp](#).

#### 32.126.2.3 `stdair::PosChannelKey::~~PosChannelKey ( )`

Destructor.

Definition at line 33 of file [PosChannelKey.cpp](#).

### 32.126.3 Member Function Documentation

#### 32.126.3.1 `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\(\)](#).

#### 32.126.3.2 `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\(\)](#).

#### 32.126.3.3 `void stdair::PosChannelKey::toStream ( std::ostream & ioOut ) const` `[virtual]`

Dump a Business Object Key into an output stream.

## Parameters

<i>ostream</i> &	the output stream.
------------------	--------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 37 of file [PosChannelKey.cpp](#).

References [toString\(\)](#).

**32.126.3.4** void [stdair::PosChannelKey::fromStream](#) ( [std::istream](#) & *ioln* ) [virtual]

Read a Business Object Key from an input stream.

## Parameters

<i>istream</i> &	the input stream.
------------------	-------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 42 of file [PosChannelKey.cpp](#).

**32.126.3.5** 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 [toString\(\)](#).

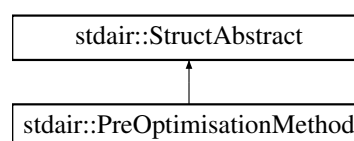
The documentation for this struct was generated from the following files:

- [stdair/bom/PosChannelKey.hpp](#)
- [stdair/bom/PosChannelKey.cpp](#)

## 32.127 stdair::PreOptimisationMethod Struct Reference

```
#include <stdair/basic/PreOptimisationMethod.hpp>
```

Inheritance diagram for [stdair::PreOptimisationMethod](#):



## Public Types

- 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](#) ()

#### 32.127.1 Detailed Description

Enumeration of PreOptimisation methods.

Definition at line 15 of file [PreOptimisationMethod.hpp](#).

#### 32.127.2 Member Enumeration Documentation

##### 32.127.2.1 enum stdair::PreOptimisationMethod::EN\_PreOptimisationMethod

Enumerator

**NONE**

**FA**

**MRT**

**LAST\_VALUE**

Definition at line 17 of file [PreOptimisationMethod.hpp](#).

#### 32.127.3 Constructor & Destructor Documentation

##### 32.127.3.1 stdair::PreOptimisationMethod::PreOptimisationMethod ( const [EN\\_PreOptimisationMethod](#) & *iPreOptimisationMethod* )

Constructor.

Definition at line 36 of file [PreOptimisationMethod.cpp](#).

##### 32.127.3.2 stdair::PreOptimisationMethod::PreOptimisationMethod ( const char *iMethod* )

Constructor.

Definition at line 41 of file [PreOptimisationMethod.cpp](#).

References [describeLabels\(\)](#), [FA](#), [LAST\\_VALUE](#), [MRT](#), and [NONE](#).

##### 32.127.3.3 stdair::PreOptimisationMethod::PreOptimisationMethod ( const [PreOptimisationMethod](#) & *iPreOptimisationMethod* )

Default copy constructor.

Definition at line 30 of file [PreOptimisationMethod.cpp](#).

## 32.127.4 Member Function Documentation

**32.127.4.1** `const std::string &stdair::PreOptimisationMethod::getLabel ( const EN_PreOptimisationMethod & iMethod )`  
`[static]`

Get the label as a string (e.g., MRT or FA).

Definition at line 60 of file [PreOptimisationMethod.cpp](#).

**32.127.4.2** `char stdair::PreOptimisationMethod::getMethodLabel ( const EN_PreOptimisationMethod & iMethod )`  
`[static]`

Get the label as a single char (e.g., 'M' or 'E').

Definition at line 65 of file [PreOptimisationMethod.cpp](#).

**32.127.4.3** `std::string stdair::PreOptimisationMethod::getMethodLabelAsString ( const EN_PreOptimisationMethod & iMethod )`  
`[static]`

Get the label as a string of a single char (e.g., "M" or "E").

Definition at line 71 of file [PreOptimisationMethod.cpp](#).

**32.127.4.4** `std::string stdair::PreOptimisationMethod::describeLabels ( )`  
`[static]`

List the labels.

Definition at line 78 of file [PreOptimisationMethod.cpp](#).

References [LAST\\_VALUE](#).

Referenced by [PreOptimisationMethod\(\)](#).

**32.127.4.5** `PreOptimisationMethod::EN_PreOptimisationMethod stdair::PreOptimisationMethod::getMethod ( )`  
`const`

Get the enumerated value.

Definition at line 90 of file [PreOptimisationMethod.cpp](#).

Referenced by [stdair::AirlineFeature::getPreOptimisationMethod\(\)](#).

**32.127.4.6** `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](#).

**32.127.4.7** `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](#).

**32.127.4.8** `bool stdair::PreOptimisationMethod::operator== ( const EN_PreOptimisationMethod & iMethod )`  
`const`

Comparison operator.

Definition at line 110 of file [PreOptimisationMethod.cpp](#).

**32.127.4.9** `void stdair::StructAbstract::toStream ( std::ostream & ioOut )`  
`const` `[inline]`, `[inherited]`

Dump a Business Object into an output stream.

## Parameters

<i>ostream</i> &	the output stream.
------------------	--------------------

Definition at line 29 of file [StructAbstract.hpp](#).

References [stdair::StructAbstract::describe\(\)](#).

**32.127.4.10** virtual void [stdair::StructAbstract::fromStream](#) ( [std::istream](#) & *ioIn* ) [inline],[virtual],  
[inherited]

Read a Business Object from an input stream.

## Parameters

<i>istream</i> &	the input stream.
------------------	-------------------

Reimplemented in [stdair::BookingRequestStruct](#), [stdair::EventStruct](#), [stdair::TravelSolutionStruct](#), [stdair::VirtualClassStruct](#), [stdair::ConfigHolderStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::ProgressStatusSet](#), [stdair::YieldRange](#), [stdair::CancellationStruct](#), [stdair::AirlineStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::FFDisutilityCurveHolderStruct](#), [stdair::FRAT5CurveHolderStruct](#), and [stdair::BreakPointStruct](#).

Definition at line 38 of file [StructAbstract.hpp](#).

Referenced by [operator>>\(\)](#).

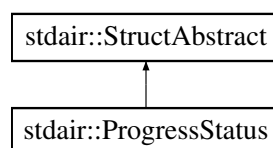
The documentation for this struct was generated from the following files:

- [stdair/basic/PreOptimisationMethod.hpp](#)
- [stdair/basic/PreOptimisationMethod.cpp](#)

## 32.128 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)

### 32.128.1 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](#).

### 32.128.2 Constructor & Destructor Documentation

**32.128.2.1** `stdair::ProgressStatus::ProgressStatus ( const Count_T &iCurrentNb, const Count_T &iExpectedNb, const Count_T &iActualNb )`

Constructor.

Parameters

<i>const</i>	Count_T& The current number.
<i>const</i>	Count_T& The expected number.
<i>const</i>	Count_T& The actual number.

Definition at line 15 of file [ProgressStatus.cpp](#).

**32.128.2.2** `stdair::ProgressStatus::ProgressStatus ( const Count_T &iExpectedNb, const Count_T &iActualNb )`

Constructor.

As no current number is given, it is set to 0.

Parameters

<i>const</i>	Count_T& The expected number.
<i>const</i>	Count_T& The actual number.

Definition at line 23 of file [ProgressStatus.cpp](#).

**32.128.2.3** `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

<i>const</i>	Count_T& The actual number.
--------------	-----------------------------

Definition at line 30 of file [ProgressStatus.cpp](#).

**32.128.2.4** `stdair::ProgressStatus::ProgressStatus ( )`

Constructor.

All the numbers are set to 0.

Definition at line 36 of file [ProgressStatus.cpp](#).



32.128.2.5 `stdair::ProgressStatus::ProgressStatus ( const ProgressStatus & iProgressStatus )`

Copy Constructor.

Definition at line 43 of file [ProgressStatus.cpp](#).

### 32.128.3 Member Function Documentation

32.128.3.1 `const Count_T& stdair::ProgressStatus::count ( ) const [inline]`

Get the current number.

Definition at line 31 of file [ProgressStatus.hpp](#).

32.128.3.2 `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\(\)](#).

32.128.3.3 `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\(\)](#).

32.128.3.4 `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\(\)](#).

32.128.3.5 `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\(\)](#).

32.128.3.6 `void stdair::ProgressStatus::setCurrentNb ( const Count_T & iCurrentNb ) [inline]`

Set the current number.

Definition at line 65 of file [ProgressStatus.hpp](#).

32.128.3.7 `void stdair::ProgressStatus::setExpectedNb ( const Count_T & iExpectedNb ) [inline]`

Set the expected number.

Definition at line 70 of file [ProgressStatus.hpp](#).

32.128.3.8 `void stdair::ProgressStatus::setActualNb ( const Count_T & iActualNb ) [inline]`

Set the actual number.

Definition at line 75 of file [ProgressStatus.hpp](#).

32.128.3.9 void stdair::ProgressStatus::reset ( )

Reset the current number (to 0).

Definition at line 50 of file [ProgressStatus.cpp](#).

References [stdair::DEFAULT\\_PROGRESS\\_STATUS](#).

32.128.3.10 Count\_T stdair::ProgressStatus::operator+= ( Count\_T *increment* ) [inline]

Increment the current number.

Definition at line 83 of file [ProgressStatus.hpp](#).

32.128.3.11 Count\_T stdair::ProgressStatus::operator++ ( ) [inline]

Increment the current number.

Definition at line 89 of file [ProgressStatus.hpp](#).

32.128.3.12 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](#).

32.128.3.13 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\(\)](#).

32.128.3.14 void stdair::StructAbstract::toStream ( std::ostream & *ioOut* ) const [inline], [inherited]

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Definition at line 29 of file [StructAbstract.hpp](#).

References [stdair::StructAbstract::describe\(\)](#).

32.128.3.15 virtual void stdair::StructAbstract::fromStream ( std::istream & *ioIn* ) [inline], [virtual], [inherited]

Read a Business Object from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented in [stdair::BookingRequestStruct](#), [stdair::EventStruct](#), [stdair::TravelSolutionStruct](#), [stdair::VirtualClassStruct](#), [stdair::ConfigHolderStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::ProgressStatusSet](#), [stdair::YieldRange](#), [stdair::CancellationStruct](#), [stdair::AirlineStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::FFDisutilityCurveHolderStruct](#), [stdair::FRAT5CurveHolderStruct](#), and [stdair::BreakPointStruct](#).

Definition at line 38 of file [StructAbstract.hpp](#).

Referenced by [operator>>\(\)](#).

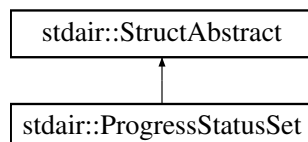
The documentation for this struct was generated from the following files:

- [stdair/basic/ProgressStatus.hpp](#)
- [stdair/basic/ProgressStatus.cpp](#)

### 32.129 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 [EventGeneratorKey\\_T](#) &iKey)
- void [setOverallStatus](#) (const [ProgressStatus](#) &iProgressStatus)
- void [fromStream](#) (std::istream &iIn)
- const std::string [describe](#) () const
- [ProgressStatusSet](#) (const [EventType::EN\\_EventType](#) &)
- [ProgressStatusSet](#) (const [ProgressStatusSet](#) &)
- [~ProgressStatusSet](#) ()
- void [toStream](#) (std::ostream &iOut) const

#### 32.129.1 Detailed Description

Structure holding a set of progress status.

Definition at line 22 of file [ProgressStatusSet.hpp](#).

#### 32.129.2 Constructor & Destructor Documentation

##### 32.129.2.1 stdair::ProgressStatusSet::ProgressStatusSet ( const EventType::EN\_EventType & iType )

Constructor .

Definition at line 20 of file [ProgressStatusSet.cpp](#).

##### 32.129.2.2 stdair::ProgressStatusSet::ProgressStatusSet ( const ProgressStatusSet & iProgressStatusSet )

Copy constructor.

Definition at line 27 of file [ProgressStatusSet.cpp](#).

##### 32.129.2.3 stdair::ProgressStatusSet::~~ProgressStatusSet ( )

Destructor.

Definition at line 36 of file [ProgressStatusSet.cpp](#).

## 32.129.3 Member Function Documentation

32.129.3.1 `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](#).

32.129.3.2 `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](#).

32.129.3.3 `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](#).

32.129.3.4 `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](#).

32.129.3.5 `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](#).

32.129.3.6 `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](#).

32.129.3.7 `void stdair::ProgressStatusSet::fromStream ( std::istream & iIn )` `[virtual]`

Read a Business Object from an input stream.

## Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented from [stdair::StructAbstract](#).

Definition at line 40 of file [ProgressStatusSet.cpp](#).

32.129.3.8 `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\(\)](#).

**32.129.3.9** `void stdair::StructAbstract::toStream ( std::ostream & ioOut ) const` `[inline],[inherited]`

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Definition at line 29 of file [StructAbstract.hpp](#).

References [stdair::StructAbstract::describe\(\)](#).

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

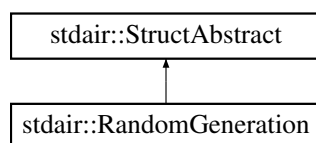
- [stdair/basic/ProgressStatusSet.hpp](#)
- [stdair/basic/ProgressStatusSet.cpp](#)

### 32.130 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](#)

#### 32.130.1 Detailed Description

Class holding a random generator.

Definition at line 17 of file [RandomGeneration.hpp](#).

### 32.130.2 Constructor & Destructor Documentation

#### 32.130.2.1 stdair::RandomGeneration::RandomGeneration ( const RandomSeed\_T & iSeed )

Main constructor.

Definition at line 27 of file [RandomGeneration.cpp](#).

#### 32.130.2.2 stdair::RandomGeneration::RandomGeneration ( )

Default constructor.

#### Note

As per Boost bug #3516 (<https://svn.boost.org/trac/boost/ticket/3516>) the seed should not be set to 0 (at least on versions of Boost lower than 1.44).

Definition at line 23 of file [RandomGeneration.cpp](#).

#### 32.130.2.3 stdair::RandomGeneration::~~RandomGeneration ( )

Destructor.

Definition at line 37 of file [RandomGeneration.cpp](#).

### 32.130.3 Member Function Documentation

#### 32.130.3.1 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\(\)\(\)](#).

#### 32.130.3.2 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\(\)](#).

#### 32.130.3.3 RealNumber\_T stdair::RandomGeneration::generateUniform ( const RealNumber\_T & iMinValue, const RealNumber\_T & iMaxValue )

Generate a randomized number following a uniform distribution between a minimum (included) and a maximum (excluded) value.

Definition at line 59 of file [RandomGeneration.cpp](#).

References [generateUniform01\(\)](#).

#### 32.130.3.4 RealNumber\_T stdair::RandomGeneration::generateNormal ( const RealNumber\_T & mu, const RealNumber\_T & sigma )

Generate a randomized number following a normal distribution specified by a mean and a standard deviation.

Definition at line 68 of file [RandomGeneration.cpp](#).

References [generateUniform01\(\)](#).

Referenced by [stdair::BookingClass::generateDemandSamples\(\)](#).

### 32.130.3.5 `RealNumber_T` `stdair::RandomGeneration::generateExponential ( const RealNumber_T & lambda )`

Generate a randomized number following an exponential distribution specified by a mean and a lambda parameter.

Definition at line 86 of file [RandomGeneration.cpp](#).

References [\\_generator](#).

### 32.130.3.6 `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](#).

### 32.130.3.7 `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](#).

### 32.130.3.8 `void` `stdair::RandomGeneration::init ( const RandomSeed_T & iSeed )`

Initialise the random generator.

A uniform random number distribution is defined, which produces "real" values between 0 and 1 (0 inclusive, 1 exclusive).

Definition at line 41 of file [RandomGeneration.cpp](#).

References [\\_generator](#).

### 32.130.3.9 `void` `stdair::StructAbstract::toStream ( std::ostream & ioOut ) const` `[inline]`, `[inherited]`

Dump a Business Object into an output stream.

Parameters

<code>ostream&amp;</code>	the output stream.
---------------------------	--------------------

Definition at line 29 of file [StructAbstract.hpp](#).

References [stdair::StructAbstract::describe\(\)](#).

### 32.130.3.10 `virtual void` `stdair::StructAbstract::fromStream ( std::istream & ioIn )` `[inline]`, `[virtual]`, `[inherited]`

Read a Business Object from an input stream.

Parameters

<code>istream&amp;</code>	the input stream.
---------------------------	-------------------

Reimplemented in [stdair::BookingRequestStruct](#), [stdair::EventStruct](#), [stdair::TravelSolutionStruct](#), [stdair::VirtualClassStruct](#), [stdair::ConfigHolderStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::ProgressStatusSet](#), [stdair::YieldRange](#), [stdair::CancellationStruct](#), [stdair::AirlineStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::FFDisutilityCurveHolderStruct](#), [stdair::FRAT5CurveHolderStruct](#), and [stdair::BreakPointStruct](#).

Definition at line 38 of file [StructAbstract.hpp](#).

Referenced by [operator>>\(\)](#).

## 32.130.4 Member Data Documentation

## 32.130.4.1 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\(\)](#).

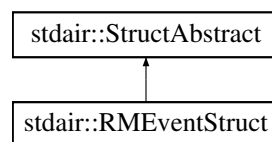
The documentation for this struct was generated from the following files:

- [stdair/basic/RandomGeneration.hpp](#)
- [stdair/basic/RandomGeneration.cpp](#)

## 32.131 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 ()`

## 32.131.1 Detailed Description

Structure holding the elements of a snapshot .

Definition at line 19 of file [RMEventStruct.hpp](#).

## 32.131.2 Constructor &amp; Destructor Documentation

## 32.131.2.1 stdair::RMEventStruct::RMEventStruct ( const AirlineCode\_T &amp; iAirlineCode, const KeyDescription\_T &amp; iFlightDateDescription, const DateTime\_T &amp; iRMEventTime )

Constructor.

Definition at line 27 of file [RMEventStruct.cpp](#).



### 32.131.2.2 `stdair::RMEventStruct::RMEventStruct ( const RMEventStruct & iRMEvent )`

Copy constructor.

Definition at line 19 of file [RMEventStruct.cpp](#).

### 32.131.2.3 `stdair::RMEventStruct::RMEventStruct ( )`

Default constructor.

It is private so that it can not be used.

Definition at line 13 of file [RMEventStruct.cpp](#).

### 32.131.2.4 `stdair::RMEventStruct::~~RMEventStruct ( )`

Destructor.

Definition at line 36 of file [RMEventStruct.cpp](#).

## 32.131.3 Member Function Documentation

### 32.131.3.1 `const AirlineCode_T& stdair::RMEventStruct::getAirlineCode ( ) const [inline]`

Get the airline code.

Definition at line 23 of file [RMEventStruct.hpp](#).

### 32.131.3.2 `const KeyDescription_T& stdair::RMEventStruct::getFlightDateDescription ( ) const [inline]`

Get the string describing the flight-date key.

Definition at line 28 of file [RMEventStruct.hpp](#).

### 32.131.3.3 `const DateTime_T& stdair::RMEventStruct::getRMEventTime ( ) const [inline]`

Get the snapshot action time.

Definition at line 33 of file [RMEventStruct.hpp](#).

### 32.131.3.4 `void stdair::RMEventStruct::toStream ( std::ostream & ioOut ) const`

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Definition at line 40 of file [RMEventStruct.cpp](#).

References [describe\(\)](#).

### 32.131.3.5 `void stdair::RMEventStruct::fromStream ( std::istream & ioIn ) [virtual]`

Read a Business Object from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented from [stdair::StructAbstract](#).

Definition at line 45 of file [RMEventStruct.cpp](#).

### 32.131.3.6 `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:

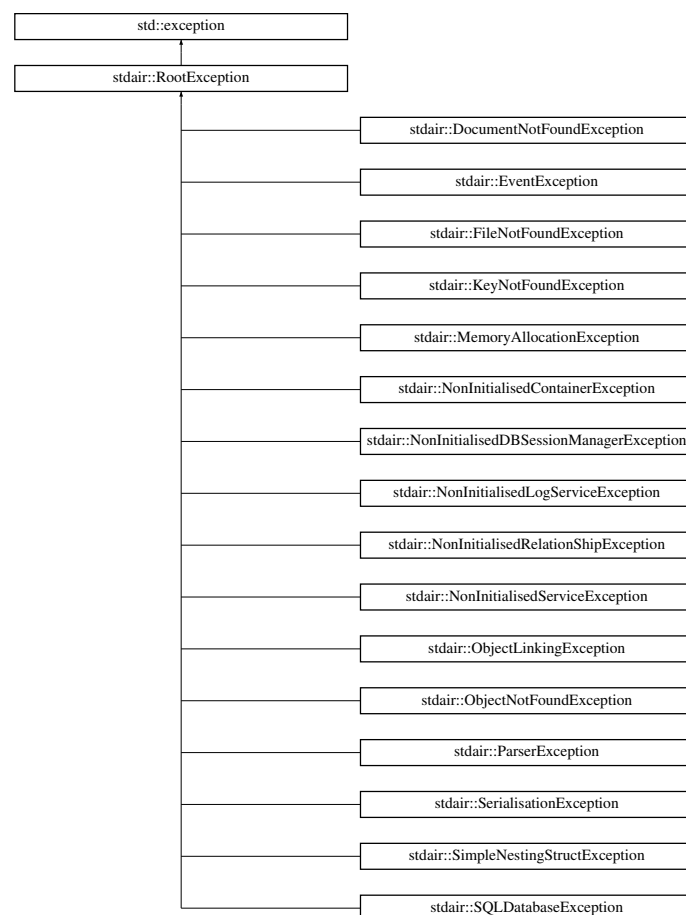
- [stdair/bom/RMEventStruct.hpp](#)
- [stdair/bom/RMEventStruct.cpp](#)

## 32.132 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](#)

### 32.132.1 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](#).

### 32.132.2 Constructor & Destructor Documentation

#### 32.132.2.1 `stdair::RootException::RootException ( const std::string & iWhat ) [inline]`

Main Constructor.

Definition at line 24 of file [stdair\\_exceptions.hpp](#).

#### 32.132.2.2 `stdair::RootException::RootException ( ) [inline]`

Default constructor.

Definition at line 28 of file [stdair\\_exceptions.hpp](#).

#### 32.132.2.3 `virtual stdair::RootException::~~RootException ( ) throw ) [inline],[virtual]`

Destructor.

Definition at line 33 of file [stdair\\_exceptions.hpp](#).

### 32.132.3 Member Function Documentation

#### 32.132.3.1 `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\(\)](#).

### 32.132.4 Member Data Documentation

#### 32.132.4.1 `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:

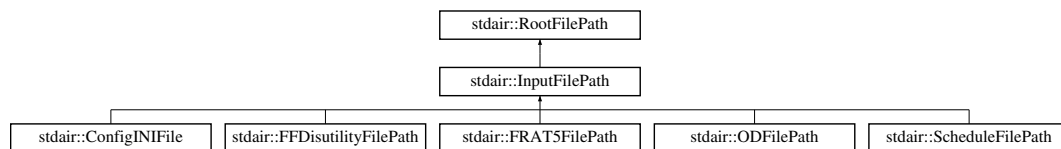
- [stdair/stdair\\_exceptions.hpp](#)

## 32.133 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

#### 32.133.1 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](#).

#### 32.133.2 Constructor & Destructor Documentation

**32.133.2.1** `stdair::RootFilePath::RootFilePath ( const Filename\_T &iFilename ) [inline]`

Main Constructor.

Definition at line 27 of file [stdair\\_file.hpp](#).

**32.133.2.2** `stdair::RootFilePath::RootFilePath ( ) [inline]`

Default constructor.

Definition at line 32 of file [stdair\\_file.hpp](#).

**32.133.2.3** `virtual stdair::RootFilePath::~~RootFilePath ( ) [inline],[virtual]`

Destructor.

Definition at line 37 of file [stdair\\_file.hpp](#).

#### 32.133.3 Member Function Documentation

**32.133.3.1** `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\(\)](#).

### 32.133.4 Member Data Documentation

#### 32.133.4.1 `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:

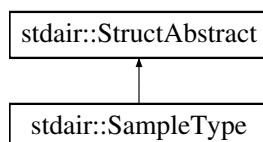
- [stdair/stdair\\_file.hpp](#)

### 32.134 `stdair::SampleType` Struct Reference

Enumeration of BOM sample types.

```
#include <stdair/basic/SampleType.hpp>
```

Inheritance diagram for `stdair::SampleType`:



#### Public Types

- enum [EN\\_SampleType](#) {  
[ALL](#) = 0, [A4P](#), [RMS](#), [INV](#),  
[SCH](#), [RAC](#), [FQT](#), [CRS](#),  
[DEM](#), [EVT](#), [CCM](#), [LAST\\_VALUE](#) }

#### Public Member Functions

- [EN\\_SampleType](#) [getType](#) () const
- std::string [getTypeAsString](#) () const
- const std::string [describe](#) () const
- bool [operator==](#) (const [EN\\_SampleType](#) &) const
- [SampleType](#) (const [EN\\_SampleType](#) &)
- [SampleType](#) (const char iType)
- [SampleType](#) (const [SampleType](#) &)
- void [toStream](#) (std::ostream &ioOut) const
- virtual void [fromStream](#) (std::istream &ioIn)

#### Static Public Member Functions

- static const std::string & [getLabel](#) (const [EN\\_SampleType](#) &)
- static char [getTypeLabel](#) (const [EN\\_SampleType](#) &)
- static std::string [getTypeLabelAsString](#) (const [EN\\_SampleType](#) &)
- static std::string [describeLabels](#) ()

### 32.134.1 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](#).

### 32.134.2 Member Enumeration Documentation

#### 32.134.2.1 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](#).

### 32.134.3 Constructor & Destructor Documentation

#### 32.134.3.1 stdair::SampleType::SampleType ( const EN\_SampleType & iSampleType )

Constructor.

Definition at line 36 of file [SampleType.cpp](#).

#### 32.134.3.2 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](#).

#### 32.134.3.3 stdair::SampleType::SampleType ( const SampleType & iSampleType )

Default copy constructor.

Definition at line 31 of file [SampleType.cpp](#).

### 32.134.4 Member Function Documentation

**32.134.4.1** `const std::string & stdair::SampleType::getLabel ( const EN_SampleType & iType ) [static]`

Get the label as a string (e.g., "Inventory" or "Schedule").

Definition at line 67 of file [SampleType.cpp](#).

**32.134.4.2** `char stdair::SampleType::getTypeLabel ( const EN_SampleType & iType ) [static]`

Get the label as a single char (e.g., 'I' or 'S').

Definition at line 72 of file [SampleType.cpp](#).

**32.134.4.3** `std::string stdair::SampleType::getTypeLabelAsString ( const EN_SampleType & iType ) [static]`

Get the label as a string of a single char (e.g., "I" or "S").

Definition at line 77 of file [SampleType.cpp](#).

**32.134.4.4** `std::string stdair::SampleType::describeLabels ( ) [static]`

List the labels.

Definition at line 84 of file [SampleType.cpp](#).

References [LAST\\_VALUE](#).

Referenced by [SampleType\(\)](#).

**32.134.4.5** `SampleType::EN_SampleType stdair::SampleType::getType ( ) const`

Get the enumerated value.

Definition at line 96 of file [SampleType.cpp](#).

**32.134.4.6** `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](#).

**32.134.4.7** `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](#).

**32.134.4.8** `bool stdair::SampleType::operator== ( const EN_SampleType & iType ) const`

Comparison operator.

Definition at line 115 of file [SampleType.cpp](#).

**32.134.4.9** `void stdair::StructAbstract::toStream ( std::ostream & ioOut ) const [inline],[inherited]`

Dump a Business Object into an output stream.

Parameters

<code>ostream&amp;</code>	the output stream.
---------------------------	--------------------

Definition at line 29 of file [StructAbstract.hpp](#).

References [stdair::StructAbstract::describe\(\)](#).

32.134.4.10 virtual void stdair::StructAbstract::fromStream ( std::istream & *ioIn* ) [inline],[virtual],  
[inherited]

Read a Business Object from an input stream.



## Parameters

<i>istream</i> &	the input stream.
------------------	-------------------

Reimplemented in [stdair::BookingRequestStruct](#), [stdair::EventStruct](#), [stdair::TravelSolutionStruct](#), [stdair::VirtualClassStruct](#), [stdair::ConfigHolderStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::ProgressStatusSet](#), [stdair::YieldRange](#), [stdair::CancellationStruct](#), [stdair::AirlineStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::FFDisutilityCurveHolderStruct](#), [stdair::FRAT5CurveHolderStruct](#), and [stdair::BreakPointStruct](#).

Definition at line 38 of file [StructAbstract.hpp](#).

Referenced by [operator>>\(\)](#).

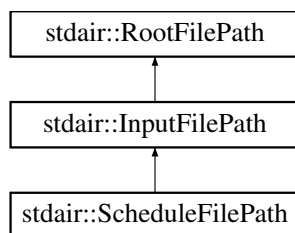
The documentation for this struct was generated from the following files:

- [stdair/basic/SampleType.hpp](#)
- [stdair/basic/SampleType.cpp](#)

### 32.135 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](#)

#### 32.135.1 Detailed Description

Schedule input file.

Definition at line 64 of file [stdair\\_file.hpp](#).

#### 32.135.2 Constructor & Destructor Documentation

**32.135.2.1** [stdair::ScheduleFilePath::ScheduleFilePath](#) ( const [Filename\\_T](#) &*iFilename* ) [\[inline\]](#), [\[explicit\]](#)

Constructor.

Definition at line 69 of file [stdair\\_file.hpp](#).

## 32.135.3 Member Function Documentation

32.135.3.1 `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\(\)](#).

## 32.135.4 Member Data Documentation

32.135.4.1 `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:

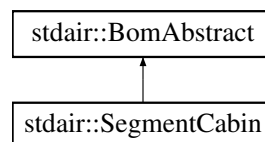
- [stdair/stdair\\_file.hpp](#)

## 32.136 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

- template<typename BOM >  
class [FacBom](#)
- template<typename BOM >  
class [FacCloneBom](#)
- class [FacBomManager](#)
- class [boost::serialization::access](#)

## 32.136.1 Detailed Description

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

Definition at line 33 of file [SegmentCabin.hpp](#).

## 32.136.2 Member Typedef Documentation

## 32.136.2.1 typedef SegmentCabinKey stdair::SegmentCabin::Key\_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 44 of file [SegmentCabin.hpp](#).

## 32.136.3 Constructor &amp; Destructor Documentation

## 32.136.3.1 stdair::SegmentCabin::SegmentCabin ( const Key\_T &amp; iKey ) [protected]

Constructor.

Definition at line 39 of file [SegmentCabin.cpp](#).

## 32.136.3.2 stdair::SegmentCabin::~~SegmentCabin ( ) [protected], [virtual]

Destructor.

Definition at line 52 of file [SegmentCabin.cpp](#).

## 32.136.4 Member Function Documentation

## 32.136.4.1 const Key\_T&amp; stdair::SegmentCabin::getKey ( ) const [inline]

Get the segment-cabin key (cabin code).

Definition at line 52 of file [SegmentCabin.hpp](#).

References [\\_key](#).

## 32.136.4.2 BomAbstract\* const stdair::SegmentCabin::getParent ( ) const [inline]

Get the parent object.

Definition at line 59 of file [SegmentCabin.hpp](#).

References [\\_parent](#).

## 32.136.4.3 const HolderMap\_T&amp; stdair::SegmentCabin::getHolderMap ( ) const [inline]

Get the map of children holders.

Definition at line 66 of file [SegmentCabin.hpp](#).

References [\\_holderMap](#).

**32.136.4.4** `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\(\)](#).

**32.136.4.5** `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\(\)](#).

**32.136.4.6** `const SegmentSnapshotTable& stdair::SegmentCabin::getSegmentSnapshotTable ( ) const` `[inline]`

Get the guilltine block.

Definition at line 88 of file [SegmentCabin.hpp](#).

References [\\_segmentSnapshotTable](#).

**32.136.4.7** `const CabinCapacity_T& stdair::SegmentCabin::getCapacity ( ) const` `[inline]`

Get the cabin capacity.

Definition at line 94 of file [SegmentCabin.hpp](#).

References [\\_capacity](#).

**32.136.4.8** `const BlockSpace_T& stdair::SegmentCabin::getBlockSpace ( ) const` `[inline]`

Get the blocked number of bookings.

Definition at line 99 of file [SegmentCabin.hpp](#).

References [\\_blockSpace](#).

**32.136.4.9** `const BlockSpace_T& stdair::SegmentCabin::getMIN ( ) const` `[inline]`

Get the blocked number of bookings.

Definition at line 104 of file [SegmentCabin.hpp](#).

References [\\_min](#).

**32.136.4.10** `const UPR_T& stdair::SegmentCabin::getUPR ( ) const` `[inline]`

Unsold Protection (UPR).

Definition at line 109 of file [SegmentCabin.hpp](#).

References [\\_upr](#).

**32.136.4.11** `const NbOfBookings_T& stdair::SegmentCabin::getBookingCounter ( ) const` `[inline]`

Get the booking counter.

Definition at line 114 of file [SegmentCabin.hpp](#).

References [\\_bookingCounter](#).

32.136.4.12 `const CommittedSpace_T& stdair::SegmentCabin::getCommittedSpace ( ) const` `[inline]`

Get the committed Space value.

Definition at line 119 of file [SegmentCabin.hpp](#).

References [\\_committedSpace](#).

32.136.4.13 `const Availability_T& stdair::SegmentCabin::getAvailabilityPool ( ) const` `[inline]`

Get the availability pool value.

Definition at line 124 of file [SegmentCabin.hpp](#).

References [\\_availabilityPool](#).

32.136.4.14 `const BidPrice_T& stdair::SegmentCabin::getCurrentBidPrice ( ) const` `[inline]`

Retrieve the current Bid-Price.

Definition at line 129 of file [SegmentCabin.hpp](#).

References [\\_currentBidPrice](#).

32.136.4.15 `const BidPriceVector_T& stdair::SegmentCabin::getBidPriceVector ( ) const` `[inline]`

Retrieve the Bid-Price Vector.

Definition at line 134 of file [SegmentCabin.hpp](#).

References [\\_bidPriceVector](#).

32.136.4.16 `const bool stdair::SegmentCabin::getFareFamilyStatus ( ) const` `[inline]`

Retrieve the status of fare family.

Definition at line 139 of file [SegmentCabin.hpp](#).

References [\\_fareFamilyActivation](#).

32.136.4.17 `const PolicyList_T& stdair::SegmentCabin::getConvexHull ( ) const` `[inline]`

Retrieve the convex hull.

Definition at line 144 of file [SegmentCabin.hpp](#).

References [\\_convexHull](#).

32.136.4.18 `void stdair::SegmentCabin::setSegmentSnapshotTable ( SegmentSnapshotTable & ioTable )` `[inline]`

Set the snapshot table.

Definition at line 151 of file [SegmentCabin.hpp](#).

References [\\_segmentSnapshotTable](#).

32.136.4.19 `void stdair::SegmentCabin::setCapacity ( const CabinCapacity_T & iCapacity )` `[inline]`

Set the cabin capacity.

Definition at line 156 of file [SegmentCabin.hpp](#).

References [\\_capacity](#).

32.136.4.20 `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](#).

32.136.4.21 `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](#).

32.136.4.22 `void stdair::SegmentCabin::setUPR ( const UPR_T & iUPR ) [inline]`

Set the Unsold Protection (UPR).

Definition at line 171 of file [SegmentCabin.hpp](#).

References [\\_upr](#).

32.136.4.23 `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](#).

32.136.4.24 `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](#).

32.136.4.25 `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](#).

32.136.4.26 `void stdair::SegmentCabin::setBidPriceVector ( const BidPriceVector_T & iBPV ) [inline]`

Set the Bid-Price Vector.

Definition at line 191 of file [SegmentCabin.hpp](#).

References [\\_bidPriceVector](#).

32.136.4.27 `void stdair::SegmentCabin::activateFareFamily ( ) [inline]`

Activate fare family.

Definition at line 196 of file [SegmentCabin.hpp](#).

References [\\_fareFamilyActivation](#).

32.136.4.28 `void stdair::SegmentCabin::updateFromReservation ( const NbOfBookings_T & iNbOfBookings )`

Register a sale.

Definition at line 85 of file [SegmentCabin.cpp](#).

References [\\_committedSpace](#).

**32.136.4.29** void stdair::SegmentCabin::resetConvexHull ( ) [inline]

Reset the convex hull.

Definition at line 206 of file [SegmentCabin.hpp](#).

References [\\_convexHull](#).

**32.136.4.30** void stdair::SegmentCabin::addPolicy ( Policy & ioPolicy )

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](#).

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

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Implements [stdair::BomAbstract](#).

Definition at line 221 of file [SegmentCabin.hpp](#).

References [toString\(\)](#).

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

Read a Business Object from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Implements [stdair::BomAbstract](#).

Definition at line 230 of file [SegmentCabin.hpp](#).

**32.136.4.33** 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\(\)](#).

**32.136.4.34** 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\(\)](#).

**32.136.4.35** const std::string stdair::SegmentCabin::describeConvexHull ( ) const

Get a string describing the convex hull.

Definition at line 72 of file [SegmentCabin.cpp](#).



References [\\_convexHull](#), and [stdair::Policy::toString\(\)](#).

32.136.4.36 `template<class Archive > void stdair::SegmentCabin::serialize ( Archive & ar, const unsigned int iFileVersion )`

Serialisation.

Definition at line 229 of file [CmdBomSerialiser.cpp](#).

References [\\_key](#).

### 32.136.5 Friends And Related Function Documentation

32.136.5.1 `template<typename BOM > friend class FacBom [friend]`

Definition at line 34 of file [SegmentCabin.hpp](#).

32.136.5.2 `template<typename BOM > friend class FacCloneBom [friend]`

Definition at line 35 of file [SegmentCabin.hpp](#).

32.136.5.3 `friend class FacBomManager [friend]`

Definition at line 36 of file [SegmentCabin.hpp](#).

32.136.5.4 `friend class boost::serialization::access [friend]`

Definition at line 37 of file [SegmentCabin.hpp](#).

### 32.136.6 Member Data Documentation

32.136.6.1 `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\(\)](#).

32.136.6.2 `BomAbstract* stdair::SegmentCabin::_parent [protected]`

Pointer on the parent class ([SegmentDate](#)).

Definition at line 305 of file [SegmentCabin.hpp](#).

Referenced by [getParent\(\)](#).

32.136.6.3 `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\(\)](#).

32.136.6.4 `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\(\)](#).

**32.136.6.5 CabinCapacity\_T** stdair::SegmentCabin::\_capacity [protected]

Capacity of the cabin.

Definition at line 318 of file [SegmentCabin.hpp](#).

Referenced by [getCapacity\(\)](#), and [setCapacity\(\)](#).

**32.136.6.6 BlockSpace\_T** stdair::SegmentCabin::\_blockSpace [protected]

Blocked capacity.

Definition at line 321 of file [SegmentCabin.hpp](#).

Referenced by [getBlockSpace\(\)](#), and [setBlockSpace\(\)](#).

**32.136.6.7 BlockSpace\_T** stdair::SegmentCabin::\_min [protected]

Blocked number of seats.

Definition at line 324 of file [SegmentCabin.hpp](#).

Referenced by [getMIN\(\)](#), and [setMIN\(\)](#).

**32.136.6.8 UPR\_T** stdair::SegmentCabin::\_upr [protected]

Unsold Protection (UPR).

Definition at line 327 of file [SegmentCabin.hpp](#).

Referenced by [getUPR\(\)](#), and [setUPR\(\)](#).

**32.136.6.9 NbOfBookings\_T** stdair::SegmentCabin::\_bookingCounter [protected]

Aggregated number of bookings.

Definition at line 330 of file [SegmentCabin.hpp](#).

Referenced by [getBookingCounter\(\)](#), and [setBookingCounter\(\)](#).

**32.136.6.10 CommittedSpace\_T** stdair::SegmentCabin::\_committedSpace [protected]

Aggregated committed space.

Definition at line 333 of file [SegmentCabin.hpp](#).

Referenced by [getCommittedSpace\(\)](#), [setCommittedSpace\(\)](#), and [updateFromReservation\(\)](#).

**32.136.6.11 Availability\_T** stdair::SegmentCabin::\_availabilityPool [protected]

Aggregated availability pool.

Definition at line 336 of file [SegmentCabin.hpp](#).

Referenced by [getAvailabilityPool\(\)](#), and [setAvailabilityPool\(\)](#).

**32.136.6.12 BidPriceVector\_T** stdair::SegmentCabin::\_bidPriceVector [protected]

Bid-Price Vector (BPV).

Definition at line 339 of file [SegmentCabin.hpp](#).

Referenced by [getBidPriceVector\(\)](#), and [setBidPriceVector\(\)](#).

**32.136.6.13 BidPrice\_T** stdair::SegmentCabin::\_currentBidPrice [protected]

Current Bid-Price (BP).

Definition at line 342 of file [SegmentCabin.hpp](#).

Referenced by [getCurrentBidPrice\(\)](#).

**32.136.6.14** `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\(\)](#).

**32.136.6.15** `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:

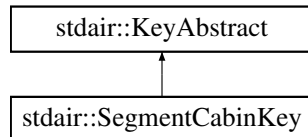
- [stdair/bom/SegmentCabin.hpp](#)
- [stdair/bom/SegmentCabin.cpp](#)
- [stdair/command/CmdBomSerialiser.cpp](#)

## 32.137 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](#)

### 32.137.1 Detailed Description

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

Definition at line 26 of file [SegmentCabinKey.hpp](#).

## 32.137.2 Constructor &amp; Destructor Documentation

## 32.137.2.1 stdair::SegmentCabinKey::SegmentCabinKey ( const CabinCode\_T &amp; iCabinCode )

Constructor.

Definition at line 23 of file [SegmentCabinKey.cpp](#).

## 32.137.2.2 stdair::SegmentCabinKey::SegmentCabinKey ( const SegmentCabinKey &amp; iKey )

Copy constructor.

Definition at line 28 of file [SegmentCabinKey.cpp](#).

## 32.137.2.3 stdair::SegmentCabinKey::~SegmentCabinKey ( )

Destructor.

Definition at line 33 of file [SegmentCabinKey.cpp](#).

## 32.137.3 Member Function Documentation

## 32.137.3.1 const CabinCode\_T&amp; stdair::SegmentCabinKey::getCabinCode ( ) const [inline]

Get the cabin code.

Definition at line 56 of file [SegmentCabinKey.hpp](#).

Referenced by [stdair::SegmentCabin::getCabinCode\(\)](#).

## 32.137.3.2 void stdair::SegmentCabinKey::toStream ( std::ostream &amp; ioOut ) const [virtual]

Dump a Business Object Key into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 37 of file [SegmentCabinKey.cpp](#).

References [toString\(\)](#).

## 32.137.3.3 void stdair::SegmentCabinKey::fromStream ( std::istream &amp; ioln ) [virtual]

Read a Business Object Key from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 42 of file [SegmentCabinKey.cpp](#).

## 32.137.3.4 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\(\)](#), [stdair::BomRetriever::retrieveDummySegmentCabin\(\)](#), and [toStream\(\)](#).

32.137.3.5 `template<class Archive > void stdair::SegmentCabinKey::serialize ( Archive & ar, const unsigned int iFileVersion )`

Serialisation.

Definition at line 68 of file [SegmentCabinKey.cpp](#).

### 32.137.4 Friends And Related Function Documentation

32.137.4.1 `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:

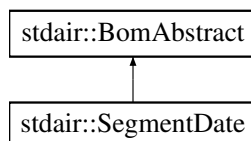
- [stdair/bom/SegmentCabinKey.hpp](#)
- [stdair/bom/SegmentCabinKey.cpp](#)

## 32.138 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

- template<typename BOM >  
class [FacBom](#)
- template<typename BOM >  
class [FacCloneBom](#)
- class [FacBomManager](#)
- class [boost::serialization::access](#)

#### 32.138.1 Detailed Description

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

Definition at line 36 of file [SegmentDate.hpp](#).

#### 32.138.2 Member Typedef Documentation

##### 32.138.2.1 typedef [SegmentDateKey](#) stdair::SegmentDate::Key\_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 47 of file [SegmentDate.hpp](#).

### 32.138.3 Constructor & Destructor Documentation

#### 32.138.3.1 `stdair::SegmentDate::SegmentDate ( const Key_T & iKey ) [protected]`

Constructor.

Definition at line 38 of file [SegmentDate.cpp](#).

#### 32.138.3.2 `stdair::SegmentDate::~~SegmentDate ( ) [protected],[virtual]`

Destructor.

Definition at line 44 of file [SegmentDate.cpp](#).

### 32.138.4 Member Function Documentation

#### 32.138.4.1 `const Key_T& stdair::SegmentDate::getKey ( ) const [inline]`

Get the segment-date key.

Definition at line 55 of file [SegmentDate.hpp](#).

References [\\_key](#).

#### 32.138.4.2 `BomAbstract* const stdair::SegmentDate::getParent ( ) const [inline]`

Get the parent object.

Definition at line 62 of file [SegmentDate.hpp](#).

References [\\_parent](#).

#### 32.138.4.3 `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\(\)](#).

#### 32.138.4.4 `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\(\)](#).

#### 32.138.4.5 `const HolderMap_T& stdair::SegmentDate::getHolderMap ( ) const [inline]`

Get the map of children holders.

Definition at line 83 of file [SegmentDate.hpp](#).

References [\\_holderMap](#).

#### 32.138.4.6 `const Date_T& stdair::SegmentDate::getBoardingDate ( ) const [inline]`

Get the boarding date.

Definition at line 90 of file [SegmentDate.hpp](#).

References [\\_boardingDate](#).

#### 32.138.4.7 `const Duration_T& stdair::SegmentDate::getBoardingTime ( ) const [inline]`

Get the boarding time.

Definition at line 97 of file [SegmentDate.hpp](#).

References [\\_boardingTime](#).

**32.138.4.8** `const Date_T& stdair::SegmentDate::getOffDate ( ) const [inline]`

Get the off date.

Definition at line 104 of file [SegmentDate.hpp](#).

References [\\_offDate](#).

**32.138.4.9** `const Duration_T& stdair::SegmentDate::getOffTime ( ) const [inline]`

Get the off time.

Definition at line 111 of file [SegmentDate.hpp](#).

References [\\_offTime](#).

**32.138.4.10** `const Duration_T& stdair::SegmentDate::getElapsedTime ( ) const [inline]`

Get the elapsed time.

Definition at line 118 of file [SegmentDate.hpp](#).

References [\\_elapsedTime](#).

**32.138.4.11** `const Distance_T& stdair::SegmentDate::getDistance ( ) const [inline]`

Get the distance.

Definition at line 125 of file [SegmentDate.hpp](#).

References [\\_distance](#).

**32.138.4.12** `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\(\)](#).

**32.138.4.13** `const Duration_T stdair::SegmentDate::getTimeOffset ( ) const`

Get the time offset between boarding and off points.

It is defined as being:

$\text{TimeOffset} = (\text{OffTime} - \text{BoardingTime}) + (\text{OffDate} - \text{BoardingDate}) * 24$

- [ElapsedTime](#).

Definition at line 55 of file [SegmentDate.cpp](#).

References [\\_boardingTime](#), [\\_elapsedTime](#), [\\_offTime](#), and [getDateOffset\(\)](#).

**32.138.4.14** `SegmentDate* stdair::SegmentDate::getOperatingSegmentDate ( ) const [inline]`

Get the "operating" segment date.

Definition at line 149 of file [SegmentDate.hpp](#).

References [\\_operatingSegmentDate](#).



32.138.4.15 `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](#).

32.138.4.16 `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](#).

32.138.4.17 `void stdair::SegmentDate::setBoardingDate ( const Date_T & iBoardingDate )` `[inline]`

Set the boarding date.

Definition at line 172 of file [SegmentDate.hpp](#).

References [\\_boardingDate](#).

32.138.4.18 `void stdair::SegmentDate::setBoardingTime ( const Duration_T & iBoardingTime )` `[inline]`

Set the boarding time.

Definition at line 179 of file [SegmentDate.hpp](#).

References [\\_boardingTime](#).

32.138.4.19 `void stdair::SegmentDate::setOffDate ( const Date_T & iOffDate )` `[inline]`

Set the off date.

Definition at line 186 of file [SegmentDate.hpp](#).

References [\\_offDate](#).

32.138.4.20 `void stdair::SegmentDate::setOffTime ( const Duration_T & iOffTime )` `[inline]`

Set the off time.

Definition at line 193 of file [SegmentDate.hpp](#).

References [\\_offTime](#).

32.138.4.21 `void stdair::SegmentDate::setElapsedTime ( const Duration_T & iElapsedTime )` `[inline]`

Set the elapsed time.

Definition at line 200 of file [SegmentDate.hpp](#).

References [\\_elapsedTime](#).

32.138.4.22 `void stdair::SegmentDate::setDistance ( const Distance_T & iDistance )` `[inline]`

Set the distance.

Definition at line 207 of file [SegmentDate.hpp](#).

References [\\_distance](#).

32.138.4.23 `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](#).

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

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Implements [stdair::BomAbstract](#).

Definition at line 233 of file [SegmentDate.hpp](#).

References [toString\(\)](#).

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

Read a Business Object from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Implements [stdair::BomAbstract](#).

Definition at line 242 of file [SegmentDate.hpp](#).

**32.138.4.26** 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 [toStream\(\)](#).

**32.138.4.27** 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\(\)](#).

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

Serialisation.

Definition at line 208 of file [CmdBomSerialiser.cpp](#).

References [\\_key](#).

## 32.138.5 Friends And Related Function Documentation

**32.138.5.1** template<typename BOM > friend class FacBom [friend]

Definition at line 37 of file [SegmentDate.hpp](#).

**32.138.5.2** template<typename BOM > friend class FacCloneBom [friend]

Definition at line 38 of file [SegmentDate.hpp](#).

### 32.138.5.3 friend class **FacBomManager** [friend]

Definition at line 39 of file [SegmentDate.hpp](#).

### 32.138.5.4 friend class **boost::serialization::access** [friend]

Definition at line 40 of file [SegmentDate.hpp](#).

## 32.138.6 Member Data Documentation

### 32.138.6.1 **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\(\)](#).

### 32.138.6.2 **BomAbstract\*** **stdair::SegmentDate::\_parent** [protected]

Pointer on the parent class ([FlightDate](#)).

Definition at line 312 of file [SegmentDate.hpp](#).

Referenced by [getParent\(\)](#).

### 32.138.6.3 **HolderMap\_T** **stdair::SegmentDate::\_holderMap** [protected]

Map holding the children ([SegmentCabin](#) objects).

Definition at line 317 of file [SegmentDate.hpp](#).

Referenced by [getHolderMap\(\)](#).

### 32.138.6.4 **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\(\)](#).

### 32.138.6.5 **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 [stdair::FacBomManager::addToList\(\)](#), and [getMarketingSegmentDateList\(\)](#).

### 32.138.6.6 **Date\_T** **stdair::SegmentDate::\_boardingDate** [protected]

Boarding date.

Definition at line 338 of file [SegmentDate.hpp](#).

Referenced by [getBoardingDate\(\)](#), [getDateOffset\(\)](#), and [setBoardingDate\(\)](#).

**32.138.6.7** `Duration_T stdair::SegmentDate::_boardingTime` [protected]

Boarding time.

Definition at line 343 of file [SegmentDate.hpp](#).

Referenced by [getBoardingTime\(\)](#), [getTimeOffset\(\)](#), and [setBoardingTime\(\)](#).

**32.138.6.8** `Date_T stdair::SegmentDate::_offDate` [protected]

Landing date.

Definition at line 348 of file [SegmentDate.hpp](#).

Referenced by [getDateOffset\(\)](#), [getOffDate\(\)](#), and [setOffDate\(\)](#).

**32.138.6.9** `Duration_T stdair::SegmentDate::_offTime` [protected]

Landing time.

Definition at line 353 of file [SegmentDate.hpp](#).

Referenced by [getOffTime\(\)](#), [getTimeOffset\(\)](#), and [setOffTime\(\)](#).

**32.138.6.10** `Duration_T stdair::SegmentDate::_elapsedTime` [protected]

Trip elapsed time.

Definition at line 358 of file [SegmentDate.hpp](#).

Referenced by [getElapsedTime\(\)](#), [getTimeOffset\(\)](#), and [setElapsedTime\(\)](#).

**32.138.6.11** `Distance_T stdair::SegmentDate::_distance` [protected]

Trip distance.

Definition at line 363 of file [SegmentDate.hpp](#).

Referenced by [getDistance\(\)](#), and [setDistance\(\)](#).

**32.138.6.12** `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:

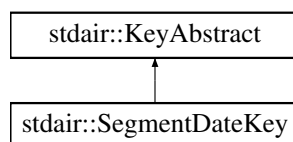
- [stdair/bom/SegmentDate.hpp](#)
- [stdair/bom/SegmentDate.cpp](#)
- [stdair/command/CmdBomSerialiser.cpp](#)

**32.139** `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](#)

### 32.139.1 Detailed Description

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

Definition at line 24 of file [SegmentDateKey.hpp](#).

### 32.139.2 Constructor & Destructor Documentation

#### 32.139.2.1 stdair::SegmentDateKey::SegmentDateKey ( const [AirportCode\\_T](#) & *iBoardingPoint*, const [AirportCode\\_T](#) & *iOffPoint* )

Main constructor.

Definition at line 25 of file [SegmentDateKey.cpp](#).

#### 32.139.2.2 stdair::SegmentDateKey::SegmentDateKey ( const [SegmentDateKey](#) & *iKey* )

Copy constructor.

Definition at line 31 of file [SegmentDateKey.cpp](#).

#### 32.139.2.3 stdair::SegmentDateKey::~~SegmentDateKey ( )

Destructor.

Definition at line 36 of file [SegmentDateKey.cpp](#).

### 32.139.3 Member Function Documentation

#### 32.139.3.1 const [AirportCode\\_T](#)& stdair::SegmentDateKey::getBoardingPoint ( ) const [inline]

Get the boarding point.

Definition at line 51 of file [SegmentDateKey.hpp](#).

Referenced by [stdair::SegmentDate::getBoardingPoint\(\)](#), and [stdair::OnDDateKey::getOrigin\(\)](#).

**32.139.3.2** `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\(\)](#).

**32.139.3.3** `void stdair::SegmentDateKey::toStream ( std::ostream & ioOut ) const [virtual]`

Dump a Business Object Key into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 40 of file [SegmentDateKey.cpp](#).

References [toString\(\)](#).

**32.139.3.4** `void stdair::SegmentDateKey::fromStream ( std::istream & ioin ) [virtual]`

Read a Business Object Key from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 45 of file [SegmentDateKey.cpp](#).

**32.139.3.5** `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\(\)](#).

**32.139.3.6** `template<class Archive > void stdair::SegmentDateKey::serialize ( Archive & ar, const unsigned int iFileVersion )`

Serialisation.

Definition at line 72 of file [SegmentDateKey.cpp](#).

## 32.139.4 Friends And Related Function Documentation

**32.139.4.1** `friend class boost::serialization::access [friend]`

Definition at line 25 of file [SegmentDateKey.hpp](#).

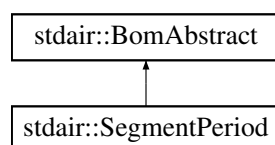
The documentation for this struct was generated from the following files:

- [stdair/bom/SegmentDateKey.hpp](#)
- [stdair/bom/SegmentDateKey.cpp](#)

### 32.140 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

- `template<typename BOM >`  
class [FacBom](#)
- `template<typename BOM >`  
class [FacCloneBom](#)
- class [FacBomManager](#)

## 32.140.1 Detailed Description

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

Definition at line 15 of file [SegmentPeriod.hpp](#).

## 32.140.2 Member Typedef Documentation

32.140.2.1 `typedef SegmentPeriodKey stdair::SegmentPeriod::Key_T`

Definition allowing to retrieve the associated BOM key type.

Definition at line 23 of file [SegmentPeriod.hpp](#).

## 32.140.3 Constructor &amp; Destructor Documentation

32.140.3.1 `stdair::SegmentPeriod::SegmentPeriod ( const Key_T & iKey ) [protected]`

Main constructor.

Definition at line 13 of file [SegmentPeriod.cpp](#).

32.140.3.2 `stdair::SegmentPeriod::~~SegmentPeriod ( ) [protected],[virtual]`

Destructor.

Definition at line 29 of file [SegmentPeriod.cpp](#).

## 32.140.4 Member Function Documentation

32.140.4.1 `const Key_T& stdair::SegmentPeriod::getKey ( ) const [inline]`

Get the segment-period key.

Definition at line 28 of file [SegmentPeriod.hpp](#).

References [\\_key](#).



**32.140.4.2 BomAbstract\*** `const stdair::SegmentPeriod::getParent ( ) const` `[inline]`

Get the parent object.

Definition at line 31 of file [SegmentPeriod.hpp](#).

References [\\_parent](#).

**32.140.4.3 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\(\)](#).

**32.140.4.4 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\(\)](#).

**32.140.4.5 const Duration\_T&** `stdair::SegmentPeriod::getBoardingTime ( ) const` `[inline]`

Get the boarding time.

Definition at line 42 of file [SegmentPeriod.hpp](#).

References [\\_boardingTime](#).

**32.140.4.6 const Duration\_T&** `stdair::SegmentPeriod::getOffTime ( ) const` `[inline]`

Get the off time.

Definition at line 45 of file [SegmentPeriod.hpp](#).

References [\\_offTime](#).

**32.140.4.7 const DateOffset\_T&** `stdair::SegmentPeriod::getBoardingDateOffset ( ) const` `[inline]`

Get the boarding date offset.

Definition at line 48 of file [SegmentPeriod.hpp](#).

References [\\_boardingDateOffset](#).

**32.140.4.8 const DateOffset\_T&** `stdair::SegmentPeriod::getOffDateOffset ( ) const` `[inline]`

Get the off date offset.

Definition at line 53 of file [SegmentPeriod.hpp](#).

References [\\_offDateOffset](#).

**32.140.4.9 const Duration\_T&** `stdair::SegmentPeriod::getElapsedTime ( ) const` `[inline]`

Get the elapsed time.

Definition at line 56 of file [SegmentPeriod.hpp](#).

References [\\_elapsedTime](#).

**32.140.4.10 const CabinBookingClassMap\_T&** `stdair::SegmentPeriod::getCabinBookingClassMap ( ) const`  
`[inline]`

Get the cabin booking class map.

Definition at line 59 of file [SegmentPeriod.hpp](#).

References [\\_cabinBookingClassMap](#).

32.140.4.11 `const HolderMap_T& stdair::SegmentPeriod::getHolderMap ( ) const` `[inline]`

Get the map of children holders.

Definition at line 64 of file [SegmentPeriod.hpp](#).

References [\\_holderMap](#).

32.140.4.12 `void stdair::SegmentPeriod::setBoardingTime ( const Duration_T & iBoardingTime )` `[inline]`

Set the boarding time.

Definition at line 69 of file [SegmentPeriod.hpp](#).

References [\\_boardingTime](#).

32.140.4.13 `void stdair::SegmentPeriod::setOffTime ( const Duration_T & iOffTime )` `[inline]`

Set the off time.

Definition at line 74 of file [SegmentPeriod.hpp](#).

References [\\_offTime](#).

32.140.4.14 `void stdair::SegmentPeriod::setBoardingDateOffset ( const DateOffset_T & iDateOffset )` `[inline]`

Set the boarding date offset.

Definition at line 77 of file [SegmentPeriod.hpp](#).

References [\\_boardingDateOffset](#).

32.140.4.15 `void stdair::SegmentPeriod::setOffDateOffset ( const DateOffset_T & iDateOffset )` `[inline]`

Set the off date offset.

Definition at line 82 of file [SegmentPeriod.hpp](#).

References [\\_offDateOffset](#).

32.140.4.16 `void stdair::SegmentPeriod::setElapsedTime ( const Duration_T & iElapsedTime )` `[inline]`

Set the elapsed time.

Definition at line 87 of file [SegmentPeriod.hpp](#).

References [\\_elapsedTime](#).

32.140.4.17 `void stdair::SegmentPeriod::addCabinBookingClassList ( const CabinCode_T & iCabinCode, const ClassList_String_T & iClassCodeList )`

Add a pair cabin code and list of class codes within the cabin to the cabin booking class map.

Definition at line 41 of file [SegmentPeriod.cpp](#).

References [\\_cabinBookingClassMap](#).

32.140.4.18 `void stdair::SegmentPeriod::toStream ( std::ostream & ioOut ) const` `[inline],[virtual]`

Dump a Business Object into an output stream.

## Parameters

<i>ostream</i> &	the output stream.
------------------	--------------------

Implements [stdair::BomAbstract](#).

Definition at line 100 of file [SegmentPeriod.hpp](#).

References [toString\(\)](#).

**32.140.4.19** `void stdair::SegmentPeriod::fromStream ( std::istream & ioln ) [inline], [virtual]`

Read a Business Object from an input stream.

## Parameters

<i>istream</i> &	the input stream.
------------------	-------------------

Implements [stdair::BomAbstract](#).

Definition at line 104 of file [SegmentPeriod.hpp](#).

**32.140.4.20** `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\(\)](#).

**32.140.4.21** `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\(\)](#).

**32.140.5 Friends And Related Function Documentation**

**32.140.5.1** `template<typename BOM > friend class FacBom [friend]`

Definition at line 16 of file [SegmentPeriod.hpp](#).

**32.140.5.2** `template<typename BOM > friend class FacCloneBom [friend]`

Definition at line 17 of file [SegmentPeriod.hpp](#).

**32.140.5.3** `friend class FacBomManager [friend]`

Definition at line 18 of file [SegmentPeriod.hpp](#).

**32.140.6 Member Data Documentation**

**32.140.6.1** `Key_T stdair::SegmentPeriod::_key [protected]`

Definition at line 135 of file [SegmentPeriod.hpp](#).

Referenced by [describeKey\(\)](#), [getBoardingPoint\(\)](#), [getKey\(\)](#), and [getOffPoint\(\)](#).

32.140.6.2 **BomAbstract\*** stdair::SegmentPeriod::\_parent [protected]

Definition at line 136 of file [SegmentPeriod.hpp](#).

Referenced by [getParent\(\)](#).

32.140.6.3 **Duration\_T** stdair::SegmentPeriod::\_boardingTime [protected]

Definition at line 137 of file [SegmentPeriod.hpp](#).

Referenced by [getBoardingTime\(\)](#), and [setBoardingTime\(\)](#).

32.140.6.4 **Duration\_T** stdair::SegmentPeriod::\_offTime [protected]

Definition at line 138 of file [SegmentPeriod.hpp](#).

Referenced by [getOffTime\(\)](#), and [setOffTime\(\)](#).

32.140.6.5 **DateOffset\_T** stdair::SegmentPeriod::\_boardingDateOffset [protected]

Definition at line 139 of file [SegmentPeriod.hpp](#).

Referenced by [getBoardingDateOffset\(\)](#), and [setBoardingDateOffset\(\)](#).

32.140.6.6 **DateOffset\_T** stdair::SegmentPeriod::\_offDateOffset [protected]

Definition at line 140 of file [SegmentPeriod.hpp](#).

Referenced by [getOffDateOffset\(\)](#), and [setOffDateOffset\(\)](#).

32.140.6.7 **Duration\_T** stdair::SegmentPeriod::\_elapsedTime [protected]

Definition at line 141 of file [SegmentPeriod.hpp](#).

Referenced by [getElapsedTime\(\)](#), and [setElapsedTime\(\)](#).

32.140.6.8 **CabinBookingClassMap\_T** stdair::SegmentPeriod::\_cabinBookingClassMap [protected]

Definition at line 142 of file [SegmentPeriod.hpp](#).

Referenced by [addCabinBookingClassList\(\)](#), and [getCabinBookingClassMap\(\)](#).

32.140.6.9 **HolderMap\_T** stdair::SegmentPeriod::\_holderMap [protected]

Definition at line 143 of file [SegmentPeriod.hpp](#).

Referenced by [getHolderMap\(\)](#).

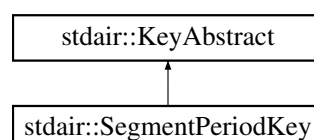
The documentation for this class was generated from the following files:

- [stdair/bom/SegmentPeriod.hpp](#)
- [stdair/bom/SegmentPeriod.cpp](#)

## 32.141 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

## 32.141.1 Detailed Description

Key of segment-period.

Definition at line 14 of file [SegmentPeriodKey.hpp](#).

## 32.141.2 Constructor &amp; Destructor Documentation

32.141.2.1 `stdair::SegmentPeriodKey::SegmentPeriodKey ( const AirportCode\_T & iBoardingPoint, const AirportCode\_T & iOffPoint )`

Constructors.

Definition at line 12 of file [SegmentPeriodKey.cpp](#).

32.141.2.2 `stdair::SegmentPeriodKey::SegmentPeriodKey ( const SegmentPeriodKey & iKey )`

Definition at line 18 of file [SegmentPeriodKey.cpp](#).

32.141.2.3 `stdair::SegmentPeriodKey::~~SegmentPeriodKey ( )`

Destructor.

Definition at line 23 of file [SegmentPeriodKey.cpp](#).

## 32.141.3 Member Function Documentation

32.141.3.1 `const AirportCode\_T& stdair::SegmentPeriodKey::getBoardingPoint ( ) const` `[inline]`

Get the boarding point.

Definition at line 29 of file [SegmentPeriodKey.hpp](#).

Referenced by [stdair::SegmentPeriod::getBoardingPoint\(\)](#).

32.141.3.2 `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\(\)](#).

32.141.3.3 `void stdair::SegmentPeriodKey::toStream ( std::ostream & ioOut ) const` `[virtual]`

Dump a Business Object Key into an output stream.

## Parameters

<i>ostream</i> &	the output stream.
------------------	--------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 27 of file [SegmentPeriodKey.cpp](#).

References [toString\(\)](#).

32.141.3.4 void stdair::SegmentPeriodKey::fromStream ( std::istream & *ioln* ) [virtual]

Read a Business Object Key from an input stream.

## Parameters

<i>istream</i> &	the input stream.
------------------	-------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 32 of file [SegmentPeriodKey.cpp](#).

32.141.3.5 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:

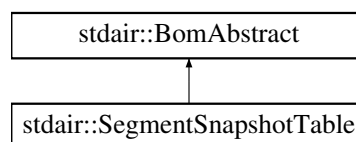
- [stdair/bom/SegmentPeriodKey.hpp](#)
- [stdair/bom/SegmentPeriodKey.cpp](#)

## 32.142 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 getConstSegmentCabinDTDSnapshotView (const SegmentDataID\_T, const SegmentDataID\_T, const DTD\_T) const
- ConstSegmentCabinDTDRangeSnapshotView\_T getConstSegmentCabinDTDRangeBookingSnapshotView (const SegmentDataID\_T, const SegmentDataID\_T, const DTD\_T, const DTD\_T) const
- SegmentCabinDTDSnapshotView\_T getSegmentCabinDTDSnapshotView (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 getConstSegmentCabinDTDRangeCancellationSnapshotView (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 getSegmentCabinDTDRangeCancellationSnapshotView (const SegmentDataID\_T, const SegmentDataID\_T, const DTD\_T, const DTD\_T)
- ConstSegmentCabinDTDSnapshotView\_T getConstSegmentCabinDTDProductOrientedNetBookingSnapshotView (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)

- [SegmentCabinDTRangeSnapshotView\\_T](#) [getSegmentCabinDTRangePriceOrientedGrossBookingSnapshotView](#) (const [SegmentDataID\\_T](#), const [SegmentDataID\\_T](#), const [DTD\\_T](#), const [DTD\\_T](#))
- [ConstSegmentCabinDTRangeSnapshotView\\_T](#) [getConstSegmentCabinDTRangeAvailabilitySnapshotView](#) (const [SegmentDataID\\_T](#), const [SegmentDataID\\_T](#), const [DTD\\_T](#)) const
- [ConstSegmentCabinDTRangeSnapshotView\\_T](#) [getConstSegmentCabinDTRangeAvailabilitySnapshotView](#) (const [SegmentDataID\\_T](#), const [SegmentDataID\\_T](#), const [DTD\\_T](#), const [DTD\\_T](#)) const
- [SegmentCabinDTRangeSnapshotView\\_T](#) [getSegmentCabinDTRangeAvailabilitySnapshotView](#) (const [SegmentDataID\\_T](#), const [SegmentDataID\\_T](#), const [DTD\\_T](#))
- [SegmentCabinDTRangeSnapshotView\\_T](#) [getSegmentCabinDTRangeAvailabilitySnapshotView](#) (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

- template<typename BOM >  
class [FacBom](#)
- class [FacBomManager](#)
- class [boost::serialization::access](#)

#### 32.142.1 Detailed Description

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

Definition at line 31 of file [SegmentSnapshotTable.hpp](#).



### 32.142.2 Member Typedef Documentation

#### 32.142.2.1 typedef SegmentSnapshotTableKey stdair::SegmentSnapshotTable::Key\_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 41 of file [SegmentSnapshotTable.hpp](#).

### 32.142.3 Constructor & Destructor Documentation

#### 32.142.3.1 stdair::SegmentSnapshotTable::SegmentSnapshotTable ( const Key\_T & iKey ) [protected]

Main constructor.

Definition at line 34 of file [SegmentSnapshotTable.cpp](#).

#### 32.142.3.2 stdair::SegmentSnapshotTable::~~SegmentSnapshotTable ( ) [protected],[virtual]

Destructor.

Definition at line 38 of file [SegmentSnapshotTable.cpp](#).

### 32.142.4 Member Function Documentation

#### 32.142.4.1 const Key\_T& stdair::SegmentSnapshotTable::getKey ( ) const [inline]

Get the segment data table key.

Definition at line 47 of file [SegmentSnapshotTable.hpp](#).

References [\\_key](#).

#### 32.142.4.2 BomAbstract\* const stdair::SegmentSnapshotTable::getParent ( ) const [inline]

Get the parent object.

Definition at line 52 of file [SegmentSnapshotTable.hpp](#).

References [\\_parent](#).

#### 32.142.4.3 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\(\)](#).

#### 32.142.4.4 const HolderMap\_T& stdair::SegmentSnapshotTable::getHolderMap ( ) const [inline]

Get the map of children holders.

Definition at line 64 of file [SegmentSnapshotTable.hpp](#).

References [\\_holderMap](#).

#### 32.142.4.5 const SegmentCabinIndexMap\_T& stdair::SegmentSnapshotTable::getSegmentCabinIndexMap ( ) const [inline]

Get the segment-cabin index map.

Definition at line 69 of file [SegmentSnapshotTable.hpp](#).

References [\\_segmentCabinIndexMap](#).

32.142.4.6 `const ClassIndexMap_T & stdair::SegmentSnapshotTable::getClassIndexMap ( ) const` `[inline]`

Get the class index map.

Definition at line 74 of file [SegmentSnapshotTable.hpp](#).

References [\\_classIndexMap](#).

32.142.4.7 `const ClassIndex_T & stdair::SegmentSnapshotTable::getClassIndex ( const MapKey_T & iKey ) const`

Get the index corresponding to the given class.

Definition at line 88 of file [SegmentSnapshotTable.cpp](#).

References [\\_classIndexMap](#).

32.142.4.8 `const SegmentDataID_T & stdair::SegmentSnapshotTable::getSegmentDataID ( const SegmentCabin & iSegmentCabin ) const`

Get the segment data ID corresponding to the givent segment-cabin.

Definition at line 97 of file [SegmentSnapshotTable.cpp](#).

References [\\_segmentCabinIndexMap](#).

32.142.4.9 `ConstSegmentCabinDTDSnapshotView_T stdair::SegmentSnapshotTable::getConstSegmentCabinDTD↔  
BookingSnapshotView ( const SegmentDataID_T iSCIdxBegin, const SegmentDataID_T iSCIdxEnd, const  
DTD_T iDTD ) 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](#).

32.142.4.10 `ConstSegmentCabinDTDRangeSnapshotView_T stdair::SegmentSnapshotTable::getConstSegment↔  
CabinDTDRangeBookingSnapshotView ( const SegmentDataID_T iSCIdxBegin, const SegmentDataID_T  
iSCIdxEnd, const DTD_T iDTDBegin, const DTD_T iDTDEnd ) const`

Get the const view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 119 of file [SegmentSnapshotTable.cpp](#).

32.142.4.11 `SegmentCabinDTDSnapshotView_T stdair::SegmentSnapshotTable::getSegmentCabinDTDBooking↔  
SnapshotView ( const SegmentDataID_T iSCIdxBegin, const SegmentDataID_T iSCIdxEnd, const DTD_T  
iDTD )`

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](#).

32.142.4.12 `SegmentCabinDTDRangeSnapshotView_T stdair::SegmentSnapshotTable::getSegmentCabinDTDRange↔  
BookingSnapshotView ( const SegmentDataID_T iSCIdxBegin, const SegmentDataID_T iSCIdxEnd, const  
DTD_T iDTDBegin, const DTD_T iDTDEnd )`

Get the view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 142 of file [SegmentSnapshotTable.cpp](#).

References [\\_bookingSnapshotBlock](#), and [\\_classIndexMap](#).

**32.142.4.13** `ConstSegmentCabinDTDSnapshotView_T` `stdair::SegmentSnapshotTable::getConstSegmentCabinDTD↔  
CancellationSnapshotView ( const SegmentDataID_T iSCldxBegin, const SegmentDataID_T iSCldxEnd,  
const DTD_T iDTD ) const`

Get the const view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 155 of file [SegmentSnapshotTable.cpp](#).

References [\\_cancellationSnapshotBlock](#), and [\\_classIndexMap](#).

**32.142.4.14** `ConstSegmentCabinDTDRangeSnapshotView_T` `stdair::SegmentSnapshotTable::getConst↔  
SegmentCabinDTDRangeCancellationSnapshotView ( const SegmentDataID_T iSCldxBegin, const  
SegmentDataID_T iSCldxEnd, const DTD_T iDTDBegin, const DTD_T iDTDEnd ) const`

Get the const view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 168 of file [SegmentSnapshotTable.cpp](#).

**32.142.4.15** `SegmentCabinDTDSnapshotView_T` `stdair::SegmentSnapshotTable::getSegmentCabinDTDCancellation↔  
SnapshotView ( const SegmentDataID_T iSCldxBegin, const SegmentDataID_T iSCldxEnd, const DTD_T  
iDTD )`

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](#).

**32.142.4.16** `SegmentCabinDTDRangeSnapshotView_T` `stdair::SegmentSnapshotTable::getSegmentCabinDTDRange↔  
CancellationSnapshotView ( const SegmentDataID_T iSCldxBegin, const SegmentDataID_T iSCldxEnd,  
const DTD_T iDTDBegin, const DTD_T iDTDEnd )`

Get the view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 191 of file [SegmentSnapshotTable.cpp](#).

References [\\_cancellationSnapshotBlock](#), and [\\_classIndexMap](#).

**32.142.4.17** `ConstSegmentCabinDTDSnapshotView_T` `stdair::SegmentSnapshotTable::getConstSegment↔  
CabinDTDProductOrientedNetBookingSnapshotView ( const SegmentDataID_T iSCldxBegin, const  
SegmentDataID_T iSCldxEnd, const DTD_T iDTD ) const`

Get the const view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 204 of file [SegmentSnapshotTable.cpp](#).

References [\\_classIndexMap](#), and [\\_productOrientedNetBookingSnapshotBlock](#).

**32.142.4.18** `ConstSegmentCabinDTDRangeSnapshotView_T` `stdair::SegmentSnapshotTable::getConstSegment↔  
CabinDTDRangeProductOrientedNetBookingSnapshotView ( const SegmentDataID_T iSCldxBegin, const  
SegmentDataID_T iSCldxEnd, const DTD_T iDTDBegin, const DTD_T iDTDEnd ) const`

Get the const view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 217 of file [SegmentSnapshotTable.cpp](#).

**32.142.4.19** `SegmentCabinDTDSnapshotView_T` `stdair::SegmentSnapshotTable::getSegmentCabinDTDProduct↔  
OrientedNetBookingSnapshotView ( const SegmentDataID_T iSCldxBegin, const SegmentDataID_T  
iSCldxEnd, const DTD_T iDTD )`

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](#).

**32.142.4.20** `SegmentCabinDTDRangeSnapshotView_T` `stdair::SegmentSnapshotTable::getSegmentCabinDTDRangeProductOrientedNetBookingSnapshotView ( const SegmentDataID_T iSCldxBegin, const SegmentDataID_T iSCldxEnd, const DTD_T iDTDBegin, const DTD_T iDTDEnd )`

Get the view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 240 of file [SegmentSnapshotTable.cpp](#).

References [\\_classIndexMap](#), and [\\_productOrientedNetBookingSnapshotBlock](#).

**32.142.4.21** `ConstSegmentCabinDTDSnapshotView_T` `stdair::SegmentSnapshotTable::getConstSegmentCabinDTDPPriceOrientedNetBookingSnapshotView ( const SegmentDataID_T iSCldxBegin, const SegmentDataID_T iSCldxEnd, const DTD_T iDTD ) const`

Get the const view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 254 of file [SegmentSnapshotTable.cpp](#).

References [\\_classIndexMap](#), and [\\_priceOrientedNetBookingSnapshotBlock](#).

**32.142.4.22** `ConstSegmentCabinDTDRangeSnapshotView_T` `stdair::SegmentSnapshotTable::getConstSegmentCabinDTDRangePriceOrientedNetBookingSnapshotView ( const SegmentDataID_T iSCldxBegin, const SegmentDataID_T iSCldxEnd, const DTD_T iDTDBegin, const DTD_T iDTDEnd ) const`

Get the const view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 267 of file [SegmentSnapshotTable.cpp](#).

**32.142.4.23** `SegmentCabinDTDSnapshotView_T` `stdair::SegmentSnapshotTable::getSegmentCabinDTDPPriceOrientedNetBookingSnapshotView ( const SegmentDataID_T iSCldxBegin, const SegmentDataID_T iSCldxEnd, const DTD_T iDTD )`

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](#).

**32.142.4.24** `SegmentCabinDTDRangeSnapshotView_T` `stdair::SegmentSnapshotTable::getSegmentCabinDTDRangePriceOrientedNetBookingSnapshotView ( const SegmentDataID_T iSCldxBegin, const SegmentDataID_T iSCldxEnd, const DTD_T iDTDBegin, const DTD_T iDTDEnd )`

Get the view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 290 of file [SegmentSnapshotTable.cpp](#).

References [\\_classIndexMap](#), and [\\_priceOrientedNetBookingSnapshotBlock](#).

**32.142.4.25** `ConstSegmentCabinDTDSnapshotView_T` `stdair::SegmentSnapshotTable::getConstSegmentCabinDTDPProductOrientedGrossBookingSnapshotView ( const SegmentDataID_T iSCldxBegin, const SegmentDataID_T iSCldxEnd, const DTD_T iDTD ) const`

Get the const view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 303 of file [SegmentSnapshotTable.cpp](#).

References [\\_classIndexMap](#), and [\\_productOrientedGrossBookingSnapshotBlock](#).

**32.142.4.26** `ConstSegmentCabinDTDRangeSnapshotView_T` `stdair::SegmentSnapshotTable::getConstSegmentCabinDTDRangeProductOrientedGrossBookingSnapshotView ( const SegmentDataID_T iSCldxBegin, const SegmentDataID_T iSCldxEnd, const DTD_T iDTDBegin, const DTD_T iDTDEnd ) const`

Get the const view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 316 of file [SegmentSnapshotTable.cpp](#).

**32.142.4.27** `SegmentCabinDTDSnapshotView_T` stdair::SegmentSnapshotTable::getSegmentCabinDTDPProductOrientedGrossBookingSnapshotView ( const SegmentDataID\_T iSCldxBegin, const SegmentDataID\_T iSCldxEnd, const DTD\_T iDTD )

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](#).

**32.142.4.28** `SegmentCabinDTDRangeSnapshotView_T` stdair::SegmentSnapshotTable::getSegmentCabinDTDRangeProductOrientedGrossBookingSnapshotView ( const SegmentDataID\_T iSCldxBegin, const SegmentDataID\_T iSCldxEnd, const DTD\_T iDTDBegin, const DTD\_T iDTDEnd )

Get the view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 339 of file [SegmentSnapshotTable.cpp](#).

References [\\_classIndexMap](#), and [\\_productOrientedGrossBookingSnapshotBlock](#).

**32.142.4.29** `ConstSegmentCabinDTDSnapshotView_T` stdair::SegmentSnapshotTable::getConstSegmentCabinDTDPPriceOrientedGrossBookingSnapshotView ( const SegmentDataID\_T iSCldxBegin, const SegmentDataID\_T iSCldxEnd, const DTD\_T iDTD ) const

Get the const view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 353 of file [SegmentSnapshotTable.cpp](#).

References [\\_classIndexMap](#), and [\\_priceOrientedGrossBookingSnapshotBlock](#).

**32.142.4.30** `ConstSegmentCabinDTDRangeSnapshotView_T` stdair::SegmentSnapshotTable::getConstSegmentCabinDTDRangePriceOrientedGrossBookingSnapshotView ( const SegmentDataID\_T iSCldxBegin, const SegmentDataID\_T iSCldxEnd, const DTD\_T iDTDBegin, const DTD\_T iDTDEnd ) const

Get the const view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 366 of file [SegmentSnapshotTable.cpp](#).

**32.142.4.31** `SegmentCabinDTDSnapshotView_T` stdair::SegmentSnapshotTable::getSegmentCabinDTDPPriceOrientedGrossBookingSnapshotView ( const SegmentDataID\_T iSCldxBegin, const SegmentDataID\_T iSCldxEnd, const DTD\_T iDTD )

Get the view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 378 of file [SegmentSnapshotTable.cpp](#).

**32.142.4.32** `SegmentCabinDTDRangeSnapshotView_T` stdair::SegmentSnapshotTable::getSegmentCabinDTDRangePriceOrientedGrossBookingSnapshotView ( const SegmentDataID\_T iSCldxBegin, const SegmentDataID\_T iSCldxEnd, const DTD\_T iDTDBegin, const DTD\_T iDTDEnd )

Get the view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 390 of file [SegmentSnapshotTable.cpp](#).

**32.142.4.33** `ConstSegmentCabinDTDSnapshotView_T` stdair::SegmentSnapshotTable::getConstSegmentCabinDTDAvailabilitySnapshotView ( const SegmentDataID\_T iSCldxBegin, const SegmentDataID\_T iSCldxEnd, const DTD\_T iDTD ) const

Get the const view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 402 of file [SegmentSnapshotTable.cpp](#).

**32.142.4.34** `ConstSegmentCabinDTDRangeSnapshotView_T stdair::SegmentSnapshotTable::getConstSegmentCabinDTDRangeAvailabilitySnapshotView ( const SegmentDataID_T iSCldxBegin, const SegmentDataID_T iSCldxEnd, const DTD_T iDTDBegin, const DTD_T iDTDEnd ) const`

Get the const view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 414 of file [SegmentSnapshotTable.cpp](#).

**32.142.4.35** `SegmentCabinDTDSnapshotView_T stdair::SegmentSnapshotTable::getSegmentCabinDTDAvailabilitySnapshotView ( const SegmentDataID_T iSCldxBegin, const SegmentDataID_T iSCldxEnd, const DTD_T iDTD )`

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](#).

**32.142.4.36** `SegmentCabinDTDRangeSnapshotView_T stdair::SegmentSnapshotTable::getSegmentCabinDTDRangeAvailabilitySnapshotView ( const SegmentDataID_T iSCldxBegin, const SegmentDataID_T iSCldxEnd, const DTD_T iDTDBegin, const DTD_T iDTDEnd )`

Get the view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 437 of file [SegmentSnapshotTable.cpp](#).

References [\\_availabilitySnapshotBlock](#), and [\\_classIndexMap](#).

**32.142.4.37** `void stdair::SegmentSnapshotTable::initSnapshotBlocks ( const SegmentCabinIndexMap_T & iSegmentCabinIndexMap, const ClassIndexMap_T & iClassIndexMap )`

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](#).

**32.142.4.38** `void stdair::SegmentSnapshotTable::toStream ( std::ostream & ioOut ) const [inline], [virtual]`

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Implements [stdair::BomAbstract](#).

Definition at line 283 of file [SegmentSnapshotTable.hpp](#).

References [toString\(\)](#).

**32.142.4.39** `void stdair::SegmentSnapshotTable::fromStream ( std::istream & ioIn ) [inline], [virtual]`

Read a Business Object from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Implements [stdair::BomAbstract](#).

Definition at line 292 of file [SegmentSnapshotTable.hpp](#).

**32.142.4.40** `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\(\)](#).

**32.142.4.41** `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\(\)](#).

**32.142.4.42** `template<class Archive > void stdair::SegmentSnapshotTable::serialize ( Archive & ar, const unsigned int iFileVersion )`

Serialisation.

Definition at line 464 of file [SegmentSnapshotTable.cpp](#).

References [\\_key](#).

## 32.142.5 Friends And Related Function Documentation

**32.142.5.1** `template<typename BOM > friend class FacBom [friend]`

Definition at line 32 of file [SegmentSnapshotTable.hpp](#).

**32.142.5.2** `friend class FacBomManager [friend]`

Definition at line 33 of file [SegmentSnapshotTable.hpp](#).

**32.142.5.3** `friend class boost::serialization::access [friend]`

Definition at line 34 of file [SegmentSnapshotTable.hpp](#).

## 32.142.6 Member Data Documentation

**32.142.6.1** `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\(\)](#).

**32.142.6.2** `BomAbstract* stdair::SegmentSnapshotTable::_parent [protected]`

Pointer on the parent class ([Inventory](#)).

Definition at line 355 of file [SegmentSnapshotTable.hpp](#).

Referenced by [getParent\(\)](#).

**32.142.6.3** `HolderMap_T stdair::SegmentSnapshotTable::_holderMap [protected]`

Map holding the children.



Definition at line 358 of file [SegmentSnapshotTable.hpp](#).

Referenced by [getHolderMap\(\)](#).

#### 32.142.6.4 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\(\)](#).

#### 32.142.6.5 ClassIndexMap\_T stdair::SegmentSnapshotTable::\_classIndexMap [protected]

Map holding the value type (class, etc) within a segment-cabin inside the snapshot blocks.

Definition at line 365 of file [SegmentSnapshotTable.hpp](#).

Referenced by [getClassIndex\(\)](#), [getClassIndexMap\(\)](#), [getConstSegmentCabinDTDBBookingSnapshotView\(\)](#), [getConstSegmentCabinDTDCancellationSnapshotView\(\)](#), [getConstSegmentCabinDTDPriceOrientedGrossBookingSnapshotView\(\)](#), [getConstSegmentCabinDTDPriceOrientedNetBookingSnapshotView\(\)](#), [getConstSegmentCabinDTDProductOrientedGrossBookingSnapshotView\(\)](#), [getConstSegmentCabinDTDProductOrientedNetBookingSnapshotView\(\)](#), [getSegmentCabinDTDAvailabilitySnapshotView\(\)](#), [getSegmentCabinDTDBBookingSnapshotView\(\)](#), [getSegmentCabinDTDCancellationSnapshotView\(\)](#), [getSegmentCabinDTDPriceOrientedNetBookingSnapshotView\(\)](#), [getSegmentCabinDTDProductOrientedGrossBookingSnapshotView\(\)](#), [getSegmentCabinDTDProductOrientedNetBookingSnapshotView\(\)](#), [getSegmentCabinDTDRangeAvailabilitySnapshotView\(\)](#), [getSegmentCabinDTDRangeBookingSnapshotView\(\)](#), [getSegmentCabinDTDRangeCancellationSnapshotView\(\)](#), [getSegmentCabinDTDRangePriceOrientedNetBookingSnapshotView\(\)](#), [getSegmentCabinDTDRangeProductOrientedGrossBookingSnapshotView\(\)](#), [getSegmentCabinDTDRangeProductOrientedNetBookingSnapshotView\(\)](#), and [initSnapshotBlocks\(\)](#).

#### 32.142.6.6 SnapshotBlock\_T stdair::SegmentSnapshotTable::\_bookingSnapshotBlock [protected]

Booking snapshot block.

Definition at line 368 of file [SegmentSnapshotTable.hpp](#).

Referenced by [getConstSegmentCabinDTDBBookingSnapshotView\(\)](#), [getSegmentCabinDTDBBookingSnapshotView\(\)](#), [getSegmentCabinDTDRangeBookingSnapshotView\(\)](#), and [initSnapshotBlocks\(\)](#).

#### 32.142.6.7 SnapshotBlock\_T stdair::SegmentSnapshotTable::\_cancellationSnapshotBlock [protected]

Cancellation snapshot block.

Definition at line 371 of file [SegmentSnapshotTable.hpp](#).

Referenced by [getConstSegmentCabinDTDCancellationSnapshotView\(\)](#), [getSegmentCabinDTDCancellationSnapshotView\(\)](#), [getSegmentCabinDTDRangeCancellationSnapshotView\(\)](#), and [initSnapshotBlocks\(\)](#).

#### 32.142.6.8 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\(\)](#).

#### 32.142.6.9 SnapshotBlock\_T stdair::SegmentSnapshotTable::\_priceOrientedNetBookingSnapshotBlock [protected]

Price oriented net booking block.

Definition at line 377 of file [SegmentSnapshotTable.hpp](#).



Referenced by [getConstSegmentCabinDTDPriceOrientedNetBookingSnapshotView\(\)](#), [getSegmentCabinDTDPriceOrientedNetBookingSnapshotView\(\)](#), [getSegmentCabinDTRangePriceOrientedNetBookingSnapshotView\(\)](#), and [initSnapshotBlocks\(\)](#).

#### 32.142.6.10 **SnapshotBlock\_T** stdair::SegmentSnapshotTable::\_productOrientedGrossBookingSnapshotBlock [protected]

Product oriented gross booking block.

Definition at line 380 of file [SegmentSnapshotTable.hpp](#).

Referenced by [getConstSegmentCabinDTDProductOrientedGrossBookingSnapshotView\(\)](#), [getSegmentCabinDTDProductOrientedGrossBookingSnapshotView\(\)](#), [getSegmentCabinDTRangeProductOrientedGrossBookingSnapshotView\(\)](#), and [initSnapshotBlocks\(\)](#).

#### 32.142.6.11 **SnapshotBlock\_T** stdair::SegmentSnapshotTable::\_priceOrientedGrossBookingSnapshotBlock [protected]

Price oriented gross booking block.

Definition at line 383 of file [SegmentSnapshotTable.hpp](#).

Referenced by [getConstSegmentCabinDTDPriceOrientedGrossBookingSnapshotView\(\)](#), and [initSnapshotBlocks\(\)](#).

#### 32.142.6.12 **SnapshotBlock\_T** stdair::SegmentSnapshotTable::\_availabilitySnapshotBlock [protected]

Availability block.

Definition at line 386 of file [SegmentSnapshotTable.hpp](#).

Referenced by [getSegmentCabinDTDAvailabilitySnapshotView\(\)](#), [getSegmentCabinDTRangeAvailabilitySnapshotView\(\)](#), and [initSnapshotBlocks\(\)](#).

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

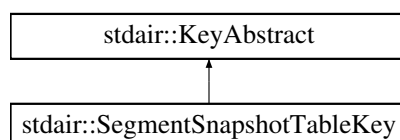
- [stdair/bom/SegmentSnapshotTable.hpp](#)
- [stdair/bom/SegmentSnapshotTable.cpp](#)

### 32.143 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](#)

#### 32.143.1 Detailed Description

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

Definition at line 26 of file [SegmentSnapshotTableKey.hpp](#).

#### 32.143.2 Constructor & Destructor Documentation

##### 32.143.2.1 stdair::SegmentSnapshotTableKey::SegmentSnapshotTableKey ( const TableID\_T & iTableID )

Constructor.

Definition at line 26 of file [SegmentSnapshotTableKey.cpp](#).

##### 32.143.2.2 stdair::SegmentSnapshotTableKey::SegmentSnapshotTableKey ( const SegmentSnapshotTableKey & iKey )

Copy constructor.

Definition at line 31 of file [SegmentSnapshotTableKey.cpp](#).

##### 32.143.2.3 stdair::SegmentSnapshotTableKey::~~SegmentSnapshotTableKey ( )

Destructor.

Definition at line 36 of file [SegmentSnapshotTableKey.cpp](#).

#### 32.143.3 Member Function Documentation

##### 32.143.3.1 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\(\)](#).

##### 32.143.3.2 void stdair::SegmentSnapshotTableKey::toStream ( std::ostream & ioOut ) const [virtual]

Dump a Business Object Key into an output stream.

#### Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 40 of file [SegmentSnapshotTableKey.cpp](#).

References [toString\(\)](#).

##### 32.143.3.3 void stdair::SegmentSnapshotTableKey::fromStream ( std::istream & ioIn ) [virtual]

Read a Business Object Key from an input stream.

## Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 45 of file [SegmentSnapshotTableKey.cpp](#).

32.143.3.4 `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\(\)](#).

32.143.3.5 `template<class Archive > void stdair::SegmentSnapshotTableKey::serialize ( Archive & ar, const unsigned int iFileVersion )`

Serialisation.

Definition at line 71 of file [SegmentSnapshotTableKey.cpp](#).

#### 32.143.4 Friends And Related Function Documentation

32.143.4.1 `friend class boost::serialization::access` `[friend]`

Definition at line 27 of file [SegmentSnapshotTableKey.hpp](#).

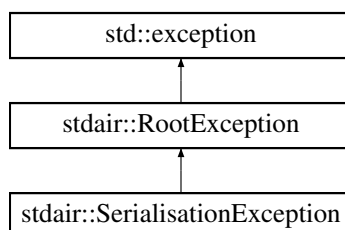
The documentation for this struct was generated from the following files:

- [stdair/bom/SegmentSnapshotTableKey.hpp](#)
- [stdair/bom/SegmentSnapshotTableKey.cpp](#)

### 32.144 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`

## 32.144.1 Detailed Description

Serialisation.

Definition at line 119 of file [stdair\\_exceptions.hpp](#).

## 32.144.2 Constructor &amp; Destructor Documentation

32.144.2.1 `stdair::SerialisationException::SerialisationException ( const std::string & iWhat )` `[inline]`

Constructor.

Definition at line 122 of file [stdair\\_exceptions.hpp](#).

## 32.144.3 Member Function Documentation

32.144.3.1 `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\(\)](#).

## 32.144.4 Member Data Documentation

32.144.4.1 `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\(\)](#).

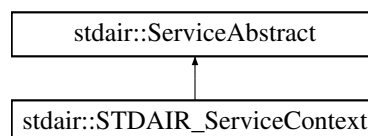
The documentation for this class was generated from the following file:

- [stdair/stdair\\_exceptions.hpp](#)

## 32.145 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](#) ()

### 32.145.1 Detailed Description

Base class for the Service layer.

Definition at line 15 of file [ServiceAbstract.hpp](#).

### 32.145.2 Constructor & Destructor Documentation

**32.145.2.1** virtual stdair::ServiceAbstract::~~ServiceAbstract ( ) [inline],[virtual]

Destructor.

Definition at line 21 of file [ServiceAbstract.hpp](#).

**32.145.2.2** stdair::ServiceAbstract::ServiceAbstract ( ) [inline],[protected]

Display of the structure. Protected Default Constructor to ensure this class is abstract.

Definition at line 46 of file [ServiceAbstract.hpp](#).

### 32.145.3 Member Function Documentation

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

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Definition at line 28 of file [ServiceAbstract.hpp](#).

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

Read a Business Object from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Definition at line 35 of file [ServiceAbstract.hpp](#).

Referenced by [operator>>\(\)](#).

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

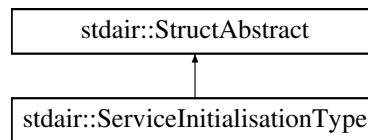
- stdair/service/[ServiceAbstract.hpp](#)

## 32.146 stdair::ServiceInitialisationType Struct Reference

Enumeration of service initialisation types.

```
#include <stdair/basic/ServiceInitialisationType.hpp>
```

Inheritance diagram for stdair::ServiceInitialisationType:



### Public Types

- enum [EN\\_ServiceInitialisationType](#) { [NOT\\_YET\\_INITIALISED](#) = 0, [FILE\\_PARSING](#), [BUILTIN\\_SAMPLE](#), [L<AST\\_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](#) ()

#### 32.146.1 Detailed Description

Enumeration of service initialisation types.

Definition at line 17 of file [ServiceInitialisationType.hpp](#).

#### 32.146.2 Member Enumeration Documentation

##### 32.146.2.1 enum stdair::ServiceInitialisationType::EN\_ServiceInitialisationType

Enumerator

***NOT\_YET\_INITIALISED***  
***FILE\_PARSING***

**BUILTIN\_SAMPLE****LAST\_VALUE**

Definition at line 19 of file [ServiceInitialisationType.hpp](#).

**32.146.3 Constructor & Destructor Documentation**

**32.146.3.1** `stdair::ServiceInitialisationType::ServiceInitialisationType ( const EN_ServiceInitialisationType & iServiceInitialisationType )`

Main constructor.

Definition at line 36 of file [ServiceInitialisationType.cpp](#).

**32.146.3.2** `stdair::ServiceInitialisationType::ServiceInitialisationType ( const char iType )`

Alternative constructor.

Definition at line 65 of file [ServiceInitialisationType.cpp](#).

**32.146.3.3** `stdair::ServiceInitialisationType::ServiceInitialisationType ( const std::string & iType )`

Alternative constructor.

Definition at line 71 of file [ServiceInitialisationType.cpp](#).

References [getType\(\)](#).

**32.146.3.4** `stdair::ServiceInitialisationType::ServiceInitialisationType ( const ServiceInitialisationType & iServiceInitialisationType )`

Default copy constructor.

Definition at line 30 of file [ServiceInitialisationType.cpp](#).

**32.146.4 Member Function Documentation**

**32.146.4.1** `const std::string & stdair::ServiceInitialisationType::getLabel ( const EN_ServiceInitialisationType & iType ) [static]`

Get the label as a string (e.g., "Not yet initialised", "File parsing" or "Built-in sample BOM").

Definition at line 81 of file [ServiceInitialisationType.cpp](#).

**32.146.4.2** `ServiceInitialisationType::EN_ServiceInitialisationType stdair::ServiceInitialisationType::getType ( const char iTypeChar ) [static]`

Get the type value from parsing a single char (e.g., 'N', 'F', 'B').

Definition at line 42 of file [ServiceInitialisationType.cpp](#).

References [BUILTIN\\_SAMPLE](#), [describeLabels\(\)](#), [FILE\\_PARSING](#), [LAST\\_VALUE](#), and [NOT\\_YET\\_INITIALISED](#).

**32.146.4.3** `char stdair::ServiceInitialisationType::getTypeLabel ( const EN_ServiceInitialisationType & iType ) [static]`

Get the label as a single char (e.g., 'N', 'F', 'B').

Definition at line 87 of file [ServiceInitialisationType.cpp](#).

32.146.4.4 `std::string stdair::ServiceInitialisationType::getTypeLabelAsString ( const EN_ServiceInitialisationType & iType ) [static]`

Get the label as a string of a single char (e.g., "N", "F", "B").

Definition at line 93 of file [ServiceInitialisationType.cpp](#).

32.146.4.5 `std::string stdair::ServiceInitialisationType::describeLabels ( ) [static]`

List the labels.

Definition at line 100 of file [ServiceInitialisationType.cpp](#).

References [LAST\\_VALUE](#).

Referenced by [getType\(\)](#).

32.146.4.6 `ServiceInitialisationType::EN_ServiceInitialisationType stdair::ServiceInitialisationType::getType ( ) const`

Get the enumerated value.

Definition at line 113 of file [ServiceInitialisationType.cpp](#).

Referenced by [ServiceInitialisationType\(\)](#).

32.146.4.7 `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](#).

32.146.4.8 `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](#).

32.146.4.9 `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](#).

32.146.4.10 `bool stdair::ServiceInitialisationType::operator== ( const EN_ServiceInitialisationType & iType ) const`

Comparison operator.

Definition at line 139 of file [ServiceInitialisationType.cpp](#).

32.146.4.11 `void stdair::StructAbstract::toStream ( std::ostream & ioOut ) const [inline],[inherited]`

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Definition at line 29 of file [StructAbstract.hpp](#).

References [stdair::StructAbstract::describe\(\)](#).

32.146.4.12 `virtual void stdair::StructAbstract::fromStream ( std::istream & ioIn ) [inline],[virtual],[inherited]`

Read a Business Object from an input stream.



## Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented in [stdair::BookingRequestStruct](#), [stdair::EventStruct](#), [stdair::TravelSolutionStruct](#), [stdair::VirtualClassStruct](#), [stdair::ConfigHolderStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::ProgressStatusSet](#), [stdair::YieldRange](#), [stdair::CancellationStruct](#), [stdair::AirlineStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::FFDisutilityCurveHolderStruct](#), [stdair::FRAT5CurveHolderStruct](#), and [stdair::BreakPointStruct](#).

Definition at line 38 of file [StructAbstract.hpp](#).

Referenced by [operator>>\(\)](#).

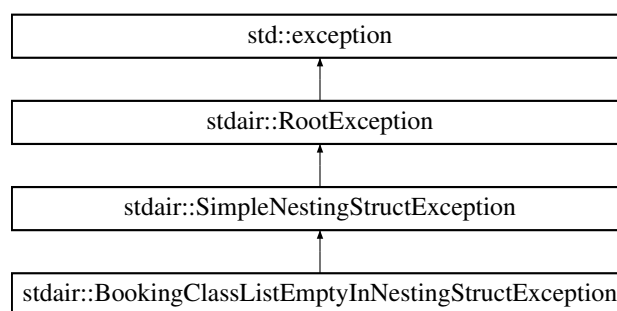
The documentation for this struct was generated from the following files:

- [stdair/basic/ServiceInitialisationType.hpp](#)
- [stdair/basic/ServiceInitialisationType.cpp](#)

### 32.147 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](#)

#### 32.147.1 Detailed Description

Simple Nesting Structure.

Definition at line 211 of file [stdair\\_exceptions.hpp](#).

#### 32.147.2 Constructor & Destructor Documentation

32.147.2.1 [stdair::SimpleNestingStructException::SimpleNestingStructException \( const std::string & iWhat \)](#) [inline]

Constructor.

Definition at line 214 of file [stdair\\_exceptions.hpp](#).

## 32.147.3 Member Function Documentation

32.147.3.1 `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\(\)](#).

## 32.147.4 Member Data Documentation

32.147.4.1 `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\(\)](#).

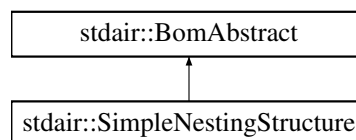
The documentation for this class was generated from the following file:

- [stdair/stdair\\_exceptions.hpp](#)

## 32.148 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

- `template<typename BOM >`  
  class [FacBom](#)
- class [FacBomManager](#)
- class [boost::serialization::access](#)

### 32.148.1 Detailed Description

Structure holding a nesting node map according to the yield.

Definition at line 26 of file [SimpleNestingStructure.hpp](#).

### 32.148.2 Member Typedef Documentation

#### 32.148.2.1 `typedef NestingStructureKey stdair::SimpleNestingStructure::Key_T`

Definition allowing to retrieve the associated BOM key type.

Definition at line 36 of file [SimpleNestingStructure.hpp](#).

### 32.148.3 Constructor & Destructor Documentation

#### 32.148.3.1 `stdair::SimpleNestingStructure::SimpleNestingStructure ( const Key_T & iKey )`

Main constructor.

Definition at line 36 of file [SimpleNestingStructure.cpp](#).

#### 32.148.3.2 `stdair::SimpleNestingStructure::~~SimpleNestingStructure ( ) [virtual]`

Destructor.

Definition at line 41 of file [SimpleNestingStructure.cpp](#).

### 32.148.4 Member Function Documentation

#### 32.148.4.1 `const Key_T& stdair::SimpleNestingStructure::getKey ( ) const [inline]`

Get the nesting key.

Definition at line 41 of file [SimpleNestingStructure.hpp](#).

#### 32.148.4.2 `BomAbstract* const stdair::SimpleNestingStructure::getParent ( ) const [inline]`

Get the parent object.

Definition at line 46 of file [SimpleNestingStructure.hpp](#).

#### 32.148.4.3 `const HolderMap_T& stdair::SimpleNestingStructure::getHolderMap ( ) const [inline]`

Get the map of children holders.

Definition at line 53 of file [SimpleNestingStructure.hpp](#).

#### 32.148.4.4 `const NestingNodeList_T & stdair::SimpleNestingStructure::getNestingNodeList ( ) const`

Get the nesting node list

Definition at line 115 of file [SimpleNestingStructure.cpp](#).

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

Dump a Business Object into an output stream.

## Parameters

<i>ostream</i> &	the output stream.
------------------	--------------------

Implements [stdair::BomAbstract](#).

Definition at line 69 of file [SimpleNestingStructure.hpp](#).

References [toString\(\)](#).

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

Read a Business Object from an input stream.

## Parameters

<i>istream</i> &	the input stream.
------------------	-------------------

Implements [stdair::BomAbstract](#).

Definition at line 78 of file [SimpleNestingStructure.hpp](#).

32.148.4.7 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\(\)](#).

32.148.4.8 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\(\)](#).

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

Serialisation.

## 32.148.5 Friends And Related Function Documentation

32.148.5.1 template<typename BOM > friend class FacBom [friend]

Definition at line 27 of file [SimpleNestingStructure.hpp](#).

32.148.5.2 friend class FacBomManager [friend]

Definition at line 28 of file [SimpleNestingStructure.hpp](#).

32.148.5.3 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](#)

## 32.149 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](#)

### 32.149.1 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](#).

### 32.149.2 Constructor & Destructor Documentation

#### 32.149.2.1 swift::SKeymap::SKeymap ( bool *PrintableBound* = false ) [inline], [explicit]

Creates a new keymap.

##### Parameters

<i>PrintableBound</i>	if true - the printable characters are bound if false - the keymap is empty
-----------------------	---

Definition at line 319 of file [SReadline.hpp](#).

#### 32.149.2.2 swift::SKeymap::SKeymap ( Keymap *Pattern* ) [inline], [explicit]

Creates a new keymap which is a copy of Pattern.

##### Parameters

<i>Pattern</i>	A keymap to be copied.
----------------	------------------------

Definition at line 342 of file [SReadline.hpp](#).

32.149.2.3 `swift::SKeymap::~~SKeymap ( ) [inline]`

Frees the allocated keymap.

Definition at line 354 of file [SReadline.hpp](#).

32.149.2.4 `swift::SKeymap::SKeymap ( const SKeymap & rhs ) [inline]`

Copy constructor.

Parameters

<i>rhs</i>	Right hand side object of <a href="#">SKeymap</a>
------------	---

Definition at line 395 of file [SReadline.hpp](#).

### 32.149.3 Member Function Documentation

32.149.3.1 `void swift::SKeymap::Bind ( int Key, KeyCallback Callback ) [inline]`

Binds the given key to a function.

Parameters

<i>Key</i>	A key to be bound
<i>Callback</i>	A function to be called when the Key is pressed

Definition at line 366 of file [SReadline.hpp](#).

32.149.3.2 `void swift::SKeymap::Unbind ( int Key ) [inline]`

Unbinds the given key.

Parameters

<i>Key</i>	A key to be unbound
------------	---------------------

Definition at line 381 of file [SReadline.hpp](#).

32.149.3.3 `SKeymap& swift::SKeymap::operator= ( const SKeymap & rhs ) [inline]`

operator=

Parameters

<i>rhs</i>	Right hand side object of <a href="#">SKeymap</a>
------------	---

Definition at line 407 of file [SReadline.hpp](#).

### 32.149.4 Friends And Related Function Documentation

32.149.4.1 `friend class SReadline [friend]`

Definition at line 415 of file [SReadline.hpp](#).

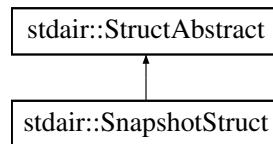
The documentation for this class was generated from the following file:

- [stdair/ui/cmdline/SReadline.hpp](#)

## 32.150 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 &ioln)
- const std::string [describe](#) () const
- [SnapshotStruct](#) (const [AirlineCode\\_T](#) &, const [DateTime\\_T](#) &)
- [SnapshotStruct](#) (const [SnapshotStruct](#) &)
- [~SnapshotStruct](#) ()

### 32.150.1 Detailed Description

Structure holding the elements of a snapshot .

Definition at line 19 of file [SnapshotStruct.hpp](#).

### 32.150.2 Constructor & Destructor Documentation

#### 32.150.2.1 stdair::SnapshotStruct::SnapshotStruct ( const [AirlineCode\\_T](#) & *iAirlineCode*, const [DateTime\\_T](#) & *iSnapshotTime* )

Constructor.

Definition at line 26 of file [SnapshotStruct.cpp](#).

#### 32.150.2.2 stdair::SnapshotStruct::SnapshotStruct ( const [SnapshotStruct](#) & *iSnapshot* )

Copy constructor.

Definition at line 19 of file [SnapshotStruct.cpp](#).

#### 32.150.2.3 stdair::SnapshotStruct::~~SnapshotStruct ( )

Destructor.

Definition at line 32 of file [SnapshotStruct.cpp](#).

### 32.150.3 Member Function Documentation

#### 32.150.3.1 const [AirlineCode\\_T](#) & stdair::SnapshotStruct::getAirlineCode ( ) const [inline]

Get the airline code.

Definition at line 23 of file [SnapshotStruct.hpp](#).



**32.150.3.2** `const DateTime_T& stdair::SnapshotStruct::getSnapshotTime ( ) const` `[inline]`

Get the snapshot action time.

Definition at line 28 of file [SnapshotStruct.hpp](#).

**32.150.3.3** `void stdair::SnapshotStruct::toStream ( std::ostream & ioOut ) const`

Dump a Business Object into an output stream.

Parameters

<code>ostream&amp;</code>	the output stream.
---------------------------	--------------------

Definition at line 36 of file [SnapshotStruct.cpp](#).

References [describe\(\)](#).

**32.150.3.4** `void stdair::SnapshotStruct::fromStream ( std::istream & ioIn )` `[virtual]`

Read a Business Object from an input stream.

Parameters

<code>istream&amp;</code>	the input stream.
---------------------------	-------------------

Reimplemented from [stdair::StructAbstract](#).

Definition at line 41 of file [SnapshotStruct.cpp](#).

**32.150.3.5** `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\(\)](#).

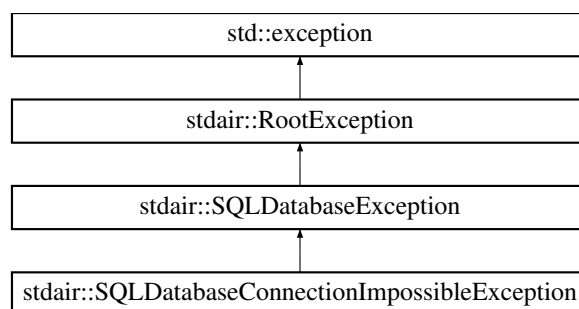
The documentation for this struct was generated from the following files:

- [stdair/bom/SnapshotStruct.hpp](#)
- [stdair/bom/SnapshotStruct.cpp](#)

## 32.151 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](#)

## 32.151.1 Detailed Description

Database connection.

Definition at line 196 of file [stdair\\_exceptions.hpp](#).

## 32.151.2 Constructor &amp; Destructor Documentation

32.151.2.1 `stdair::SQLDatabaseConnectionImpossibleException::SQLDatabaseConnectionImpossibleException ( const std::string & iWhat ) [inline]`

Constructor.

Definition at line 199 of file [stdair\\_exceptions.hpp](#).

## 32.151.3 Member Function Documentation

32.151.3.1 `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\(\)](#).

## 32.151.4 Member Data Documentation

32.151.4.1 `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\(\)](#).

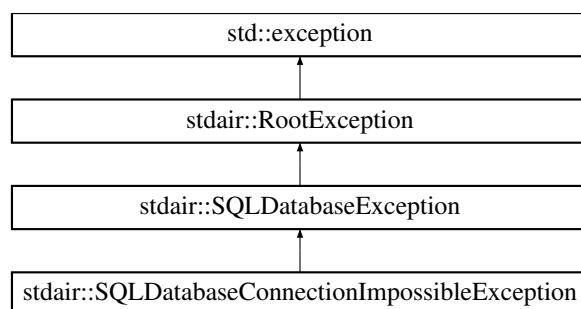
The documentation for this class was generated from the following file:

- [stdair/stdair\\_exceptions.hpp](#)

## 32.152 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](#)

#### 32.152.1 Detailed Description

Database.

Definition at line 181 of file [stdair\\_exceptions.hpp](#).

#### 32.152.2 Constructor & Destructor Documentation

32.152.2.1 `stdair::SQLDatabaseException::SQLDatabaseException ( const std::string & iWhat ) [inline]`

Constructor.

Definition at line 184 of file [stdair\\_exceptions.hpp](#).

#### 32.152.3 Member Function Documentation

32.152.3.1 `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\(\)](#).

#### 32.152.4 Member Data Documentation

32.152.4.1 `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](#)

## 32.153 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.*

### 32.153.1 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](#).

### 32.153.2 Constructor & Destructor Documentation

32.153.2.1 `swift::SReadline::SReadline ( const size_t Limit = DefaultHistoryLimit ) [inline]`

Constructs the object, sets the completion function.

## Parameters

<i>Limit</i>	History size
--------------	--------------

Definition at line 431 of file [SReadline.hpp](#).

**32.153.2.2** `swift::SReadline::SReadline ( const std::string & historyFileName, const size_t Limit = DefaultHistoryLimit ) [inline]`

Constructs the object, sets the completion function, loads history.

## Parameters

<i>historyFileName</i>	File name to load history from
<i>Limit</i>	History size

Definition at line 446 of file [SReadline.hpp](#).

References [LoadHistory\(\)](#).

**32.153.2.3** `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\(\)](#).

**32.153.3 Member Function Documentation**

**32.153.3.1** `std::string swift::SReadline::GetLine ( const std::string & Prompt ) [inline]`

Gets a single line from a user.

## Parameters

<i>Prompt</i>	A printed prompt
---------------	------------------

## Returns

A string which was actually inputed

Definition at line 473 of file [SReadline.hpp](#).

Referenced by [GetLine\(\)](#).

**32.153.3.2** `template<typename Container > std::string swift::SReadline::GetLine ( const std::string & Prompt, Container & ReadTokens ) [inline]`

Gets a single line from a user.

## Parameters

<i>Prompt</i>	A printed prompt
<i>ReadTokens</i>	A user inputed string splitted into tokens. The container is cleared first

## Returns

A string which was actually inputed

Definition at line 487 of file [SReadline.hpp](#).

References [GetLine\(\)](#).

32.153.3.3 `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

<i>Prompt</i>	A printed prompt
<i>BreakOut</i>	it is set to true if the EOF found
<i>ReadTokens</i>	A user inputted string splitted into tokens. The container is cleared first

## Returns

A string which was actually inputted

Definition at line 502 of file [SReadline.hpp](#).

References [GetLine\(\)](#).

**32.153.3.4** `std::string swift::SReadline::GetLine ( const std::string & Prompt, bool & BreakOut )` `[inline]`

Gets a single line from a user.

## Parameters

<i>Prompt</i>	A printed prompt
<i>BreakOut</i>	it is set to true if the EOF found

## Returns

A string which was actually inputted

Definition at line 517 of file [SReadline.hpp](#).

**32.153.3.5** `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](#).

**32.153.3.6** `bool swift::SReadline::SaveHistory ( std::ostream & OS )` `[inline]`

Saves the history to the given file stream.

## Parameters

<i>OS</i>	output file stream
-----------	--------------------

## Returns

true if success

Definition at line 564 of file [SReadline.hpp](#).

Referenced by [SaveHistory\(\)](#), and [~SReadline\(\)](#).

**32.153.3.7** `bool swift::SReadline::SaveHistory ( const std::string & FileName )` `[inline]`

Saves the history to the given file.

## Parameters



<i>FileName</i>	File name to save the history to
-----------------	----------------------------------

**Returns**

true if success

Definition at line 581 of file [SReadline.hpp](#).

References [SaveHistory\(\)](#).

**32.153.3.8** 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\(\)](#).

**32.153.3.9** bool swift::SReadline::LoadHistory ( std::istream & *IS* ) [inline]

Loads a history from a file stream.

**Parameters**

<i>IS</i>	Input file stream
-----------	-------------------

**Returns**

true if success

Definition at line 604 of file [SReadline.hpp](#).

References [ClearHistory\(\)](#).

Referenced by [LoadHistory\(\)](#), and [SReadline\(\)](#).

**32.153.3.10** bool swift::SReadline::LoadHistory ( const std::string & *FileName* ) [inline]

Loads a history from the given file.

**Parameters**

<i>FileName</i>	File name to be load from
-----------------	---------------------------

**Returns**

true if success

Definition at line 629 of file [SReadline.hpp](#).

References [LoadHistory\(\)](#).

**32.153.3.11** 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

<i>Container</i>	A container which has all the user possible commands.
------------------	---

Definition at line 658 of file [SReadline.hpp](#).

32.153.3.12 `void swift::SReadline::SetKeymap ( SKeymap & NewKeymap ) [inline]`

Sets the given keymap.

#### Parameters

<i>NewKeymap</i>	The keymap that should be used from now.
------------------	--

Definition at line 675 of file [SReadline.hpp](#).

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

- [stdair/ui/cmdline/SReadline.hpp](#)

## 32.154 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.*

### 32.154.1 Detailed Description

Interface for the STDAIR Services.

Definition at line 44 of file [STDAIR\\_Service.hpp](#).

### 32.154.2 Constructor & Destructor Documentation

#### 32.154.2.1 stdair::STDAIR\_Service::STDAIR\_Service ( )

Default constructor.

Definition at line 45 of file [STDAIR\\_Service.cpp](#).

#### 32.154.2.2 stdair::STDAIR\_Service::STDAIR\_Service ( const BasLogParams & iLogParams )

Constructor.

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

Moreover, a reference on an output stream is given, so that log outputs can be directed onto that stream.

Parameters

<a href="#">in</a>	<a href="#">const</a>	<a href="#">BasLogParams</a> & Parameters for the output log stream.
--------------------	-----------------------	--

Definition at line 61 of file [STDAIR\\_Service.cpp](#).

#### 32.154.2.3 stdair::STDAIR\_Service::STDAIR\_Service ( const BasLogParams & iLogParams, const BasDBParams & iDBParams )

Constructor.

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

A reference on an output stream is given, so that log outputs can be directed onto that stream.

Moreover, database connection parameters are given, so that database events can use the corresponding access.

## Parameters

in	const	<a href="#">BasLogParams</a> & Parameters for the output log stream.
in	const	<a href="#">BasDBParams</a> & Parameters for the database session.

Definition at line 75 of file [STDAIR\\_Service.cpp](#).

### 32.154.2.4 stdair::STDAIR\_Service::~~STDAIR\_Service ( )

Destructor.

Definition at line 93 of file [STDAIR\\_Service.cpp](#).

### 32.154.3 Member Function Documentation

#### 32.154.3.1 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](#).

#### 32.154.3.2 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

<i>const</i>	CabinCapacity_T& Cabin capacity for revenue management optimisation.
--------------	--

Definition at line 187 of file [STDAIR\\_Service.cpp](#).

### 32.154.3.3 void stdair::STDAIR\_Service::buildDummyLegSegmentAccesses ( BomRoot & iBomRoot )

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

<i>BomRoot&amp;</i>	Top of the BOM tree, to which the sample should be attached.
---------------------	--

Definition at line 204 of file [STDAIR\\_Service.cpp](#).

### 32.154.3.4 void stdair::STDAIR\_Service::buildSampleTravelSolutionForPricing ( TravelSolutionList\_T & ioTravelSolutionList )

Build a sample list of travel solutions.

As of now (March 2011), that list is made of the following travel solutions:

- BA9
- LHR-SYD
- 2011-06-10

## Parameters

<i>TravelSolutionList_T&amp;</i>	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](#).

### 32.154.3.5 void stdair::STDAIR\_Service::buildSampleTravelSolutions ( TravelSolutionList\_T & ioTravelSolutionList )

Build a sample list of travel solutions.

As of now (March 2011), that list is made of the following travel solutions:

- BA9
- LHR-SYD
- 2011-06-10
- Q
- WTP: 900
- Change fee: 20; Non refundable; Saturday night stay

## Parameters

<i>TravelSolutionList_T&amp;</i>	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](#).

```
32.154.3.6 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

<i>const</i>	bool <i>isForCRS</i> Whether the sample booking request is for CRS.
--------------	---

#### Returns

[BookingRequestStruct](#)& Sample booking request structure.

Definition at line 229 of file [STDAIR\\_Service.cpp](#).

```
32.154.3.7 void stdair::STDAIR_Service::clonePersistentBom ( )
```

Clone the persistent Bom.

Definition at line 635 of file [STDAIR\\_Service.cpp](#).

References [stdair::FacSupervisor::cleanCloneBomLayer\(\)](#), and [stdair::FacSupervisor::instance\(\)](#).

```
32.154.3.8 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

<i>const</i>	AirlineCode& Airline for which the flight-dates should be displayed. If set to "all" (default), all the inventories will be displayed.
<i>const</i>	FlightNumber_T& Flight number for which all the departure dates should be displayed. If set to 0 (the default), all the flight numbers will be displayed.

Definition at line 242 of file [STDAIR\\_Service.cpp](#).

References [stdair::BomJSONExport::jsonExportFlightDateList\(\)](#).

**32.154.3.9** `std::string stdair::STDAIR_Service::jsonExportFlightDateObjects ( const AirlineCode_T & iAirlineCode, const FlightNumber_T & iFlightNumber, 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

<i>const</i>	AirlineCode_T& Airline code of the flight to dump.
<i>const</i>	FlightNumber_T& Flight number of the flight to dump.
<i>const</i>	Date_T& Departure date of the flight to dump.

## Returns

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

Definition at line 262 of file [STDAIR\\_Service.cpp](#).

References [stdair::BomJSONExport::jsonExportFlightDateObjects\(\)](#), and [stdair::BomRetriever::retrieveFlightDateFromKeySet\(\)](#).

**32.154.3.10** `std::string stdair::STDAIR_Service::jsonExportEventObject ( const EventStruct & iEventStruct ) 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\\_NE\\_T](#), [stdair::EventType::RM](#), [stdair::EventType::SKD\\_CHG](#), and [stdair::EventType::SNAPSHOT](#).

**32.154.3.11** `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::ConfigHolderStruct::jsonExport\(\)](#).

**32.154.3.12** `bool stdair::STDAIR_Service::jsonImportConfiguration ( const JSONString & iJSONString ) const`

Extract the configuration ptree from the given JSON-formatted string and add it to the configuration holder



## Parameters

<i>const</i>	<a href="#">JSONString</a> & JSON-formatted string.
--------------	---

## Returns

bool State whether the extracting has been successful.

Definition at line 342 of file [STDAIR\\_Service.cpp](#).

References [stdair::BomJSONImport::jsonImportConfig\(\)](#).

**32.154.3.13** `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

<i>const</i>	AirlineCode& Airline for which the flight-dates should be displayed. If set to "all" (the default), all the inventories will be displayed.
<i>const</i>	FlightNumber_T& Flight number for which all the departure dates should be displayed. If set to 0 (the default), all the flight numbers will be displayed.

## Returns

std::string Output string in which the BOM tree is logged/dumped.

Definition at line 428 of file [STDAIR\\_Service.cpp](#).

References [stdair::BomDisplay::list\(\)](#).

**32.154.3.14** `std::string stdair::STDAIR_Service::listAirportPairDateRange ( ) const`

Display the list of airports pairs and date ranges (contained within the BOM tree)

## Parameters

<i>std::ostream&amp;</i>	Output stream in which the airport pairs and date ranges are logged/dumped.
--------------------------	---

Definition at line 446 of file [STDAIR\\_Service.cpp](#).

References [stdair::BomDisplay::listAirportPairDateRange\(\)](#).

**32.154.3.15** `bool stdair::STDAIR_Service::check ( const AirlineCode_T & iAirlineCode, const FlightNumber_T & iFlightNumber, const Date_T & iDepartureDate ) const`

Check whether the given flight-date is a valid one.

## Parameters

<i>const</i>	<a href="#">stdair::AirlineCode_T</a> & Airline code of the flight to check.
<i>const</i>	<a href="#">stdair::FlightNumber_T</a> & Flight number of the flight to check.
<i>const</i>	<a href="#">stdair::Date_T</a> & 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::BomRetriever::retrieveFlightDateFromKeySet\(\)](#).

32.154.3.16 `bool stdair::STDAIR_Service::check ( const AirportCode_T & ioOrigin, const AirportCode_T & ioDestination,  
const Date_T & iDepartureDate ) const`

Check whether the given couple airportpair-date is a valid one.

## Parameters

<i>const</i>	<a href="#">stdair::AirportCode_T</a> & Origin airport of the fare rule to check.
<i>const</i>	<a href="#">stdair::AirportCode_T</a> & Destination airport of the fare rule to check.
<i>const</i>	<a href="#">stdair::Date_T</a> & 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::BomRetriever::retrieveDatePeriodListFromKeySet\(\)](#).

32.154.3.17 `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\(\)](#).

32.154.3.18 `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](#).

32.154.3.19 `std::string stdair::STDAIR_Service::csvDisplay ( const BomRoot & iBomRoot ) const`

Recursively display (dump in the returned string) the objects of the BOM tree.

## Parameters

<i>const</i>	<a href="#">BomRoot</a> & Reference on the <a href="#">BomRoot</a> 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\(\)](#).

32.154.3.20 `std::string stdair::STDAIR_Service::csvDisplay ( const AirlineCode_T & iAirlineCode, const FlightNumber_T & iFlightNumber, const Date_T & iDepartureDate ) const`

Recursively display (dump in the returned string) the flight-date corresponding to the parameters given as input.

## Parameters

<i>const</i>	AirlineCode_T& Airline code of the flight to display.
<i>const</i>	FlightNumber_T& Flight number of the flight to display.
<i>const</i>	Date_T& Departure date of the flight to display.

**Returns**

std::string Output string in which the BOM tree is logged/dumped.

Definition at line 555 of file [STDAIR\\_Service.cpp](#).

References [stdair::BomDisplay::csvDisplay\(\)](#), and [stdair::BomRetriever::retrieveFlightDateFromKeySet\(\)](#).

**32.154.3.21** std::string stdair::STDAIR\_Service::csvDisplay ( const TravelSolutionList\_T & iTravelSolutionList ) const

Display (dump in the returned string) the full list of travel solution structures.

**Returns**

std::string Output string in which the list of travel solutions is logged/dumped.

Definition at line 587 of file [STDAIR\\_Service.cpp](#).

References [stdair::BomDisplay::csvDisplay\(\)](#).

**32.154.3.22** std::string stdair::STDAIR\_Service::csvDisplay ( const AirportCode\_T & iOrigin, const AirportCode\_T & iDestination, const Date\_T & iDepartureDate ) const

Recursively display (dump in the returned string) the fare-rules corresponding to the parameters given as input.

**Parameters**

<i>const</i>	AirportCode_T& Origin airport of the fare-rules to display
<i>const</i>	AirportCode_T& Destination airport of the fare-rules to display.
<i>const</i>	Date_T& Departure date of the fare-rules to display.

**Returns**

std::string Output string in which the BOM tree is logged/dumped.

Definition at line 598 of file [STDAIR\\_Service.cpp](#).

References [stdair::BomDisplay::csvDisplay\(\)](#), and [stdair::BomRetriever::retrieveDatePeriodListFromKeySet\(\)](#).

**32.154.3.23** 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](#).

**32.154.3.24** 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](#).

### 32.154.3.25 **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](#).

### 32.154.3.26 **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](#).

### 32.154.3.27 **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](#).

### 32.154.3.28 **void** stdair::STDAIR\_Service::importINIConfig ( const **ConfigINIFile** & *iConfigINIFile* )

Import the configuration INI input file (format cfg).

#### Parameters

<i>const</i>	<a href="#">ConfigINIFile</a> & INI input file.
--------------	---

Definition at line 375 of file [STDAIR\\_Service.cpp](#).

References [stdair::BomINIImport::importINIConfig\(\)](#).

### 32.154.3.29 **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

<i>const</i>	std::string& Value to add in the configuration tree.
<i>const</i>	std::string& Path to create (or to look for).

Definition at line 391 of file [STDAIR\\_Service.cpp](#).

References [stdair::ConfigHolderStruct::addValue\(\)](#).

### 32.154.3.30 **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

<i>ValueType</i> &	Value to add in the configuration tree.
<i>const</i>	std::string& Path to look for.

Definition at line 552 of file [STDAIR\\_Service.hpp](#).

References [stdair::ConfigHolderStruct::exportValue\(\)](#).

### 32.154.3.31 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::ConfigHolderStruct::updateAirlineFeatures\(\)](#).

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

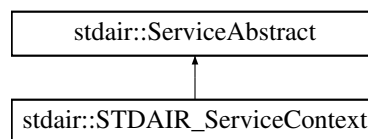
- [stdair/STDAIR\\_Service.hpp](#)
- [stdair/service/STDAIR\\_Service.cpp](#)

## 32.155 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](#)

### 32.155.1 Detailed Description

Class holding the context of the Stdair services.

Definition at line 25 of file [STDAIR\\_ServiceContext.hpp](#).

### 32.155.2 Member Function Documentation

#### 32.155.2.1 virtual void stdair::ServiceAbstract::toStream ( std::ostream & ioOut ) const [inline], [virtual], [inherited]

Dump a Business Object into an output stream.

## Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Definition at line 28 of file [ServiceAbstract.hpp](#).

**32.155.2.2** `virtual void stdair::ServiceAbstract::fromStream ( std::istream & ioIn )` `[inline]`, `[virtual]`, `[inherited]`

Read a Business Object from an input stream.

## Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Definition at line 35 of file [ServiceAbstract.hpp](#).

Referenced by [operator>>\(\)](#).

### 32.155.3 Friends And Related Function Documentation

**32.155.3.1** `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](#).

**32.155.3.2** `friend class FacSTDAIRServiceContext` `[friend]`

Definition at line 30 of file [STDAIR\\_ServiceContext.hpp](#).

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

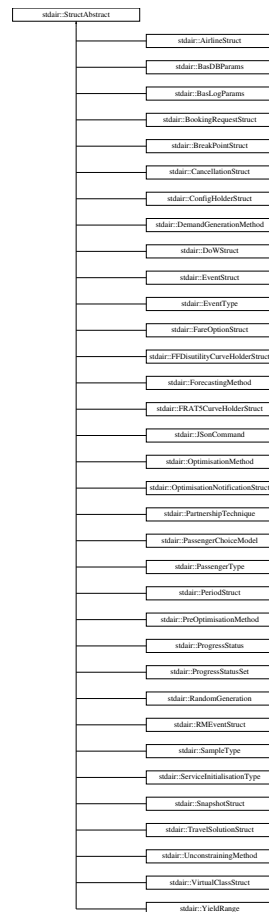
- [stdair/service/STDAIR\\_ServiceContext.hpp](#)

## 32.156 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](#) ()

### 32.156.1 Detailed Description

Base class for the light structures.

Definition at line 16 of file [StructAbstract.hpp](#).

### 32.156.2 Constructor & Destructor Documentation

**32.156.2.1** virtual stdair::StructAbstract::~[~StructAbstract](#) ( ) [inline],[virtual]

Destructor.

Definition at line 22 of file [StructAbstract.hpp](#).



**32.156.2.2** `stdair::StructAbstract::StructAbstract ( ) [inline], [protected]`

Protected Default Constructor to ensure this class is abstract.

Definition at line 49 of file [StructAbstract.hpp](#).

### 32.156.3 Member Function Documentation

**32.156.3.1** `void stdair::StructAbstract::toStream ( std::ostream & ioOut ) const [inline]`

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Definition at line 29 of file [StructAbstract.hpp](#).

References [describe\(\)](#).

**32.156.3.2** `virtual void stdair::StructAbstract::fromStream ( std::istream & ioln ) [inline], [virtual]`

Read a Business Object from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented in [stdair::BookingRequestStruct](#), [stdair::EventStruct](#), [stdair::TravelSolutionStruct](#), [stdair::VirtualClassStruct](#), [stdair::ConfigHolderStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::ProgressStatusSet](#), [stdair::YieldRange](#), [stdair::CancellationStruct](#), [stdair::AirlineStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::FFDisutilityCurveHolderStruct](#), [stdair::FRAT5CurveHolderStruct](#), and [stdair::BreakPointStruct](#).

Definition at line 38 of file [StructAbstract.hpp](#).

Referenced by [operator>>\(\)](#).

**32.156.3.3** `virtual const std::string stdair::StructAbstract::describe ( ) const [pure virtual]`

Display of the structure.

Implemented in [stdair::BookingRequestStruct](#), [stdair::EventStruct](#), [stdair::TravelSolutionStruct](#), [stdair::ConfigHolderStruct](#), [stdair::FareOptionStruct](#), [stdair::VirtualClassStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::ProgressStatus](#), [stdair::BasDBParams](#), [stdair::ProgressStatusSet](#), [stdair::YieldRange](#), [stdair::PartnershipTechnique](#), [stdair::SampleType](#), [stdair::ServiceInitialisationType](#), [stdair::DemandGenerationMethod](#), [stdair::CancellationStruct](#), [stdair::BasLogParams](#), [stdair::RandomGeneration](#), [stdair::EventType](#), [stdair::JsonCommand](#), [stdair::AirlineStruct](#), [stdair::ForecastingMethod](#), [stdair::RMEventStruct](#), [stdair::PassengerChoiceModel](#), [stdair::OptimisationMethod](#), [stdair::PassengerType](#), [stdair::PreOptimisationMethod](#), [stdair::UnconstrainingMethod](#), [stdair::SnapshotStruct](#), [stdair::PeriodStruct](#), [stdair::DoWStruct](#), [stdair::FFDisutilityCurveHolderStruct](#), [stdair::FRAT5CurveHolderStruct](#), and [stdair::BreakPointStruct](#).

Referenced by [toStream\(\)](#).

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

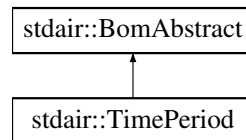
- [stdair/basic/StructAbstract.hpp](#)

## 32.157 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

- template<typename BOM >  
class [FacBom](#)
- template<typename BOM >  
class [FacCloneBom](#)
- class [FacBomManager](#)

#### 32.157.1 Detailed Description

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

Definition at line 18 of file [TimePeriod.hpp](#).

#### 32.157.2 Member Typedef Documentation

##### 32.157.2.1 typedef TimePeriodKey stdair::TimePeriod::Key\_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 28 of file [TimePeriod.hpp](#).

### 32.157.3 Constructor & Destructor Documentation

#### 32.157.3.1 `stdair::TimePeriod::TimePeriod ( const Key_T & iKey )` [protected]

Main constructor.

Definition at line 27 of file [TimePeriod.cpp](#).

#### 32.157.3.2 `stdair::TimePeriod::~~TimePeriod ( )` [protected],[virtual]

Destructor.

Definition at line 32 of file [TimePeriod.cpp](#).

### 32.157.4 Member Function Documentation

#### 32.157.4.1 `void stdair::TimePeriod::toStream ( std::ostream & ioOut ) const` [inline],[virtual]

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Implements [stdair::BomAbstract](#).

Definition at line 38 of file [TimePeriod.hpp](#).

References [toString\(\)](#).

#### 32.157.4.2 `void stdair::TimePeriod::fromStream ( std::istream & ioin )` [inline],[virtual]

Read a Business Object from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Implements [stdair::BomAbstract](#).

Definition at line 47 of file [TimePeriod.hpp](#).

#### 32.157.4.3 `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\(\)](#).

#### 32.157.4.4 `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\(\)](#).

#### 32.157.4.5 `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](#).

**32.157.4.6 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](#).

**32.157.4.7 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](#).

**32.157.4.8 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\(\)](#).

**32.157.4.9 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\(\)](#).

**32.157.4.10 bool** `stdair::TimePeriod::isDepartureTimeValid ( const Time_T & iFlightTime ) const`

Check if the given departure time is included in the departure period of the segment path.

Definition at line 44 of file [TimePeriod.cpp](#).

References [getTimeRangeEnd\(\)](#), [getTimeRangeStart\(\)](#), and [STDAIR\\_LOG\\_DEBUG](#).

## 32.157.5 Friends And Related Function Documentation

**32.157.5.1 template<typename BOM > friend class FacBom** [friend]

Definition at line 19 of file [TimePeriod.hpp](#).

**32.157.5.2 template<typename BOM > friend class FacCloneBom** [friend]

Definition at line 20 of file [TimePeriod.hpp](#).

**32.157.5.3 friend class FacBomManager** [friend]

Definition at line 21 of file [TimePeriod.hpp](#).

## 32.157.6 Member Data Documentation

**32.157.6.1 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\(\)](#).

**32.157.6.2 BomAbstract\* stdair::TimePeriod::\_parent** [protected]

Pointer on the parent class ([Inventory](#)).

Definition at line 138 of file [TimePeriod.hpp](#).

Referenced by [getParent\(\)](#).

**32.157.6.3 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:

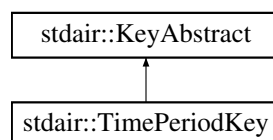
- [stdair/bom/TimePeriod.hpp](#)
- [stdair/bom/TimePeriod.cpp](#)

**32.158 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

**32.158.1 Detailed Description**

Key of time-period.

Definition at line 15 of file [TimePeriodKey.hpp](#).

## 32.158.2 Constructor &amp; Destructor Documentation

## 32.158.2.1 stdair::TimePeriodKey::TimePeriodKey ( const Time\_T &amp; iTimeRangeStart, const Time\_T &amp; iTimeRangeEnd )

Main constructor.

Definition at line 21 of file [TimePeriodKey.cpp](#).

## 32.158.2.2 stdair::TimePeriodKey::TimePeriodKey ( const TimePeriodKey &amp; iKey )

Copy constructor.

Definition at line 28 of file [TimePeriodKey.cpp](#).

## 32.158.2.3 stdair::TimePeriodKey::~~TimePeriodKey ( )

Destructor.

Definition at line 34 of file [TimePeriodKey.cpp](#).

## 32.158.3 Member Function Documentation

## 32.158.3.1 const Time\_T&amp; stdair::TimePeriodKey::getTimeRangeStart ( ) const [inline]

Get the time period start.

Definition at line 35 of file [TimePeriodKey.hpp](#).

Referenced by [stdair::TimePeriod::getTimeRangeStart\(\)](#).

## 32.158.3.2 const Time\_T&amp; stdair::TimePeriodKey::getTimeRangeEnd ( ) const [inline]

Get the time period end.

Definition at line 42 of file [TimePeriodKey.hpp](#).

Referenced by [stdair::TimePeriod::getTimeRangeEnd\(\)](#).

## 32.158.3.3 void stdair::TimePeriodKey::toStream ( std::ostream &amp; ioOut ) const [virtual]

Dump a Business Object Key into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 38 of file [TimePeriodKey.cpp](#).

References [toString\(\)](#).

## 32.158.3.4 void stdair::TimePeriodKey::fromStream ( std::istream &amp; ioIn ) [virtual]

Read a Business Object Key from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 43 of file [TimePeriodKey.cpp](#).

### 32.158.3.5 `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:

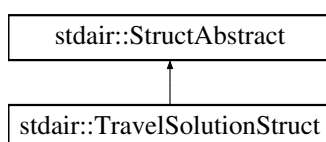
- [stdair/bom/TimePeriodKey.hpp](#)
- [stdair/bom/TimePeriodKey.cpp](#)

## 32.159 `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 ( )`

### 32.159.1 Detailed Description

Structure holding the elements of a travel solution.

Definition at line 24 of file [TravelSolutionStruct.hpp](#).

### 32.159.2 Constructor & Destructor Documentation

#### 32.159.2.1 stdair::TravelSolutionStruct::TravelSolutionStruct ( )

Default constructor.

Definition at line 15 of file [TravelSolutionStruct.cpp](#).

#### 32.159.2.2 stdair::TravelSolutionStruct::~~TravelSolutionStruct ( )

Destructor.

Definition at line 19 of file [TravelSolutionStruct.cpp](#).

### 32.159.3 Member Function Documentation

#### 32.159.3.1 const SegmentPath\_T& stdair::TravelSolutionStruct::getSegmentPath ( ) const [inline]

Get the segment path.

Definition at line 28 of file [TravelSolutionStruct.hpp](#).

#### 32.159.3.2 const ClassAvailabilityMapHolder\_T& stdair::TravelSolutionStruct::getClassAvailabilityMapHolder ( ) const [inline]

Get the holder of availabilities.

Definition at line 33 of file [TravelSolutionStruct.hpp](#).

#### 32.159.3.3 const ClassObjectIDMapHolder\_T& stdair::TravelSolutionStruct::getClassObjectIDMapHolder ( ) const [inline]

Get the holder of object ID's.

Definition at line 38 of file [TravelSolutionStruct.hpp](#).

#### 32.159.3.4 const ClassYieldMapHolder\_T& stdair::TravelSolutionStruct::getClassYieldMapHolder ( ) const [inline]

Get the holder of yields.

Definition at line 43 of file [TravelSolutionStruct.hpp](#).

#### 32.159.3.5 const BidPriceVectorHolder\_T& stdair::TravelSolutionStruct::getBidPriceVectorHolder ( ) const [inline]

Get the holder of bid price vectors.

Definition at line 48 of file [TravelSolutionStruct.hpp](#).

#### 32.159.3.6 const ClassBpvMapHolder\_T& stdair::TravelSolutionStruct::getClassBpvMapHolder ( ) const [inline]

Get the holder of class - bid price reference.

Definition at line 53 of file [TravelSolutionStruct.hpp](#).



**32.159.3.7** `const FareOptionList_T& stdair::TravelSolutionStruct::getFareOptionList ( ) const` `[inline]`

Get the list of fare options.

Definition at line 58 of file [TravelSolutionStruct.hpp](#).

**32.159.3.8** `FareOptionList_T& stdair::TravelSolutionStruct::getFareOptionListRef ( )` `[inline]`

Get the non-const list of fare options.

Definition at line 63 of file [TravelSolutionStruct.hpp](#).

**32.159.3.9** `const FareOptionStruct& stdair::TravelSolutionStruct::getChosenFareOption ( ) const` `[inline]`

Get the chosen fare option.

Definition at line 68 of file [TravelSolutionStruct.hpp](#).

**32.159.3.10** `void stdair::TravelSolutionStruct::addSegment ( const std::string & iKey )`

Add a segment key to the segment path.

Definition at line 154 of file [TravelSolutionStruct.cpp](#).

**32.159.3.11** `void stdair::TravelSolutionStruct::addClassAvailabilityMap ( const ClassAvailabilityMap_T & iMap )`

Add a class availability map.

Definition at line 160 of file [TravelSolutionStruct.cpp](#).

**32.159.3.12** `void stdair::TravelSolutionStruct::addClassObjectIDMap ( const ClassObjectIDMap_T & iMap )`

Add a class object ID map.

Definition at line 166 of file [TravelSolutionStruct.cpp](#).

**32.159.3.13** `void stdair::TravelSolutionStruct::addClassYieldMap ( const ClassYieldMap_T & iMap )`

Add a class yield map.

Definition at line 172 of file [TravelSolutionStruct.cpp](#).

**32.159.3.14** `void stdair::TravelSolutionStruct::addBidPriceVector ( const BidPriceVector_T & iBpv )`

Add a bid price vector.

Definition at line 178 of file [TravelSolutionStruct.cpp](#).

**32.159.3.15** `void stdair::TravelSolutionStruct::addClassBpvMap ( const ClassBpvMap_T & iMap )`

Add a class bpv reference map.

Definition at line 184 of file [TravelSolutionStruct.cpp](#).

**32.159.3.16** `void stdair::TravelSolutionStruct::addFareOption ( const FareOptionStruct & iFareOption )`

Add a fare option.

Definition at line 190 of file [TravelSolutionStruct.cpp](#).

**32.159.3.17** `void stdair::TravelSolutionStruct::setChosenFareOption ( const FareOptionStruct & iChosenFO )`  
`[inline]`

Set the chosen fare option.

Definition at line 97 of file [TravelSolutionStruct.hpp](#).

32.159.3.18 void stdair::TravelSolutionStruct::toStream ( std::ostream & *ioOut* ) const

Dump a Business Object into an output stream.

## Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Definition at line 23 of file [TravelSolutionStruct.cpp](#).

References [describe\(\)](#).

32.159.3.19 `void stdair::TravelSolutionStruct::fromStream ( std::istream & ioln ) [virtual]`

Read a Business Object from an input stream.

## Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented from [stdair::StructAbstract](#).

Definition at line 28 of file [TravelSolutionStruct.cpp](#).

32.159.3.20 `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\(\)](#).

32.159.3.21 `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\(\)](#).

32.159.3.22 `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](#)

## 32.160 `soci::type_conversion< stdair::AirlineStruct >` Struct Template Reference

```
#include <stdair/dbadaptor/DbAirline.hpp>
```

## Public Types

- typedef values [base\\_type](#)

**Static Public Member Functions**

- static void [from\\_base](#) (values const &iAirlineValues, indicator, [stdair::AirlineStruct](#) &ioAirline)
- static void [to\\_base](#) (const [stdair::AirlineStruct](#) &iAirline, values &ioAirlineValues, indicator &ioIndicator)

**32.160.1 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 [Dbairline.hpp](#).

**32.160.2 Member Typedef Documentation****32.160.2.1 typedef values soci::type\_conversion< stdair::AirlineStruct >::base\_type**

Definition at line 27 of file [Dbairline.hpp](#).

**32.160.3 Member Function Documentation****32.160.3.1 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 [Dbairline.cpp](#).

References [stdair::AirlineStruct::setAirlineCode\(\)](#), and [stdair::AirlineStruct::setAirlineName\(\)](#).

**32.160.3.2 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 [Dbairline.cpp](#).

References [stdair::AirlineStruct::getAirlineCode\(\)](#), and [stdair::AirlineStruct::getAirlineName\(\)](#).

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

- [stdair/dbadaptor/Dbairline.hpp](#)
- [stdair/dbadaptor/Dbairline.cpp](#)

**32.161 TypeWithSize< size > Class Template Reference**

```
#include <stdair/basic/float_utils_google.hpp>
```

**Public Types**

- typedef void [UInt](#)

**32.161.1 Detailed Description**

```
template<size_t size>class TypeWithSize< size >
```

Definition at line 54 of file [float\\_utils\\_google.hpp](#).

### 32.161.2 Member Typedef Documentation

#### 32.161.2.1 `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](#)

### 32.162 `TypeWithSize< 4 >` Class Template Reference

```
#include <stdair/basic/float_utils_google.hpp>
```

#### Public Types

- typedef int [Int](#)
- typedef unsigned int [UInt](#)

#### 32.162.1 Detailed Description

```
template<>class TypeWithSize< 4 >
```

Definition at line 63 of file [float\\_utils\\_google.hpp](#).

#### 32.162.2 Member Typedef Documentation

##### 32.162.2.1 `typedef int TypeWithSize< 4 >::Int`

Definition at line 69 of file [float\\_utils\\_google.hpp](#).

##### 32.162.2.2 `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](#)

### 32.163 `TypeWithSize< 8 >` Class Template Reference

```
#include <stdair/basic/float_utils_google.hpp>
```

#### Public Types

- typedef long long [Int](#)
- typedef unsigned long long [UInt](#)

#### 32.163.1 Detailed Description

```
template<>class TypeWithSize< 8 >
```

Definition at line 75 of file [float\\_utils\\_google.hpp](#).

## 32.163.2 Member Typedef Documentation

## 32.163.2.1 typedef long long TypeWithSize&lt; 8 &gt;::Int

Definition at line 81 of file [float\\_utils\\_google.hpp](#).

## 32.163.2.2 typedef unsigned long long TypeWithSize&lt; 8 &gt;::UInt

Definition at line 82 of file [float\\_utils\\_google.hpp](#).

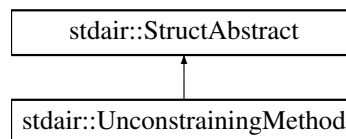
The documentation for this class was generated from the following file:

- [stdair/basic/float\\_utils\\_google.hpp](#)

## 32.164 stdair::UnconstrainingMethod Struct Reference

```
#include <stdair/basic/UnconstrainingMethod.hpp>
```

Inheritance diagram for stdair::UnconstrainingMethod:



## Public Types

- 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](#) ()

## 32.164.1 Detailed Description

Enumeration of unconstraining methods.

Definition at line 15 of file [UnconstrainingMethod.hpp](#).

### 32.164.2 Member Enumeration Documentation

#### 32.164.2.1 enum `stdair::UnconstrainingMethod::EN_UnconstrainingMethod`

Enumerator

***EM***

***LAST\_VALUE***

Definition at line 17 of file [UnconstrainingMethod.hpp](#).

### 32.164.3 Constructor & Destructor Documentation

#### 32.164.3.1 `stdair::UnconstrainingMethod::UnconstrainingMethod ( const EN_UnconstrainingMethod & iUnconstrainingMethod )`

Constructor.

Definition at line 36 of file [UnconstrainingMethod.cpp](#).

#### 32.164.3.2 `stdair::UnconstrainingMethod::UnconstrainingMethod ( const char iMethod )`

Constructor.

Definition at line 41 of file [UnconstrainingMethod.cpp](#).

References [describeLabels\(\)](#), [EM](#), and [LAST\\_VALUE](#).

#### 32.164.3.3 `stdair::UnconstrainingMethod::UnconstrainingMethod ( const UnconstrainingMethod & iUnconstrainingMethod )`

Default copy constructor.

Definition at line 30 of file [UnconstrainingMethod.cpp](#).

### 32.164.4 Member Function Documentation

#### 32.164.4.1 `const std::string & stdair::UnconstrainingMethod::getLabel ( const EN_UnconstrainingMethod & iMethod ) [static]`

Get the label as a string (e.g., "Expectation-Maximisation")

Definition at line 58 of file [UnconstrainingMethod.cpp](#).

#### 32.164.4.2 `char stdair::UnconstrainingMethod::getMethodLabel ( const EN_UnconstrainingMethod & iMethod ) [static]`

Get the label as a single char (e.g., 'T' or 'B').

Definition at line 63 of file [UnconstrainingMethod.cpp](#).

#### 32.164.4.3 `std::string stdair::UnconstrainingMethod::getMethodLabelAsString ( const EN_UnconstrainingMethod & iMethod ) [static]`

Get the label as a string of a single char (e.g., "T" or "B").

Definition at line 69 of file [UnconstrainingMethod.cpp](#).

#### 32.164.4.4 `std::string stdair::UnconstrainingMethod::describeLabels ( ) [static]`

List the labels.

Definition at line 76 of file [UnconstrainingMethod.cpp](#).

References [LAST\\_VALUE](#).

Referenced by [UnconstrainingMethod\(\)](#).

**32.164.4.5** `UnconstrainingMethod::EN_UnconstrainingMethod stdair::UnconstrainingMethod::getMethod ( ) const`

Get the enumerated value.

Definition at line 88 of file [UnconstrainingMethod.cpp](#).

Referenced by [stdair::AirlineFeature::getUnconstrainingMethod\(\)](#).

**32.164.4.6** `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](#).

**32.164.4.7** `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](#).

**32.164.4.8** `bool stdair::UnconstrainingMethod::operator== ( const EN_UnconstrainingMethod & iMethod ) const`

Comparison operator.

Definition at line 108 of file [UnconstrainingMethod.cpp](#).

**32.164.4.9** `void stdair::StructAbstract::toStream ( std::ostream & ioOut ) const [inline],[inherited]`

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Definition at line 29 of file [StructAbstract.hpp](#).

References [stdair::StructAbstract::describe\(\)](#).

**32.164.4.10** `virtual void stdair::StructAbstract::fromStream ( std::istream & ioIn ) [inline],[virtual],[inherited]`

Read a Business Object from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented in [stdair::BookingRequestStruct](#), [stdair::EventStruct](#), [stdair::TravelSolutionStruct](#), [stdair::VirtualClassStruct](#), [stdair::ConfigHolderStruct](#), [stdair::FareOptionStruct](#), [stdair::OptimisationNotificationStruct](#), [stdair::ProgressStatusSet](#), [stdair::YieldRange](#), [stdair::CancellationStruct](#), [stdair::AirlineStruct](#), [stdair::RMEventStruct](#), [stdair::SnapshotStruct](#), [stdair::FFDisutilityCurveHolderStruct](#), [stdair::FRAT5CurveHolderStruct](#), and [stdair::BreakPointStruct](#).

Definition at line 38 of file [StructAbstract.hpp](#).

Referenced by [operator>>\(\)](#).

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

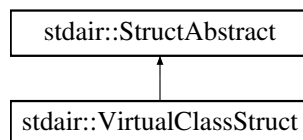
- [stdair/basic/UnconstrainingMethod.hpp](#)
- [stdair/basic/UnconstrainingMethod.cpp](#)



### 32.165 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](#) ()

#### 32.165.1 Detailed Description

Structure holding the elements of a virtual class.

Definition at line 24 of file [VirtualClassStruct.hpp](#).

#### 32.165.2 Constructor & Destructor Documentation

##### 32.165.2.1 stdair::VirtualClassStruct::VirtualClassStruct ( const VirtualClassStruct & iVC )

Default copy constructor.

Definition at line 19 of file [VirtualClassStruct.cpp](#).

##### 32.165.2.2 stdair::VirtualClassStruct::VirtualClassStruct ( const BookingClassList\_T & ioBookingClassList )

Constructor.

Definition at line 26 of file [VirtualClassStruct.cpp](#).

## 32.165.2.3 stdair::VirtualClassStruct::~~VirtualClassStruct ( )

Destructor.

Definition at line 31 of file [VirtualClassStruct.cpp](#).

## 32.165.3 Member Function Documentation

## 32.165.3.1 const BookingClassList\_T&amp; stdair::VirtualClassStruct::getBookingClassList ( ) const [inline]

Get the list of booking class.

Definition at line 28 of file [VirtualClassStruct.hpp](#).

## 32.165.3.2 const Yield\_T&amp; stdair::VirtualClassStruct::getYield ( ) const [inline]

Get the yield (average price paid for that virtual class).

Definition at line 33 of file [VirtualClassStruct.hpp](#).

Referenced by [stdair::LegCabin::displayVirtualClassList\(\)](#).

## 32.165.3.3 const MeanValue\_T&amp; stdair::VirtualClassStruct::getMean ( ) const [inline]

Get the mean value of the demand distribution.

Definition at line 38 of file [VirtualClassStruct.hpp](#).

## 32.165.3.4 const StdDevValue\_T&amp; stdair::VirtualClassStruct::getStdDev ( ) const [inline]

Get the standard deviation of the demand distribution.

Definition at line 43 of file [VirtualClassStruct.hpp](#).

## 32.165.3.5 const BookingLimit\_T&amp; stdair::VirtualClassStruct::getCumulatedBookingLimit ( ) const [inline]

Get the booking limit of the class.

Definition at line 48 of file [VirtualClassStruct.hpp](#).

Referenced by [stdair::LegCabin::displayVirtualClassList\(\)](#).

## 32.165.3.6 const ProtectionLevel\_T&amp; stdair::VirtualClassStruct::getCumulatedProtection ( ) const [inline]

Get the protection level of the class.

Definition at line 53 of file [VirtualClassStruct.hpp](#).

Referenced by [stdair::LegCabin::displayVirtualClassList\(\)](#).

## 32.165.3.7 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\(\)](#).

## 32.165.3.8 void stdair::VirtualClassStruct::setYield ( const Yield\_T &amp; iYield ) [inline]

Set the yield (average price paid for that virtual class).

Definition at line 63 of file [VirtualClassStruct.hpp](#).

## 32.165.3.9 void stdair::VirtualClassStruct::setMean ( const MeanValue\_T &amp; iMean ) [inline]

Set the mean value of the demand distribution.

Definition at line 68 of file [VirtualClassStruct.hpp](#).

32.165.3.10 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](#).

32.165.3.11 void stdair::VirtualClassStruct::setCumulatedBookingLimit ( const BookingLimit\_T & iBL ) [inline]

Set the booking limit of the class.

Definition at line 78 of file [VirtualClassStruct.hpp](#).

32.165.3.12 void stdair::VirtualClassStruct::setCumulatedProtection ( const ProtectionLevel\_T & iP ) [inline]

Set the protection level of the class.

Definition at line 83 of file [VirtualClassStruct.hpp](#).

32.165.3.13 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](#).

32.165.3.14 void stdair::VirtualClassStruct::toStream ( std::ostream & ioOut ) const

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Definition at line 36 of file [VirtualClassStruct.cpp](#).

References [describe\(\)](#).

32.165.3.15 void stdair::VirtualClassStruct::fromStream ( std::istream & ioIn ) [virtual]

Read a Business Object from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented from [stdair::StructAbstract](#).

Definition at line 41 of file [VirtualClassStruct.cpp](#).

32.165.3.16 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:

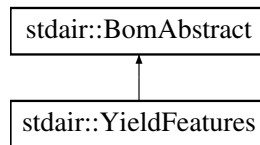
- [stdair/bom/VirtualClassStruct.hpp](#)
- [stdair/bom/VirtualClassStruct.cpp](#)

## 32.166 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

- template<typename BOM >  
class [FacBom](#)
- template<typename BOM >  
class [FacCloneBom](#)
- class [FacBomManager](#)

#### 32.166.1 Detailed Description

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

Definition at line 19 of file [YieldFeatures.hpp](#).

### 32.166.2 Member Typedef Documentation

#### 32.166.2.1 typedef YieldFeaturesKey stdair::YieldFeatures::Key\_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 29 of file [YieldFeatures.hpp](#).

### 32.166.3 Constructor & Destructor Documentation

#### 32.166.3.1 stdair::YieldFeatures::YieldFeatures ( const Key\_T & iKey ) [protected]

Main constructor.

Definition at line 28 of file [YieldFeatures.cpp](#).

#### 32.166.3.2 stdair::YieldFeatures::~~YieldFeatures ( ) [protected],[virtual]

Destructor.

Definition at line 33 of file [YieldFeatures.cpp](#).

### 32.166.4 Member Function Documentation

#### 32.166.4.1 void stdair::YieldFeatures::toStream ( std::ostream & ioOut ) const [inline],[virtual]

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Implements [stdair::BomAbstract](#).

Definition at line 38 of file [YieldFeatures.hpp](#).

References [toString\(\)](#).

#### 32.166.4.2 void stdair::YieldFeatures::fromStream ( std::istream & ioln ) [inline],[virtual]

Read a Business Object from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Implements [stdair::BomAbstract](#).

Definition at line 47 of file [YieldFeatures.hpp](#).

#### 32.166.4.3 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\(\)](#).

#### 32.166.4.4 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\(\)](#).

**32.166.4.5** `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](#).

**32.166.4.6** `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](#).

**32.166.4.7** `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](#).

**32.166.4.8** `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\(\)](#).

**32.166.4.9** `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\(\)](#).

**32.166.4.10** `bool stdair::YieldFeatures::isTripTypeValid ( const TripType_T & iBookingRequestTripType ) const`

Check whether the fare rule trip type corresponds to the booking request trip type.

Definition at line 45 of file [YieldFeatures.cpp](#).

References [getTripType\(\)](#), [stdair::TRIP\\_TYPE\\_INBOUND](#), [stdair::TRIP\\_TYPE\\_OUTBOUND](#), and [stdair::TRIP\\_TYPE\\_ROUND\\_TRIP](#).

## 32.166.5 Friends And Related Function Documentation

**32.166.5.1** `template<typename BOM> friend class FacBom` [\[friend\]](#)

Definition at line 20 of file [YieldFeatures.hpp](#).

**32.166.5.2** `template<typename BOM> friend class FacCloneBom` [\[friend\]](#)

Definition at line 21 of file [YieldFeatures.hpp](#).

### 32.166.5.3 friend class **FacBomManager** [*friend*]

Definition at line 22 of file [YieldFeatures.hpp](#).

### 32.166.6 Member Data Documentation

#### 32.166.6.1 **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\(\)](#).

#### 32.166.6.2 **BomAbstract\*** stdair::YieldFeatures::\_parent [*protected*]

Pointer on the parent class.

Definition at line 143 of file [YieldFeatures.hpp](#).

Referenced by [getParent\(\)](#).

#### 32.166.6.3 **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:

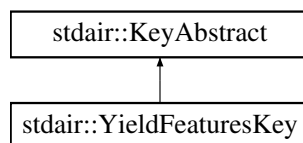
- [stdair/bom/YieldFeatures.hpp](#)
- [stdair/bom/YieldFeatures.cpp](#)

## 32.167 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

### 32.167.1 Detailed Description

Key of date-period.

Definition at line 18 of file [YieldFeaturesKey.hpp](#).

### 32.167.2 Constructor & Destructor Documentation

#### 32.167.2.1 stdair::YieldFeaturesKey::YieldFeaturesKey ( const TripType\_T & iTripType, const CabinCode\_T & iCabin )

Main constructor.

Definition at line 21 of file [YieldFeaturesKey.cpp](#).

#### 32.167.2.2 stdair::YieldFeaturesKey::YieldFeaturesKey ( const YieldFeaturesKey & iKey )

Copy constructor.

Definition at line 27 of file [YieldFeaturesKey.cpp](#).

#### 32.167.2.3 stdair::YieldFeaturesKey::~YieldFeaturesKey ( )

Destructor.

Definition at line 32 of file [YieldFeaturesKey.cpp](#).

### 32.167.3 Member Function Documentation

#### 32.167.3.1 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\(\)](#).

#### 32.167.3.2 const CabinCode\_T& stdair::YieldFeaturesKey::getCabinCode ( ) const [inline]

Get the cabin.

Definition at line 51 of file [YieldFeaturesKey.hpp](#).

Referenced by [stdair::YieldFeatures::getCabinCode\(\)](#).

#### 32.167.3.3 void stdair::YieldFeaturesKey::toStream ( std::ostream & ioOut ) const [virtual]

Dump a Business Object Key into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 36 of file [YieldFeaturesKey.cpp](#).

References [toString\(\)](#).

#### 32.167.3.4 void stdair::YieldFeaturesKey::fromStream ( std::istream & iIn ) [virtual]

Read a Business Object Key from an input stream.



## Parameters

<i>istream</i> &	the input stream.
------------------	-------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 41 of file [YieldFeaturesKey.cpp](#).

32.167.3.5 `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\(\)](#).

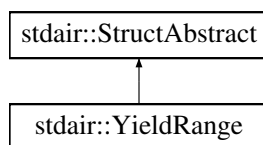
The documentation for this struct was generated from the following files:

- [stdair/bom/YieldFeaturesKey.hpp](#)
- [stdair/bom/YieldFeaturesKey.cpp](#)

## 32.168 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

### 32.168.1 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](#).

### 32.168.2 Constructor & Destructor Documentation

#### 32.168.2.1 stdair::YieldRange::YieldRange ( )

Constructors.

Definition at line 13 of file [YieldRange.cpp](#).

#### 32.168.2.2 stdair::YieldRange::YieldRange ( const YieldRange & iYieldRange )

Definition at line 20 of file [YieldRange.cpp](#).

#### 32.168.2.3 stdair::YieldRange::YieldRange ( const Yield\_T iUpperYield )

Definition at line 27 of file [YieldRange.cpp](#).

#### 32.168.2.4 stdair::YieldRange::YieldRange ( const Yield\_T iUpperYield, const Yield\_T iAverageYield )

Definition at line 33 of file [YieldRange.cpp](#).

#### 32.168.2.5 stdair::YieldRange::YieldRange ( const Yield\_T iUpperYield, const Yield\_T iAverageYield, const Yield\_T iLowerYield )

Definition at line 40 of file [YieldRange.cpp](#).

#### 32.168.2.6 stdair::YieldRange::~YieldRange ( ) [virtual]

Constructors.

Definition at line 48 of file [YieldRange.cpp](#).

### 32.168.3 Member Function Documentation

#### 32.168.3.1 Yield\_T stdair::YieldRange::getUpperYield ( ) const [inline]

Getter for the upper yield of the range.

Definition at line 39 of file [YieldRange.hpp](#).

#### 32.168.3.2 Yield\_T stdair::YieldRange::getAverageYield ( ) const [inline]

Getter for the average yield of the range.

Definition at line 43 of file [YieldRange.hpp](#).

#### 32.168.3.3 Yield\_T stdair::YieldRange::getLowerYield ( ) const [inline]

Getter for the lower yield of the range.

Definition at line 47 of file [YieldRange.hpp](#).

#### 32.168.3.4 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](#).

32.168.3.5 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](#).

32.168.3.6 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](#).

32.168.3.7 void stdair::YieldRange::toStream ( std::ostream & ioOut ) const

Dump a Business Object into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Definition at line 52 of file [YieldRange.cpp](#).

32.168.3.8 void stdair::YieldRange::fromStream ( std::istream & iIn ) [virtual]

Read a Business Object from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented from [stdair::StructAbstract](#).

Definition at line 58 of file [YieldRange.cpp](#).

32.168.3.9 const std::string stdair::YieldRange::describe ( ) const [virtual]

Display of the structure.

Implements [stdair::StructAbstract](#).

Definition at line 62 of file [YieldRange.cpp](#).

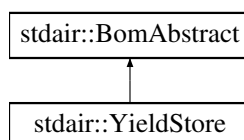
The documentation for this class was generated from the following files:

- [stdair/basic/YieldRange.hpp](#)
- [stdair/basic/YieldRange.cpp](#)

## 32.169 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**

- template<typename BOM >  
class [FacBom](#)
- class [FacBomManager](#)

**32.169.1 Detailed Description**

Class representing the actual attributes for an airline [YieldStore](#).

Definition at line 18 of file [YieldStore.hpp](#).

**32.169.2 Member Typedef Documentation****32.169.2.1 typedef YieldStoreKey stdair::YieldStore::Key\_T**

Definition allowing to retrieve the associated BOM key type.

Definition at line 25 of file [YieldStore.hpp](#).

**32.169.3 Constructor & Destructor Documentation****32.169.3.1 stdair::YieldStore::YieldStore ( const Key\_T & iKey ) [protected]**

Default constructors.

Definition at line 13 of file [YieldStore.cpp](#).

**32.169.3.2 stdair::YieldStore::YieldStore ( const YieldStore & ) [protected]****32.169.3.3 stdair::YieldStore::~~YieldStore ( ) [protected]**

Destructor.

Definition at line 17 of file [YieldStore.cpp](#).

## 32.169.4 Member Function Documentation

32.169.4.1 `void stdair::YieldStore::toStream ( std::ostream & ioOut ) const` `[inline]`,`[virtual]`

Dump a Business Object into an output stream.

## Parameters

<i>ostream</i> &	the output stream.
------------------	--------------------

Implements [stdair::BomAbstract](#).

Definition at line 31 of file [YieldStore.hpp](#).

References [toString\(\)](#).

32.169.4.2 `BomAbstract* const stdair::YieldStore::getParent ( ) const` `[inline]`

Get the parent object.

Definition at line 34 of file [YieldStore.hpp](#).

References [\\_parent](#).

32.169.4.3 `void stdair::YieldStore::fromStream ( std::istream & ioIn )` `[inline]`,`[virtual]`

Read a Business Object from an input stream.

## Parameters

<i>istream</i> &	the input stream.
------------------	-------------------

Implements [stdair::BomAbstract](#).

Definition at line 38 of file [YieldStore.hpp](#).

32.169.4.4 `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\(\)](#).

32.169.4.5 `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\(\)](#).

32.169.4.6 `const Key_T& stdair::YieldStore::getKey ( ) const` `[inline]`

Get the [YieldStore](#) key.

Definition at line 49 of file [YieldStore.hpp](#).

References [\\_key](#).

32.169.4.7 `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\(\)](#).

### 32.169.5 Friends And Related Function Documentation

**32.169.5.1** `template<typename BOM > friend class FacBom` [friend]

Definition at line 19 of file [YieldStore.hpp](#).

**32.169.5.2** `friend class FacBomManager` [friend]

Definition at line 20 of file [YieldStore.hpp](#).

### 32.169.6 Member Data Documentation

**32.169.6.1** `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\(\)](#).

**32.169.6.2** `BomAbstract* stdair::YieldStore::_parent` [protected]

Definition at line 67 of file [YieldStore.hpp](#).

Referenced by [getParent\(\)](#).

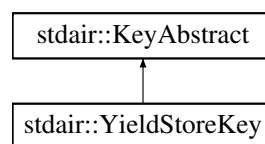
The documentation for this class was generated from the following files:

- [stdair/bom/YieldStore.hpp](#)
- [stdair/bom/YieldStore.cpp](#)

## 32.170 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

### 32.170.1 Detailed Description

Key of [YieldStore](#).

Definition at line 14 of file [YieldStoreKey.hpp](#).

### 32.170.2 Constructor & Destructor Documentation

#### 32.170.2.1 `stdair::YieldStoreKey::YieldStoreKey ( const AirlineCode_T & iAirlineCode )`

Constructors.

Definition at line 10 of file [YieldStoreKey.cpp](#).

#### 32.170.2.2 `stdair::YieldStoreKey::YieldStoreKey ( const YieldStoreKey & iKey )`

Definition at line 14 of file [YieldStoreKey.cpp](#).

#### 32.170.2.3 `stdair::YieldStoreKey::~YieldStoreKey ( )`

Destructor.

Definition at line 19 of file [YieldStoreKey.cpp](#).

### 32.170.3 Member Function Documentation

#### 32.170.3.1 `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\(\)](#).

#### 32.170.3.2 `void stdair::YieldStoreKey::toStream ( std::ostream & ioOut ) const` `[virtual]`

Dump a Business Object Key into an output stream.

Parameters

<i>ostream&amp;</i>	the output stream.
---------------------	--------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 23 of file [YieldStoreKey.cpp](#).

References [toString\(\)](#).

#### 32.170.3.3 `void stdair::YieldStoreKey::fromStream ( std::istream & ioIn )` `[virtual]`

Read a Business Object Key from an input stream.

Parameters

<i>istream&amp;</i>	the input stream.
---------------------	-------------------

Reimplemented from [stdair::KeyAbstract](#).

Definition at line 28 of file [YieldStoreKey.cpp](#).

#### 32.170.3.4 `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](#)

## 33 File Documentation

### 33.1 batches/stdair.cpp File Reference

### 33.2 stdair.cpp

```

00001
00005 // STL
00006 #include <cassert>
00007 #include <iostream>
00008 #include <sstream>
00009 #include <fstream>
00010 #include <string>
00011 // Boost (Extended STL)
00012 #include <boost/date_time/posix_time/posix_time.hpp>
00013 #include <boost/date_time/gregorian/gregorian.hpp>
00014 #include <boost/program_options.hpp>
00015 #include <boost/tokenizer.hpp>
00016 #include <boost/lexical_cast.hpp>
00017 // StdAir
00018 #include <stdair/stdair_types.hpp>
00019 #include <stdair/bom/BomArchive.hpp>
00020 #include <stdair/bom/BookingRequestStruct.hpp>
00021 #include <stdair/bom/TravelSolutionStruct.hpp>
00022 #include <stdair/service/Logger.hpp>
00023 #include <stdair/STDAIR_Service.hpp>
00024 #include <stdair/config/stdair-paths.hpp>
00025
00026 // ////////// Constants //////////
00030 const std::string K_STDAIR_DEFAULT_LOG_FILENAME ("stdair.log");
00031
00035 const std::string K_STDAIR_DEFAULT_INPUT_FILENAME (STDAIR_SAMPLE_DIR
00036                                                    "/schedule01.csv");
00037
00042 const bool K_STDAIR_DEFAULT_BUILT_IN_INPUT = false;
00043
00049 const bool K_STDAIR_DEFAULT_BUILT_FOR_RMOL = false;
00050
00056 const bool K_STDAIR_DEFAULT_BUILT_FOR_CRS = false;
00057
00062 const int K_STDAIR_EARLY_RETURN_STATUS = 99;
00063
00064 // ////////// Parsing of Options & Configuration //////////
00065 // A helper function to simplify the main part.
00066 template<class T> std::ostream& operator<< (std::ostream& os,
00067                                           const std::vector<T>& v) {
00068     std::copy (v.begin(), v.end(), std::ostream_iterator<T> (std::cout, " "));
00069     return os;
00070 }
00071
00073 int readConfiguration (int argc, char* argv[], bool& ioIsBuiltin,
00074                      bool& ioIsForRMOL, bool& ioIsForCRS,
00075                      stdair::Filename_T& ioInputFilename,
00076                      std::string& ioLogFilename) {
00077     // Default for the built-in input
00078     ioIsBuiltin = K_STDAIR_DEFAULT_BUILT_IN_INPUT;
00079
00080     // Default for the RMOL input
00081     ioIsForRMOL = K_STDAIR_DEFAULT_BUILT_FOR_RMOL;
00082
00083     // Default for the CRS input
00084     ioIsForCRS = K_STDAIR_DEFAULT_BUILT_FOR_CRS;
00085
00086     // Declare a group of options that will be allowed only on command line
00087     boost::program_options::options_description generic ("Generic options");

```



```

00088     generic.add_options()
00089         ("prefix", "print installation prefix")
00090         ("version,v", "print version string")
00091         ("help,h", "produce help message");
00092
00093     // Declare a group of options that will be allowed both on command
00094     // line and in config file
00095
00096     boost::program_options::options_description config ("Configuration");
00097     config.add_options()
00098         ("builtin,b",
00099          "The sample BOM tree can be either built-in or parsed from an input file. That latter must then be
00100          given with the -i/--input option")
00101         ("rmol,r",
00102          "Build a sample BOM tree for RMOL (i.e., a dummy flight-date with a single leg-cabin)")
00103         ("crs,c",
00104          "Build a sample BOM tree for CRS")
00105         ("input,i",
00106          boost::program_options::value< std::string >(&ioInputFilename)->default_value(
00107            K_STDAIR_DEFAULT_INPUT_FILENAME),
00108          "(CVS) input file for the demand distributions")
00109         ("log,l",
00110          boost::program_options::value< std::string >(&ioLogFilename)->default_value(
00111            K_STDAIR_DEFAULT_LOG_FILENAME),
00112          "Filename for the logs")
00113         ;
00114
00115     // Hidden options, will be allowed both on command line and
00116     // in config file, but will not be shown to the user.
00117     boost::program_options::options_description hidden ("Hidden options");
00118     hidden.add_options()
00119         ("copyright",
00120          boost::program_options::value< std::vector<std::string> >(),
00121          "Show the copyright (license)");
00122
00123     boost::program_options::options_description cmdline_options;
00124     cmdline_options.add(generic).add(config).add(hidden);
00125
00126     boost::program_options::options_description config_file_options;
00127     config_file_options.add(config).add(hidden);
00128     boost::program_options::options_description visible ("Allowed options");
00129     visible.add(generic).add(config);
00130
00131     boost::program_options::positional_options_description p;
00132     p.add ("copyright", -1);
00133
00134     boost::program_options::variables_map vm;
00135     boost::program_options::store (boost::program_options::command_line_parser (argc, argv).
00136                                   options (cmdline_options).positional(p).run(), vm);
00137
00138     std::ifstream ifs ("stdair.cfg");
00139     boost::program_options::store (parse_config_file (ifs, config_file_options),
00140                                   vm);
00141     boost::program_options::notify (vm);
00142
00143     if (vm.count ("help")) {
00144         std::cout << visible << std::endl;
00145         return K_STDAIR_EARLY_RETURN_STATUS;
00146     }
00147
00148     if (vm.count ("version")) {
00149         std::cout << PACKAGE_NAME << ", version " << PACKAGE_VERSION << std::endl;
00150         return K_STDAIR_EARLY_RETURN_STATUS;
00151     }
00152
00153     if (vm.count ("prefix")) {
00154         std::cout << "Installation prefix: " << PREFIXDIR << std::endl;
00155         return K_STDAIR_EARLY_RETURN_STATUS;
00156     }
00157
00158     if (vm.count ("builtin")) {
00159         ioIsBuiltin = true;
00160     }
00161
00162     if (vm.count ("rmol")) {
00163         ioIsForRMOL = true;
00164
00165         // The RMOL sample tree takes precedence over the default built-in BOM tree
00166         ioIsBuiltin = false;
00167     }
00168
00169     if (vm.count ("crs")) {
00170         ioIsForCRS = true;
00171
00172         // The RMOL sample tree takes precedence over the default built-in BOM tree
00173         ioIsBuiltin = false;

```

```

00172     }
00173
00174     const std::string isBuiltinStr = (ioIsBuiltin == true)?"yes":"no";
00175     std::cout << "The BOM should be built-in? " << isBuiltinStr << std::endl;
00176
00177     const std::string isForRMOLStr = (ioIsForRMOL == true)?"yes":"no";
00178     std::cout << "The BOM should be built-in for RMOL? " << isForRMOLStr
00179         << std::endl;
00180
00181     const std::string isForCRSStr = (ioIsForCRS == true)?"yes":"no";
00182     std::cout << "The BOM should be built-in for CRS? " << isForCRSStr
00183         << std::endl;
00184
00185     if (ioIsBuiltin == false && ioIsForRMOL == false && ioIsForCRS == false) {
00186         if (vm.count ("input")) {
00187             ioInputFilename = vm["input"].as< std::string >();
00188             std::cout << "Input filename is: " << ioInputFilename << std::endl;
00189         } else {
00190             std::cerr << "Either one among the -b/--builtin, -r/--rmol, -c/--crs "
00191                 << "or -i/--input options must be specified" << std::endl;
00192         }
00193     }
00194 }
00195
00196 if (vm.count ("log")) {
00197     ioLogFilename = vm["log"].as< std::string >();
00198     std::cout << "Log filename is: " << ioLogFilename << std::endl;
00199 }
00200
00201 return 0;
00202 }
00203
00204
00205 // ////////////////////////////////// M A I N //////////////////////////////////
00206 int main (int argc, char* argv[]) {
00207
00208     // State whether the BOM tree should be built-in or parsed from an
00209     // input file
00210     bool isBuiltin;
00211
00212     // State whether a sample BOM tree should be built for RMOL.
00213     bool isForRMOL;
00214
00215     // State whether a sample BOM tree should be built for the CRS.
00216     bool isForCRS;
00217
00218     // Input file name
00219     stdair::Filename_T lInputFilename;
00220
00221     // Output log File
00222     std::string lLogFilename;
00223
00224     // Call the command-line option parser
00225     const int lOptionParserStatus =
00226         readConfiguration (argc, argv, isBuiltin, isForRMOL, isForCRS,
00227             lInputFilename, lLogFilename);
00228
00229     if (lOptionParserStatus == K_STDAIR_EARLY_RETURN_STATUS) {
00230         return 0;
00231     }
00232
00233     // Set the log parameters
00234     std::ofstream logOutputFile;
00235     // Open and clean the log outputfile
00236     logOutputFile.open (lLogFilename.c_str());
00237     logOutputFile.clear();
00238
00239     const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG,
00240         logOutputFile);
00241     stdair::STDAIR_Service stdairService (lLogParams);
00242
00243     // DEBUG
00244     STDAIR_LOG_DEBUG ("Welcome to stdair");
00245
00246     // Check whether or not a (CSV) input file should be read
00247     if (isBuiltin == true || isForRMOL == true || isForCRS == true) {
00248         if (isForRMOL == true) {
00249             // Build the sample BOM tree for RMOL
00250             stdairService.buildDummyInventory (300);
00251         } else if (isForCRS == true) {
00252             //
00253             stdair::TravelSolutionList_T lTravelSolutionList;
00254             stdairService.buildSampleTravelSolutions (lTravelSolutionList);
00255
00256             // Build the sample BOM tree for CRS
00257

```

```

00258     const stdair::BookingRequestStruct& lBookingRequest =
00259         stdairService.buildSampleBookingRequest();
00260
00261     // DEBUG: Display the travel solution and booking request
00262     STDAIR_LOG_DEBUG ("Booking request: " << lBookingRequest.
display());
00263
00264     const std::string& lCSVDump =
00265         stdairService.csvDisplay (lTravelSolutionList);
00266     STDAIR_LOG_DEBUG (lCSVDump);
00267
00268     } else {
00269         assert (isBuiltin == true);
00270
00271         // Build a sample BOM tree
00272         stdairService.buildSampleBom();
00273     }
00274
00275     } else {
00276         // Read the input file
00277         //stdairService.readFromInputFile (lInputFilename);
00278
00279         // DEBUG
00280         STDAIR_LOG_DEBUG ("StdAir will parse " << lInputFilename
00281             << " and build the corresponding BOM tree.");
00282     }
00283
00284     // DEBUG: Display the whole persistent BOM tree
00285     const std::string& lCSVDump = stdairService.csvDisplay ();
00286     STDAIR_LOG_DEBUG (lCSVDump);
00287
00288     // Close the Log outputFile
00289     logOutputFile.close();
00290
00291     /*
00292     Note: as that program is not intended to be run on a server in
00293     production, it is better not to catch the exceptions. When it
00294     happens (that an exception is throwned), that way we get the
00295     call stack.
00296     */
00297
00298     return 0;
00299 }
00300

```

### 33.3 doc/local/authors.doc File Reference

### 33.4 doc/local/codingrules.doc File Reference

### 33.5 doc/local/copyright.doc File Reference

### 33.6 doc/local/documentation.doc File Reference

### 33.7 doc/local/features.doc File Reference

### 33.8 doc/local/help\_wanted.doc File Reference

### 33.9 doc/local/howto\_release.doc File Reference

### 33.10 doc/local/index.doc File Reference

### 33.11 doc/local/installation.doc File Reference

### 33.12 doc/local/linking.doc File Reference

### 33.13 doc/local/test.doc File Reference

### 33.14 doc/local/users\_guide.doc File Reference

## 33.15 doc/local/verification.doc File Reference

## 33.16 doc/tutorial/tutorial.doc File Reference

## 33.17 stdair/basic/BasChronometer.cpp File Reference

```
#include <cassert>
#include <stdair/basic/BasChronometer.hpp>
```

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.18 BasChronometer.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // Stdair
00007 #include <stdair/basic/BasChronometer.hpp>
00008
00009 namespace stdair {
00010
00011 // //////////////////////////////////////
00012 BasChronometer::BasChronometer () : _startTimeLaunched (false) {
00013 }
00014
00015 // //////////////////////////////////////
00016 void BasChronometer::start () {
00017     // Get the time-stamp of now, and store it for later use
00018     _startTime = boost::posix_time::microsec_clock::local_time();
00019
00020     // Update the boolean which states whether the chronometer
00021     // is launched
00022     _startTimeLaunched = true;
00023 }
00024
00025 // //////////////////////////////////////
00026 double BasChronometer::elapsed () const {
00027     assert (_startTimeLaunched == true);
00028
00029     // Get the time-stamp of now
00030     const boost::posix_time::ptime lStopTime =
00031         boost::posix_time::microsec_clock::local_time();
00032
00033     // Calculate the time elapsed since the last time-stamp
00034     const boost::posix_time::time_duration lElapsedTime =
00035         lStopTime - _startTime;
00036
00037     // Derived the corresponding number of milliseconds
00038     const double lElapsedTimeInMicroSeconds =
00039         static_cast<const double> (lElapsedTime.total_microseconds());
00040
00041     // The elapsed time given in return is expressed in seconds
00042     return (lElapsedTimeInMicroSeconds / 1e6);
00043 }
00044
00045 }
```

## 33.19 stdair/basic/BasChronometer.hpp File Reference

```
#include <boost/date_time/posix_time/posix_time.hpp>
```

## Classes

- struct [stdair::BasChronometer](#)

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.20 BasChronometer.hpp

```

00001 #ifndef __STDAIR_BAS_BASCHRONOMETER_HPP
00002 #define __STDAIR_BAS_BASCHRONOMETER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // Boost (STL Extension)
00008 // Boost Date-Time (http://boost.org/doc/html/date_time/posix_time.html)
00009 #include <boost/date_time/posix_time/posix_time.hpp>
00010
00011 namespace stdair {
00012
00013     struct BasChronometer {
00014         BasChronometer();
00015
00016         void start ();
00017
00018         std::string getStart () const {
00019             return boost::posix_time::to_simple_string (_startTime);
00020         }
00021
00022         double elapsed () const;
00023
00024     private:
00025         boost::posix_time::ptime _startTime;
00026         bool _startTimeLaunched;
00027     };
00028
00029 }
00030 #endif // __STDAIR_BAS_BASCHRONOMETER_HPP

```

### 33.21 stdair/basic/BasConst.cpp File Reference

```

#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/basic/BasConst_Event.hpp>
#include <stdair/basic/BasConst_Request.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_BookingClass.hpp>
#include <stdair/basic/BasConst_Yield.hpp>
#include <stdair/basic/BasConst_DefaultObject.hpp>
#include <stdair/basic/BasConst_Period_BOM.hpp>
#include <stdair/basic/BasConst_TravelSolution.hpp>
#include <stdair/basic/BasConst_SellUpCurves.hpp>

```

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## Functions

- const std::string [stdair::DEFAULT\\_BOM\\_ROOT\\_KEY](#) (" -- ROOT -- ")
- const double [stdair::DEFAULT\\_EPSILON\\_VALUE](#) (0.0001)
- const unsigned int [stdair::DEFAULT\\_FLIGHT\\_SPEED](#) (900)
- const NbOfFlightDates\_T [stdair::DEFAULT\\_NB\\_OF\\_FLIGHTDATES](#) (0.0)
- const Duration\_T [stdair::NULL\\_BOOST\\_TIME\\_DURATION](#) (-1,-1,-1)
- const 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\\_NBFAIRLINES](#) (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::HOURL\\_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 DTDFrattMap\_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]

### 33.22 BasConst.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // StdAir
00005 #include <stdair/basic/BasConst_General.hpp>
00006 #include <stdair/basic/BasConst_BomDisplay.hpp>
00007 #include <stdair/basic/BasConst_Event.hpp>
00008 #include <stdair/basic/BasConst_Request.hpp>
00009 #include <stdair/basic/BasConst_Inventory.hpp>
00010 #include <stdair/basic/BasConst_BookingClass.hpp>
00011 #include <stdair/basic/BasConst_Yield.hpp>
00012 #include <stdair/basic/BasConst_DefaultObject.hpp>
00013 #include <stdair/basic/BasConst_Period_BOM.hpp>
00014 #include <stdair/basic/BasConst_TravelSolution.hpp>
00015 #include <stdair/basic/BasConst_SellUpCurves.hpp>
00016
00017 namespace stdair {
00018
00019     // ////////// General //////////
00021     const std::string DEFAULT_BOM_ROOT_KEY (" -- ROOT -- ");
00022
00024     const double DEFAULT_EPSILON_VALUE (0.0001);
00025
00027     const unsigned int DEFAULT_FLIGHT_SPEED (900);
00028
00030     const NbOfFlightDates_T DEFAULT_NB_OF_FLIGHTDATES (0.0);
00031
00033     const Duration_T NULL_BOOST_TIME_DURATION (-1, -1, -1);
00034
00036     const Duration_T DEFAULT_NULL_DURATION (0, 0, 0);
00037
00039     const unsigned int DEFAULT_NB_OF_DAYS_IN_A_YEAR (365);
00040
00042     const unsigned int DEFAULT_NUMBER_OF_SUBDIVISIONS (1000);
00043
00044     // ////////// (Flight-)Period-related BOM //////////
00046     const DayDuration_T DEFAULT_DAY_DURATION (0);
00047
00049     const DatePeriod_T BOOST_DEFAULT_DATE_PERIOD (
Date_T (2007, 1, 1),
Date_T (2007, 1, 1));
00050
00053     const std::string DOW_STR[] =
00054         {"Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"};
00055
00057     const DOW_String_T DEFAULT_DOW_STRING ("0000000");
00058
00060     const DateOffset_T DEFAULT_DATE_OFFSET (0);
00061
00062
00063     // // ////////// General //////////
00065     const Date_T DEFAULT_DATE (2010, boost::gregorian::Jan, 1);
00066
00068     const DateTime_T DEFAULT_DATETIME (DEFAULT_DATE,

```

```

        NULL_BOOST_TIME_DURATION);
00069
00071     const Duration_T DEFAULT_EPSILON_DURATION (0, 0, 0, 1);
00072
00074     const Count_T SECONDS_IN_ONE_DAY (86400);
00075
00077     const Count_T MILLISECONDS_IN_ONE_SECOND (1000);
00078
00080     const RandomSeed_T DEFAULT_RANDOM_SEED (120765987);
00081
00082     // ////////// Default BOM tree objects //////////
00084     const AirportCode_T AIRPORT_LHR ("LHR");
00085
00087     const AirportCode_T AIRPORT_SYD ("SYD");
00088
00090     const CityCode_T POS_LHR ("LHR");
00091
00093     const Date_T DATE_20110115 (2011, boost::gregorian::Jan, 15);
00094     const Date_T DATE_20111231 (2011, boost::gregorian::Dec, 31);
00095
00097     const DayDuration_T NO_ADVANCE_PURCHASE (0);
00098
00100     const SaturdayStay_T SATURDAY_STAY (true);
00101
00103     const SaturdayStay_T NO_SATURDAY_STAY (false);
00104
00106     const ChangeFees_T CHANGE_FEES (true);
00107
00109     const ChangeFees_T NO_CHANGE_FEES (false);
00110
00112     const NonRefundable_T NON_REFUNDABLE (true);
00113
00115     const NonRefundable_T NO_NON_REFUNDABLE (false);
00116
00118     const SaturdayStay_T DEFAULT_BOM_TREE_SATURDAY_STAY (true);
00119
00121     const ChangeFees_T DEFAULT_BOM_TREE_CHANGE_FEES (true);
00122
00124     const NonRefundable_T DEFAULT_BOM_TREE_NON_REFUNDABLE (true
);
00125
00127     const DayDuration_T NO_STAY_DURATION (0);
00128
00130     const AirlineCode_T AIRLINE_CODE_BA ("BA");
00131
00133     const CabinCode_T CABIN_Y ("Y");
00134
00136     const ClassCode_T CLASS_CODE_Y ("Y");
00137
00138     // ////////// Travel solutions related objects //////////
00140     const ClassCode_T CLASS_CODE_Q ("Q");
00141
00142     // ////////// Booking request related objects //////////
00144     const AirportCode_T AIRPORT_SIN ("SIN");
00145
00147     const AirportCode_T AIRPORT_BKK ("BKK");
00148
00150     const CityCode_T POS_SIN ("SIN");
00151
00153     const CabinCode_T CABIN_ECO ("Eco");
00154
00156     const FrequentFlyer_T FREQUENT_FLYER_MEMBER ("M");
00157
00158     // ////////// Default //////////
00160     const FamilyCode_T DEFAULT_FAMILY_CODE ("0");
00161
00163     const PolicyCode_T DEFAULT_POLICY_CODE ("0");
00164
00166     const NestingStructureCode_T
DEFAULT_NESTING_STRUCTURE_CODE ("DEFAULT");
00167
00169     const NestingStructureCode_T
DISPLAY_NESTING_STRUCTURE_CODE ("Display Nesting");
00170
00172     const NestingStructureCode_T
YIELD_BASED_NESTING_STRUCTURE_CODE ("Yield-Based Nesting");
00173
00175     const NestingNodeCode_T DEFAULT_NESTING_NODE_CODE ("0");
00176
00178     const NbOfAirlines_T DEFAULT_NBFAIRLINES (0);
00179
00181     const FlightPathCode_T DEFAULT_FLIGHTPATH_CODE ("");
00182
00183     // ////////// Booking-class-related BOM //////////
00185     const Distance_T DEFAULT_DISTANCE_VALUE (0);
00186
00188     const ClassCode_T DEFAULT_CLOSED_CLASS_CODE ("CC");

```

```

00189
00192     const NbOfBookings_T DEFAULT_CLASS_NB_OF_BOOKINGS (0);
00193
00196     const NbOfBookings_T DEFAULT_CLASS_TOTAL_NB_OF_BOOKINGS (
00197         0);
00199     const NbOfBookings_T DEFAULT_CLASS_UNCONSTRAINED_DEMAND (
00200         0);
00202     const NbOfBookings_T DEFAULT_CLASS_REMAINING_DEMAND_MEAN
00203         (0);
00205     const NbOfBookings_T
00206     DEFAULT_CLASS_REMAINING_DEMAND_STANDARD_DEVIATION (0);
00208     const NbOfCancellations_T DEFAULT_CLASS_NB_OF_CANCELLATIONS
00209         (0);
00211     const NbOfNoShows_T DEFAULT_CLASS_NB_OF_NOSHOWS (0);
00212
00214     const CabinCapacity_T DEFAULT_CABIN_CAPACITY (100.0);
00215
00217     const CommittedSpace_T DEFAULT_COMMITTED_SPACE (0.0);
00218
00220     const BlockSpace_T DEFAULT_BLOCK_SPACE (0.0);
00221
00223     const Availability_T DEFAULT_NULL_AVAILABILITY (0.0);
00224
00226     const Availability_T DEFAULT_AVAILABILITY (9.0);
00227
00229     const Availability_T MAXIMAL_AVAILABILITY (9999.0);
00230
00232     const UnconstrainingMethod DEFAULT_UNCONSTRAINING_METHOD ('E');
00233
00235     const PartnershipTechnique DEFAULT_PARTNERSHIP_TECHNIQUE ('N');
00236
00238     const ForecastingMethod DEFAULT_FORECASTING_METHOD ('Q');
00239
00241     const PreOptimisationMethod DEFAULT_PREOPTIMISATION_METHOD ('N');
00242
00244     const OptimisationMethod DEFAULT_OPTIMISATION_METHOD ('M');
00245
00246     // ////////// (Segment-)Class-related BOM //////////
00249     const CensorshipFlag_T DEFAULT_CLASS_CENSORSHIPFLAG (false);
00250
00253     const CensorshipFlagList_T
00254     DEFAULT_CLASS_CENSORSHIPFLAG_LIST =
00255         std::vector<CensorshipFlag_T>();
00257     const BookingLimit_T DEFAULT_CLASS_BOOKING_LIMIT (9999.0);
00258
00260     const AuthorizationLevel_T
00261     DEFAULT_CLASS_AUTHORIZATION_LEVEL (9999.0);
00263     const AuthorizationLevel_T
00264     DEFAULT_CLASS_MAX_AUTHORIZATION_LEVEL (9999.0);
00266     const AuthorizationLevel_T
00267     DEFAULT_CLASS_MIN_AUTHORIZATION_LEVEL (0.0);
00269     const OverbookingRate_T DEFAULT_CLASS_OVERBOOKING_RATE (0.
00270         0);
00272     const BookingRatio_T DEFAULT_OND_BOOKING_RATE (0.0);
00273
00275     const Fare_T DEFAULT_FARE_VALUE (0.0);
00276
00278     const Yield_T DEFAULT_CLASS_YIELD_VALUE (0.0);
00279
00281     const Revenue_T DEFAULT_REVENUE_VALUE (0.0);
00282
00284     const Percentage_T DEFAULT_LOAD_FACTOR_VALUE (100.0);
00285
00286     // ////////// (Leg-)YieldRange-related BOM //////////
00289     const Yield_T DEFAULT_YIELD_VALUE (0.0);
00290
00292     const Yield_T DEFAULT_YIELD_MAX_VALUE (std::numeric_limits<double>::max());
00293
00295     const NbOfBookings_T DEFAULT_YIELD_NB_OF_BOOKINGS (0.0);
00296
00298     const Identity_T DEFAULT_BOOKING_NUMBER (0);
00299
00301     const NbOfCancellations_T DEFAULT_YIELD_NB_OF_CANCELLATIONS
00302         (0.0);
00304     const NbOfNoShows_T DEFAULT_YIELD_NB_OF_NOSHOWS (0.0);

```

```

00305
00307     const Availability_T DEFAULT_YIELD_AVAILABILITY (0.0);
00308
00311     const CensorshipFlag_T DEFAULT_YIELD_CENSORSHIPFLAG (false);
00312
00314     const BookingLimit_T DEFAULT_YIELD_BOOKING_LIMIT (0.0);
00315
00317     const OverbookingRate_T DEFAULT_YIELD_OVERBOOKING_RATE (0.
0);
00318
00319
00320     // ////////// OnD-related BOM //////////
00322     const Fare_T DEFAULT_OND_FARE_VALUE (0.0);
00323
00324
00325     // ////////// Event Generation //////////
00326
00328     const Count_T DEFAULT_PROGRESS_STATUS (0);
00329
00331     const Percentage_T MAXIMUM_PROGRESS_STATUS (100);
00332
00335     const Date_T DEFAULT_EVENT_OLDEST_DATE (2008, boost::gregorian::Jan, 1);
00336
00339     const DateTime_T DEFAULT_EVENT_OLDEST_DATETIME (
DEFAULT_EVENT_OLDEST_DATE,
                                NULL_BOOST_TIME_DURATION);
00340
00341
00342
00343     // ////////// Booking Request //////////
00345     const PartySize_T DEFAULT_PARTY_SIZE (1);
00346
00348     const DayDuration_T DEFAULT_STAY_DURATION (7);
00349
00351     const WTP_T DEFAULT_WTP (1000.0);
00352
00354     const Date_T DEFAULT_PREFERRED_DEPARTURE_DATE (
DEFAULT_DEPARTURE_DATE);
00355
00357     const Duration_T DEFAULT_PREFERRED_DEPARTURE_TIME (8, 0, 0);
00358
00360     const DateOffset_T DEFAULT_ADVANCE_PURCHASE (22);
00361
00363     const Date_T DEFAULT_REQUEST_DATE (
DEFAULT_PREFERRED_DEPARTURE_DATE
                                - DEFAULT_ADVANCE_PURCHASE);
00364
00365
00367     const Duration_T DEFAULT_REQUEST_TIME (8, 0, 0);
00368
00370     const DateTime_T DEFAULT_REQUEST_DATE_TIME (
DEFAULT_REQUEST_DATE,
                                DEFAULT_REQUEST_TIME);
00371
00372
00374     const CabinCode_T DEFAULT_PREFERRED_CABIN ("M");
00375
00377     const CityCode_T DEFAULT_POS ("ALL");
00378
00380     const ChannelLabel_T DEFAULT_CHANNEL ("DC");
00381
00383     const ChannelLabel_T CHANNEL_DN ("DN");
00384
00386     const ChannelLabel_T CHANNEL_IN ("IN");
00387
00389     const TripType_T TRIP_TYPE_ONE_WAY ("OW");
00390
00392     const TripType_T TRIP_TYPE_ROUND_TRIP ("RT");
00393
00395     const TripType_T TRIP_TYPE_INBOUND ("RI");
00396
00398     const TripType_T TRIP_TYPE_OUTBOUND ("RO");
00399
00401     const FrequentFlyer_T DEFAULT_FF_TIER ("N");
00402
00404     const PriceValue_T DEFAULT_VALUE_OF_TIME (100.0);
00405
00407     const IntDuration_T HOUR_CONVERTED_IN_SECONDS (3600);
00408
00409     // ////////// Travel Solutions //////////
00411     const Duration_T DEFAULT_MINIMAL_CONNECTION_TIME (0, 30, 0);
00412
00414     const Duration_T DEFAULT_MAXIMAL_CONNECTION_TIME (24, 0, 0);
00415
00417     const MatchingIndicator_T DEFAULT_MATCHING_INDICATOR (0.0);
00418
00420     const PriceCurrency_T DEFAULT_CURRENCY ("EUR");
00421
00423     const AvailabilityStatus_T DEFAULT_AVAILABILITY_STATUS (
false);

```

```

00424
00426     const Date_T DEFAULT_DICO_STUDIED_DATE;
00427
00428     // ////////// Inventory-related BOM //////////
00430     const AirlineCode_T DEFAULT_AIRLINE_CODE ("XX");
00431
00433     const AirlineCode_T DEFAULT_NULL_AIRLINE_CODE ("");
00434
00436     const AirlineCodeList_T DEFAULT_AIRLINE_CODE_LIST;
00437
00439     const FlightNumber_T DEFAULT_FLIGHT_NUMBER (9999);
00440
00442     const FlightNumber_T DEFAULT_FLIGHT_NUMBER_FF (255);
00443
00445     const TableID_T DEFAULT_TABLE_ID (9999);
00446
00448     const Date_T DEFAULT_DEPARTURE_DATE (1900, boost::gregorian::Jan, 1);
00449
00451     const AirportCode_T DEFAULT_AIRPORT_CODE ("XXX");
00452
00454     const AirportCode_T DEFAULT_NULL_AIRPORT_CODE ("");
00455
00457     const AirportCode_T DEFAULT_ORIGIN ("XXX");
00458
00460     const AirportCode_T DEFAULT_DESTINATION ("YYY");
00461
00463     const CabinCode_T DEFAULT_CABIN_CODE ("X");
00464
00466     const FamilyCode_T DEFAULT_FARE_FAMILY_CODE ("EcoSaver");
00467
00469     const FamilyCode_T DEFAULT_NULL_FARE_FAMILY_CODE ("NoFF");
00470
00472     const ClassCode_T DEFAULT_CLASS_CODE ("X");
00473
00475     const ClassCode_T DEFAULT_NULL_CLASS_CODE ("");
00476
00478     const ClassList_StringList_T DEFAULT_CLASS_CODE_LIST;
00479
00481     const BidPrice_T DEFAULT_BID_PRICE (0.0);
00482
00484     const BidPriceVector_T DEFAULT_BID_PRICE_VECTOR =
std::vector<BidPrice_T>();
00485
00489     const unsigned short MAXIMAL_NUMBER_OF_LEGS_IN_FLIGHT (7);
00490
00493     const unsigned short MAXIMAL_NUMBER_OF_SEGMENTS_IN_OND (3);
00494
00496     const SeatIndex_T DEFAULT_SEAT_INDEX (1);
00497
00499     const NbOfSeats_T DEFAULT_NULL_BOOKING_NUMBER (0);
00500
00502     const CapacityAdjustment_T
DEFAULT_NULL_CAPACITY_ADJUSTMENT (0);
00503
00505     const UPR_T DEFAULT_NULL_UPR (0);
00506
00508     const std::string DEFAULT_FARE_FAMILY_VALUE_TYPE ("FF");
00509
00511     const std::string DEFAULT_SEGMENT_CABIN_VALUE_TYPE ("SC");
00512
00514     const int DEFAULT_MAX_DTD = 365;
00515
00517     const DCPList_T DEFAULT_DCP_LIST =
DefaultDCPList::init();
00518     DCPList_T DefaultDCPList::init() {
00519         DCPList_T oDCPList;
00520         //oDCPList.push_back (72);
00521         oDCPList.push_back (63);oDCPList.push_back (56);oDCPList.push_back (49);
00522         oDCPList.push_back (42);oDCPList.push_back (35);oDCPList.push_back (31);
00523         oDCPList.push_back (27);oDCPList.push_back (23);oDCPList.push_back (19);
00524         oDCPList.push_back (16);oDCPList.push_back (13);oDCPList.push_back (10);
00525         oDCPList.push_back (7); oDCPList.push_back (5); oDCPList.push_back (3);
00526         oDCPList.push_back (1); oDCPList.push_back (0);
00527         return oDCPList;
00528     }
00529
00531     const FRAT5Curve_T FRAT5_CURVE_A =
00532         DefaultMap::createFRAT5CurveA();
00533     FRAT5Curve_T DefaultMap::createFRAT5CurveA() {
00534         FRAT5Curve_T oCurve;
00535         oCurve[63] = 1.05; oCurve[56] = 1.07; oCurve[49] = 1.09;
00536         oCurve[42] = 1.11; oCurve[35] = 1.14; oCurve[31] = 1.16;
00537         oCurve[27] = 1.18; oCurve[23] = 1.21; oCurve[19] = 1.24;
00538         oCurve[16] = 1.27; oCurve[13] = 1.30; oCurve[10] = 1.33;
00539         oCurve[7] = 1.37; oCurve[5] = 1.40; oCurve[3] = 1.45;
00540         oCurve[1] = 1.50;
00541         return oCurve;

```

```

00542     }
00543
00545     const FRAT5Curve_T FRAT5_CURVE_B =
00546         DefaultMap::createFRAT5CurveB();
00547     FRAT5Curve_T DefaultMap::createFRAT5CurveB() {
00548         FRAT5Curve_T oCurve;
00549         oCurve[63] = 1.20; oCurve[56] = 1.23; oCurve[49] = 1.26;
00550         oCurve[42] = 1.30; oCurve[35] = 1.35; oCurve[31] = 1.40;
00551         oCurve[27] = 1.50; oCurve[23] = 1.60; oCurve[19] = 1.80;
00552         oCurve[16] = 2.10; oCurve[13] = 2.20; oCurve[10] = 2.30;
00553         oCurve[7] = 2.40; oCurve[5] = 2.44; oCurve[3] = 2.47;
00554         oCurve[1] = 2.50;
00555         return oCurve;
00556     }
00557
00559     const FRAT5Curve_T FRAT5_CURVE_C =
00560         DefaultMap::createFRAT5CurveC();
00561     FRAT5Curve_T DefaultMap::createFRAT5CurveC() {
00562         FRAT5Curve_T oCurve;
00563         oCurve[63] = 1.40; oCurve[56] = 1.45; oCurve[49] = 1.50;
00564         oCurve[42] = 1.55; oCurve[35] = 1.60; oCurve[31] = 1.70;
00565         oCurve[27] = 1.80; oCurve[23] = 2.00; oCurve[19] = 2.30;
00566         oCurve[16] = 2.60; oCurve[13] = 3.00; oCurve[10] = 3.30;
00567         oCurve[7] = 3.40; oCurve[5] = 3.44; oCurve[3] = 3.47;
00568         oCurve[1] = 3.50;
00569         return oCurve;
00570     }
00571
00573     const FRAT5Curve_T FRAT5_CURVE_D =
00574         DefaultMap::createFRAT5CurveD();
00575     FRAT5Curve_T DefaultMap::createFRAT5CurveD() {
00576         FRAT5Curve_T oCurve;
00577         oCurve[63] = 1.60; oCurve[56] = 1.67; oCurve[49] = 1.74;
00578         oCurve[42] = 1.81; oCurve[35] = 1.88; oCurve[31] = 2.00;
00579         oCurve[27] = 2.15; oCurve[23] = 2.45; oCurve[19] = 2.75;
00580         oCurve[16] = 3.20; oCurve[13] = 3.80; oCurve[10] = 4.25;
00581         oCurve[7] = 4.35; oCurve[5] = 4.40; oCurve[3] = 4.45;
00582         oCurve[1] = 4.50;
00583         return oCurve;
00584     }
00585
00591     const FFDisutilityCurve_T FF_DISUTILITY_CURVE_A =
00592         DefaultMap::createFFDisutilityCurveA();
00593     FFDisutilityCurve_T DefaultMap::createFFDisutilityCurveA
00594     () {
00595         FFDisutilityCurve_T oCurve;
00596         oCurve[63] = 0.0098; oCurve[56] = 0.0096; oCurve[49] = 0.0093;
00597         oCurve[42] = 0.0090; oCurve[35] = 0.0086; oCurve[31] = 0.0082;
00598         oCurve[27] = 0.0077; oCurve[23] = 0.0071; oCurve[19] = 0.0065;
00599         oCurve[16] = 0.0059; oCurve[13] = 0.0052; oCurve[10] = 0.0045;
00600         oCurve[7] = 0.0039; oCurve[5] = 0.0036; oCurve[3] = 0.0033;
00601         oCurve[1] = 0.0030;
00602         return oCurve;
00603     }
00609     const FFDisutilityCurve_T FF_DISUTILITY_CURVE_B =
00610         DefaultMap::createFFDisutilityCurveB();
00611     FFDisutilityCurve_T DefaultMap::createFFDisutilityCurveB
00612     () {
00613         FFDisutilityCurve_T oCurve;
00614         oCurve[63] = 0.0082; oCurve[56] = 0.0080; oCurve[49] = 0.0078;
00615         oCurve[42] = 0.0075; oCurve[35] = 0.0072; oCurve[31] = 0.0068;
00616         oCurve[27] = 0.0064; oCurve[23] = 0.0059; oCurve[19] = 0.0054;
00617         oCurve[16] = 0.0049; oCurve[13] = 0.0044; oCurve[10] = 0.0038;
00618         oCurve[7] = 0.0033; oCurve[5] = 0.0030; oCurve[3] = 0.0028;
00619         oCurve[1] = 0.0025;
00620         return oCurve;
00621     }
00627     const FFDisutilityCurve_T FF_DISUTILITY_CURVE_C =
00628         DefaultMap::createFFDisutilityCurveC();
00629     FFDisutilityCurve_T DefaultMap::createFFDisutilityCurveC
00630     () {
00631         FFDisutilityCurve_T oCurve;
00632         oCurve[63] = 0.0065; oCurve[56] = 0.0064; oCurve[49] = 0.0062;
00633         oCurve[42] = 0.0060; oCurve[35] = 0.0057; oCurve[31] = 0.0054;
00634         oCurve[27] = 0.0051; oCurve[23] = 0.0047; oCurve[19] = 0.0043;
00635         oCurve[16] = 0.0039; oCurve[13] = 0.0035; oCurve[10] = 0.0030;
00636         oCurve[7] = 0.0026; oCurve[5] = 0.0024; oCurve[3] = 0.0022;
00637         oCurve[1] = 0.0020;
00638         return oCurve;
00639     }
00645     const FFDisutilityCurve_T FF_DISUTILITY_CURVE_D =
00646         DefaultMap::createFFDisutilityCurveD();
00647     FFDisutilityCurve_T DefaultMap::createFFDisutilityCurveD
00648     () {

```

```

00648     FFDisutilityCurve_T oCurve;
00649     oCurve[63] = 0.0050; oCurve[56] = 0.0049; oCurve[49] = 0.0047;
00650     oCurve[42] = 0.0045; oCurve[35] = 0.0043; oCurve[31] = 0.0040;
00651     oCurve[27] = 0.0037; oCurve[23] = 0.0034; oCurve[19] = 0.0030;
00652     oCurve[16] = 0.0026; oCurve[13] = 0.0022; oCurve[10] = 0.0017;
00653     oCurve[7] = 0.0013; oCurve[5] = 0.0012; oCurve[3] = 0.0011;
00654     oCurve[1] = 0.0010;
00655     return oCurve;
00656 }
00657
00663 const FFDisutilityCurve_T FF_DISUTILITY_CURVE_E =
00664     DefaultMap::createFFDisutilityCurveE();
00665 FFDisutilityCurve_T DefaultMap::createFFDisutilityCurveE
00666 () {
00667     FFDisutilityCurve_T oCurve;
00668     oCurve[63] = 0.0043; oCurve[56] = 0.0042; oCurve[49] = 0.0041;
00669     oCurve[42] = 0.0039; oCurve[35] = 0.0037; oCurve[31] = 0.0035;
00670     oCurve[27] = 0.0032; oCurve[23] = 0.0029; oCurve[19] = 0.0025;
00671     oCurve[16] = 0.0021; oCurve[13] = 0.0018; oCurve[10] = 0.0013;
00672     oCurve[7] = 0.0011; oCurve[5] = 0.0010; oCurve[3] = 0.0009;
00673     oCurve[1] = 0.0008;
00674     return oCurve;
00675 }
00681 const FFDisutilityCurve_T FF_DISUTILITY_CURVE_F =
00682     DefaultMap::createFFDisutilityCurveF();
00683 FFDisutilityCurve_T DefaultMap::createFFDisutilityCurveF
00684 () {
00685     FFDisutilityCurve_T oCurve;
00686     oCurve[63] = 0.0032; oCurve[56] = 0.0031; oCurve[49] = 0.0030;
00687     oCurve[42] = 0.0029; oCurve[35] = 0.0027; oCurve[31] = 0.0025;
00688     oCurve[27] = 0.0022; oCurve[23] = 0.0019; oCurve[19] = 0.0016;
00689     oCurve[16] = 0.0013; oCurve[13] = 0.0010; oCurve[10] = 0.0008;
00690     oCurve[7] = 0.0007; oCurve[5] = 0.0006; oCurve[3] = 0.0005;
00691     oCurve[1] = 0.0004;
00692     return oCurve;
00693 }
00695 const DTDFractMap_T DEFAULT_DTD_FRAT5COEF_MAP =
00696     DefaultDtdFractMap::init();
00697 DTDFractMap_T DefaultDtdFractMap::init() {
00698     DTDFractMap_T oDFCMap;
00699     oDFCMap[71] = 2.50583571429; oDFCMap[63] = 2.55994571429;
00700     oDFCMap[56] = 2.60841857143; oDFCMap[49] = 2.68888;
00701     oDFCMap[42] = 2.78583714286; oDFCMap[35] = 2.89091428571;
00702     oDFCMap[31] = 2.97871428571; oDFCMap[28] = 3.05521428571;
00703     oDFCMap[24] = 3.15177142857; oDFCMap[21] = 3.22164285714;
00704     oDFCMap[17] = 3.32237142857; oDFCMap[14] = 3.38697142857;
00705     oDFCMap[10] = 3.44204285714; oDFCMap[7] = 3.46202857143;
00706     oDFCMap[5] = 3.47177142857; oDFCMap[3] = 3.4792;
00707     oDFCMap[1] = 3.48947142857; // oDFCMap[0] = 3.49111428571;
00708     return oDFCMap;
00709 }
00710
00712 const DTDProbMap_T DEFAULT_DTD_PROB_MAP =
00713     DefaultDtdProbMap::init();
00714 DTDProbMap_T DefaultDtdProbMap::init() {
00715     DTDProbMap_T oDPMap;
00716     oDPMap[-330] = 0; oDPMap[-150] = 0.1; oDPMap[-92] = 0.2;
00717     oDPMap[-55] = 0.3; oDPMap[-34] = 0.4; oDPMap[-21] = 0.5;
00718     oDPMap[-12] = 0.6; oDPMap[-6] = 0.7; oDPMap[-3] = 0.8;
00719     oDPMap[-1] = 0.9; oDPMap[0] = 1.0;
00720     return oDPMap;
00721 }
00722
00723 // ////////// Key and display related //////////
00726 const std::string DEFAULT_KEY_FLD_DELIMITER ("");
00727
00730 const std::string DEFAULT_KEY_SUB_FLD_DELIMITER ("");
00731
00733 const boost::char_separator<char> DEFAULT_KEY_TOKEN_DELIMITER (";", " ");
00734
00736 const OnDStringList_T DEFAULT_OND_STRING_LIST;
00737
00738
00739 // ////////// BomManager-related constants //////////
00742 const std::string DISPLAY_LEVEL_STRING_ARRAY[51] =
00743 { "", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ",
00744   " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ",
00745   " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ",
00746   " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ",
00747   " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ",
00748   " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ",
00749   " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ",
00750   " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ",
00751   " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ",
00752   " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " "

```

```

00753     "
00754     "
00755     "
00756     "
00757     "
00758     "
00759     "
00760     "
00761     "
00762     "
00763     "
00764     "
00765     "
00766     "
00767     "
00768     "
00769     "
00770     "
00771     "
00772     "
00773     "
00774     "
00775     "
00776     "
00777     "
00778     "
00779     "
00780     "
00781     "
00782     "
    " };
00783
00784
00785 }
```

### 33.23 stdair/basic/BasConst\_BomDisplay.hpp File Reference

```

#include <string>
#include <boost/tokenizer.hpp>
```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

#### Variables

- const std::string [stdair::DEFAULT\\_KEY\\_FLD\\_DELIMITER](#)
- const std::string [stdair::DEFAULT\\_KEY\\_SUB\\_FLD\\_DELIMITER](#)
- const boost::char\_separator< char > [stdair::DEFAULT\\_KEY\\_TOKEN\\_DELIMITER](#)

### 33.24 BasConst\_BomDisplay.hpp

```

00001 #ifndef __STDAIR_BAS_BASCONST_BOMMANAGER_HPP
00002 #define __STDAIR_BAS_BASCONST_BOMMANAGER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // Boost
00010 #include <boost/tokenizer.hpp>
00011
00012 namespace stdair {
00013
00016     extern const std::string DISPLAY_LEVEL_STRING_ARRAY[51];
00017
00020     extern const std::string DEFAULT_KEY_FLD_DELIMITER;
00021
00024     extern const std::string DEFAULT_KEY_SUB_FLD_DELIMITER;
```



```

00025
00027     extern const boost::char_separator<char> DEFAULT_KEY_TOKEN_DELIMITER;
00028
00029 }
00030 #endif // __STDAIR_BAS_BASCONST_BOMMANAGER_HPP

```

### 33.25 stdair/basic/BasConst\_BookingClass.hpp File Reference

```

#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/stdair_fare_types.hpp>

```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

#### Variables

- const Distance\_T [stdair::DEFAULT\\_DISTANCE\\_VALUE](#)
- const ClassCode\_T [stdair::DEFAULT\\_CLOSED\\_CLASS\\_CODE](#)
- const NbOfBookings\_T [stdair::DEFAULT\\_CLASS\\_NB\\_OF\\_BOOKINGS](#)
- const NbOfBookings\_T [stdair::DEFAULT\\_CLASS\\_TOTAL\\_NB\\_OF\\_BOOKINGS](#)
- const NbOfBookings\_T [stdair::DEFAULT\\_CLASS\\_UNCONSTRAINED\\_DEMAND](#)
- const NbOfBookings\_T [stdair::DEFAULT\\_CLASS\\_REMAINING\\_DEMAND\\_MEAN](#)
- const NbOfBookings\_T [stdair::DEFAULT\\_CLASS\\_REMAINING\\_DEMAND\\_STANDARD\\_DEVIATION](#)
- const NbOfCancellations\_T [stdair::DEFAULT\\_CLASS\\_NB\\_OF\\_CANCELLATIONS](#)
- const NbOfNoShows\_T [stdair::DEFAULT\\_CLASS\\_NB\\_OF\\_NOSHOWS](#)
- const CabinCapacity\_T [stdair::DEFAULT\\_CABIN\\_CAPACITY](#)
- const CommittedSpace\_T [stdair::DEFAULT\\_COMMITTED\\_SPACE](#)
- const BlockSpace\_T [stdair::DEFAULT\\_BLOCK\\_SPACE](#)
- const Availability\_T [stdair::DEFAULT\\_NULL\\_AVAILABILITY](#)
- const Availability\_T [stdair::DEFAULT\\_AVAILABILITY](#)
- const CensorshipFlag\_T [stdair::DEFAULT\\_CLASS\\_CENSORSHIPFLAG](#)
- const BookingLimit\_T [stdair::DEFAULT\\_CLASS\\_BOOKING\\_LIMIT](#)
- const AuthorizationLevel\_T [stdair::DEFAULT\\_CLASS\\_AUTHORIZATION\\_LEVEL](#)
- const AuthorizationLevel\_T [stdair::DEFAULT\\_CLASS\\_MAX\\_AUTHORIZATION\\_LEVEL](#)
- const AuthorizationLevel\_T [stdair::DEFAULT\\_CLASS\\_MIN\\_AUTHORIZATION\\_LEVEL](#)
- const OverbookingRate\_T [stdair::DEFAULT\\_CLASS\\_OVERBOOKING\\_RATE](#)
- const Fare\_T [stdair::DEFAULT\\_FARE\\_VALUE](#)
- const Revenue\_T [stdair::DEFAULT\\_REVENUE\\_VALUE](#)
- const PriceCurrency\_T [stdair::DEFAULT\\_CURRENCY](#)
- const Percentage\_T [stdair::DEFAULT\\_LOAD\\_FACTOR\\_VALUE](#)
- const DayDuration\_T [stdair::DEFAULT\\_DAY\\_DURATION](#)
- const double [stdair::DEFAULT\\_EPSILON\\_VALUE](#)

## 33.26 BasConst\_BookingClass.hpp

```

00001 #ifndef __STDAIR_BAS_BASCONST_BOOKINGCLASS_HPP
00002 #define __STDAIR_BAS_BASCONST_BOOKINGCLASS_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009 #include <stdair/stdair_inventory_types.hpp>
00010 #include <stdair/stdair_demand_types.hpp>
00011 #include <stdair/stdair_fare_types.hpp>
00012
00013 namespace stdair {
00014
00015     // ////////// (Segment-)Class-related BOM //////////
00017     extern const Distance_T DEFAULT_DISTANCE_VALUE;
00018
00020     extern const ClassCode_T DEFAULT_CLOSED_CLASS_CODE;
00021
00024     extern const NbOfBookings_T DEFAULT_CLASS_NB_OF_BOOKINGS;
00025
00028     extern const NbOfBookings_T DEFAULT_CLASS_TOTAL_NB_OF_BOOKINGS
00029 ;
00031     extern const NbOfBookings_T DEFAULT_CLASS_UNCONSTRAINED_DEMAND
00032 ;
00034     extern const NbOfBookings_T DEFAULT_CLASS_REMAINING_DEMAND_MEAN
00035 ;
00038     extern const NbOfBookings_T
00039     DEFAULT_CLASS_REMAINING_DEMAND_STANDARD_DEVIATION;
00041     extern const NbOfCancellations_T
00042     DEFAULT_CLASS_NB_OF_CANCELLATIONS;
00044     extern const NbOfNoShows_T DEFAULT_CLASS_NB_OF_NOSHOWS;
00045
00047     extern const CabinCapacity_T DEFAULT_CABIN_CAPACITY;
00048
00050     extern const CommittedSpace_T DEFAULT_COMMITTED_SPACE;
00051
00053     extern const BlockSpace_T DEFAULT_BLOCK_SPACE;
00054
00056     extern const Availability_T DEFAULT_NULL_AVAILABILITY;
00057
00059     extern const Availability_T DEFAULT_AVAILABILITY;
00060
00063     extern const CensorshipFlag_T DEFAULT_CLASS_CENSORSHIPFLAG;
00064
00067     extern const CensorshipFlagList_T
00068     DEFAULT_CLASS_CENSORSHIPFLAG_LIST;
00070
00071     extern const BookingLimit_T DEFAULT_CLASS_BOOKING_LIMIT;
00073     extern const AuthorizationLevel_T
00074     DEFAULT_CLASS_AUTHORIZATION_LEVEL;
00076     extern const AuthorizationLevel_T
00077     DEFAULT_CLASS_MAX_AUTHORIZATION_LEVEL;
00079     extern const AuthorizationLevel_T
00080     DEFAULT_CLASS_MIN_AUTHORIZATION_LEVEL;
00082     extern const OverbookingRate_T DEFAULT_CLASS_OVERBOOKING_RATE
00083 ;
00085     extern const Fare_T DEFAULT_FARE_VALUE;
00086
00088     extern const Revenue_T DEFAULT_REVENUE_VALUE;
00089
00091     extern const PriceCurrency_T DEFAULT_CURRENCY;
00092
00094     extern const Percentage_T DEFAULT_LOAD_FACTOR_VALUE;
00095
00097     extern const DayDuration_T DEFAULT_DAY_DURATION;
00098
00101     extern const double DEFAULT_EPSILON_VALUE;
00102
00103 }
00104 #endif // __STDAIR_BAS_BASCONST_BOOKINGCLASS_HPP

```

### 33.27 stdair/basic/BasConst\_DefaultObject.hpp File Reference

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

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

#### Variables

- const AirportCode\_T [stdair::AIRPORT\\_LHR](#)
- const AirportCode\_T [stdair::AIRPORT\\_SYD](#)
- const CityCode\_T [stdair::POS\\_LHR](#)
- const DayDuration\_T [stdair::NO\\_ADVANCE\\_PURCHASE](#)
- const SaturdayStay\_T [stdair::SATURDAY\\_STAY](#)
- const SaturdayStay\_T [stdair::NO\\_SATURDAY\\_STAY](#)
- const ChangeFees\_T [stdair::CHANGE\\_FEES](#)
- const ChangeFees\_T [stdair::NO\\_CHANGE\\_FEES](#)
- const NonRefundable\_T [stdair::NON\\_REFUNDABLE](#)
- const NonRefundable\_T [stdair::NO\\_NON\\_REFUNDABLE](#)
- const DayDuration\_T [stdair::NO\\_STAY\\_DURATION](#)
- const CabinCode\_T [stdair::CABIN\\_Y](#)
- const AirlineCode\_T [stdair::AIRLINE\\_CODE\\_BA](#)
- const ClassCode\_T [stdair::CLASS\\_CODE\\_Y](#)
- const ClassCode\_T [stdair::CLASS\\_CODE\\_Q](#)
- const AirportCode\_T [stdair::AIRPORT\\_SIN](#)
- const AirportCode\_T [stdair::AIRPORT\\_BKK](#)
- const CityCode\_T [stdair::POS\\_SIN](#)
- const CabinCode\_T [stdair::CABIN\\_ECO](#)
- const FrequentFlyer\_T [stdair::FREQUENT\\_FLYER\\_MEMBER](#)

### 33.28 BasConst\_DefaultObject.hpp

```
00001 #ifndef __STDAIR_BAS_BASCONST_DEFAULTOBJECT_HPP
00002 #define __STDAIR_BAS_BASCONST_DEFAULTOBJECT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_types.hpp>
00009
00010 namespace stdair {
00011
00012     // ////////////////////////////////// Fare and Yield related BOM Tree////////
00014     extern const AirportCode_T AIRPORT_LHR;
00015
00017     extern const AirportCode_T AIRPORT_SYD;
00018
00020     extern const CityCode_T POS_LHR;
00021
00023     extern const DayDuration_T NO_ADVANCE_PURCHASE;
00024
00026     extern const SaturdayStay_T SATURDAY_STAY;
00027
00029     extern const SaturdayStay_T NO_SATURDAY_STAY;
00030
00032     extern const ChangeFees_T CHANGE_FEES;
00033
00035     extern const ChangeFees_T NO_CHANGE_FEES;
00036
00038     extern const NonRefundable_T NON_REFUNDABLE;
```

```

00039
00041     extern const NonRefundable_T NO_NON_REFUNDABLE;
00042
00044     extern const DayDuration_T NO_STAY_DURATION;
00045
00047     extern const CabinCode_T CABIN_Y;
00048
00050     extern const AirlineCode_T AIRLINE_CODE_BA;
00051
00053     extern const ClassCode_T CLASS_CODE_Y;
00054
00055     // ////////// Travel Solution related objects//////////
00057     extern const ClassCode_T CLASS_CODE_Q;
00058
00059     // ////////// Booking request related objects//////////
00061     extern const AirportCode_T AIRPORT_SIN;
00062
00064     extern const AirportCode_T AIRPORT_BKK;
00065
00067     extern const CityCode_T POS_SIN;
00068
00070     extern const CabinCode_T CABIN_ECO;
00071
00073     extern const FrequentFlyer_T FREQUENT_FLYER_MEMBER;
00074
00075 }
00076 #endif // __STDAIR_BAS_BASCONST_DEFAULTOBJECT_HPP

```

### 33.29 stdair/basic/BasConst\_Event.hpp File Reference

```

#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/stdair_event_types.hpp>

```

#### Namespaces

- [stdair](#)  
*Handle on the StdAir library context.*

#### 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](#)

### 33.30 BasConst\_Event.hpp

```

00001 #ifndef __STDAIR_BAS_BASCONST_EVENT_HPP
00002 #define __STDAIR_BAS_BASCONST_EVENT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009 #include <stdair/stdair_date_time_types.hpp>
00010 #include <stdair/stdair_event_types.hpp>
00011
00012 namespace stdair {
00013
00015     extern const Count_T DEFAULT_PROGRESS_STATUS;
00016
00019     extern const Date_T DEFAULT_EVENT_OLDEST_DATE;
00020
00023     extern const DateTime_T DEFAULT_EVENT_OLDEST_DATETIME;
00024
00026     extern const Percentage_T MAXIMUM_PROGRESS_STATUS;
00027
00028 }
00029 #endif // __STDAIR_BAS_BASCONST_EVENT_HPP

```

### 33.31 stdair/basic/BasConst\_General.hpp File Reference

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

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

#### Variables

- const std::string [stdair::DEFAULT\\_BOM\\_ROOT\\_KEY](#)
- const NbOfFlightDates\_T [stdair::DEFAULT\\_NB\\_OF\\_FLIGHTDATES](#)
- const unsigned int [stdair::DEFAULT\\_FLIGHT\\_SPEED](#)
- const BookingRatio\_T [stdair::DEFAULT\\_OND\\_BOOKING\\_RATE](#)
- const Count\_T [stdair::SECONDS\\_IN\\_ONE\\_DAY](#)
- const Count\_T [stdair::MILLISECONDS\\_IN\\_ONE\\_SECOND](#)
- const Date\_T [stdair::DEFAULT\\_DATE](#)
- const DateTime\_T [stdair::DEFAULT\\_DATETIME](#)
- const Duration\_T [stdair::DEFAULT\\_EPSILON\\_DURATION](#)
- const RandomSeed\_T [stdair::DEFAULT\\_RANDOM\\_SEED](#)
- const Duration\_T [stdair::NULL\\_BOOST\\_TIME\\_DURATION](#)
- const 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 ChannelLabel\_T [stdair::DEFAULT\\_CHANNEL](#)
- const unsigned int [stdair::DEFAULT\\_NUMBER\\_OF\\_SUBDIVISIONS](#)

### 33.32 BasConst\_General.hpp

```
00001 #ifndef __STDAIR_BAS_BASCONST_GENERAL_HPP
00002 #define __STDAIR_BAS_BASCONST_GENERAL_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/stdair_types.hpp>
00011
00012 namespace stdair {
00013
00014     extern const std::string DEFAULT_BOM_ROOT_KEY;
00015
00016     extern const double DEFAULT_EPSILON_VALUE;
00017
00018     extern const CabinCapacity_T DEFAULT_CABIN_CAPACITY;
00019
00020     extern const NbOfFlightDates_T DEFAULT_NB_OF_FLIGHTDATES;
00021
00022     extern const NbOfBookings_T DEFAULT_CLASS_NB_OF_BOOKINGS;
00023
00024     extern const Distance_T DEFAULT_DISTANCE_VALUE;
00025
00026     extern const unsigned int DEFAULT_FLIGHT_SPEED;
00027
00028     extern const Fare_T DEFAULT_FARE_VALUE;
00029
00030     extern const PriceCurrency_T DEFAULT_CURRENCY;
00031
00032     extern const Revenue_T DEFAULT_REVENUE_VALUE;
00033
00034 }
```

```

00045     extern const BookingRatio_T DEFAULT_OND_BOOKING_RATE;
00046
00048     extern const Count_T SECONDS_IN_ONE_DAY;
00049
00051     extern const Count_T MILLISECONDS_IN_ONE_SECOND;
00052
00054     extern const Date_T DEFAULT_DATE;
00055
00057     extern const DateTime_T DEFAULT_DATETIME;
00058
00060     extern const Duration_T DEFAULT_EPSILON_DURATION;
00061
00063     extern const RandomSeed_T DEFAULT_RANDOM_SEED;
00064
00066     extern const Duration_T NULL_BOOST_TIME_DURATION;
00067
00069     extern const Duration_T DEFAULT_NULL_DURATION;
00070
00072     extern const Fare_T DEFAULT_CLASS_FARE_VALUE;
00073
00075     extern const NbofAirlines_T DEFAULT_NB_OF_AIRLINES;
00076
00078     extern const unsigned int DEFAULT_NB_OF_DAYS_IN_A_YEAR;
00079
00081     extern const NbofBookings_T DEFAULT_CLASS_NB_OF_BOOKINGS;
00082
00084     extern const ChannelLabel_T DEFAULT_CHANNEL;
00085
00087     extern const OnDStringList_T DEFAULT_OND_STRING_LIST;
00088
00090     extern const unsigned int DEFAULT_NUMBER_OF_SUBDIVISIONS;
00091
00092 }
00093 #endif // __STDAIR_BAS_BASCONST_GENERAL_HPP

```

### 33.33 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>

```

#### Classes

- struct [stdair::DefaultDCPList](#)
- struct [stdair::DefaultDtdFratMap](#)
- struct [stdair::DefaultDtdProbMap](#)

#### Namespaces

- [stdair](#)  
*Handle on the StdAir library context.*

#### Variables

- const AirlineCode\_T [stdair::DEFAULT\\_AIRLINE\\_CODE](#)
- const AirlineCode\_T [stdair::DEFAULT\\_NULL\\_AIRLINE\\_CODE](#)
- const FlightNumber\_T [stdair::DEFAULT\\_FLIGHT\\_NUMBER](#)
- const 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 BidPrice\_T stdair::DEFAULT\_BID\_PRICE
- const unsigned short stdair::MAXIMAL\_NUMBER\_OF\_LEGS\_IN\_FLIGHT
- const unsigned short stdair::MAXIMAL\_NUMBER\_OF\_SEGMENTS\_IN\_OND
- const Availability\_T stdair::MAXIMAL\_AVAILABILITY
- const SeatIndex\_T stdair::DEFAULT\_SEAT\_INDEX
- const 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

### 33.34 BasConst\_Inventory.hpp

```

00001 #ifndef __STDAIR_BAS_BASCONST_INVENTORY_HPP
00002 #define __STDAIR_BAS_BASCONST_INVENTORY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_inventory_types.hpp>
00009 #include <stdair/stdair_date_time_types.hpp>
00010 #include <stdair/basic/ForecastingMethod.hpp>
00011 #include <stdair/basic/UnconstrainingMethod.hpp>
00012 #include <stdair/basic/PreOptimisationMethod.hpp>
00013 #include <stdair/basic/OptimisationMethod.hpp>
00014 #include <stdair/basic/PartnershipTechnique.hpp>
00015
00016 namespace stdair {
00017
00018     // ////////// Inventory-related BOM //////////
00019     extern const AirlineCode_T DEFAULT_AIRLINE_CODE;
00020
00021     extern const AirlineCode_T DEFAULT_NULL_AIRLINE_CODE;
00022
00023     extern const AirlineCodeList_T DEFAULT_AIRLINE_CODE_LIST;
00024
00025     extern const FlightNumber_T DEFAULT_FLIGHT_NUMBER;
00026
00027     extern const FlightNumber_T DEFAULT_FLIGHT_NUMBER_FF;
00028
00029     extern const TableID_T DEFAULT_TABLE_ID;
00030
00031     extern const Date_T DEFAULT_DEPARTURE_DATE;
00032
00033     extern const AirportCode_T DEFAULT_AIRPORT_CODE;
00034
00035     extern const AirportCode_T DEFAULT_NULL_AIRPORT_CODE;
00036
00037     extern const AirportCode_T DEFAULT_ORIGIN;
00038
00039     extern const AirportCode_T DEFAULT_DESTINATION;
00040
00041     extern const CabinCode_T DEFAULT_CABIN_CODE;
00042
00043     extern const FamilyCode_T DEFAULT_FARE_FAMILY_CODE;
00044
00045     extern const FamilyCode_T DEFAULT_NULL_FARE_FAMILY_CODE;
00046

```

```

00060
00062     extern const PolicyCode_T DEFAULT_POLICY_CODE;
00063
00065     extern const NestingStructureCode_T
DEFAULT_NESTING_STRUCTURE_CODE;
00066
00068     extern const NestingStructureCode_T
DISPLAY_NESTING_STRUCTURE_CODE;
00069
00071     extern const NestingStructureCode_T
YIELD_BASED_NESTING_STRUCTURE_CODE;
00072
00074     extern const NestingNodeCode_T DEFAULT_NESTING_NODE_CODE;
00075
00077     extern const ClassCode_T DEFAULT_CLASS_CODE;
00078
00080     extern const ClassCode_T DEFAULT_NULL_CLASS_CODE;
00081
00083     extern const ClassList_StringList_T
DEFAULT_CLASS_CODE_LIST;
00084
00086     extern const BidPrice_T DEFAULT_BID_PRICE;
00087
00089     extern const BidPriceVector_T DEFAULT_BID_PRICE_VECTOR;
00090
00094     extern const unsigned short MAXIMAL_NUMBER_OF_LEGS_IN_FLIGHT;
00095
00098     extern const unsigned short MAXIMAL_NUMBER_OF_SEGMENTS_IN_OND;
00099
00101     extern const Availability_T MAXIMAL_AVAILABILITY;
00102
00104     extern const SeatIndex_T DEFAULT_SEAT_INDEX;
00105
00107     extern const NbOfSeats_T DEFAULT_NULL_BOOKING_NUMBER;
00108
00110     extern const CapacityAdjustment_T
DEFAULT_NULL_CAPACITY_ADJUSTMENT;
00111
00113     extern const UPR_T DEFAULT_NULL_UPR;
00114
00116     extern const std::string DEFAULT_FARE_FAMILY_VALUE_TYPE;
00117
00119     extern const std::string DEFAULT_SEGMENT_CABIN_VALUE_TYPE;
00120
00122     extern const int DEFAULT_MAX_DTD;
00123
00125     extern const DCPList_T DEFAULT_DCP_LIST;
00126     struct DefaultDCPList { static DCPList_T init(); };
00127
00129     extern const DTDFratMap_T DEFAULT_DTD_FRAT5COEF_MAP;
00130     struct DefaultDtdFratMap { static DTDFratMap_T
init(); };
00131
00133     extern const DTDProbMap_T DEFAULT_DTD_PROB_MAP;
00134     struct DefaultDtdProbMap { static DTDProbMap_T
init(); };
00135
00137     extern const ForecastingMethod DEFAULT_FORECASTING_METHOD;
00138
00140     extern const UnconstrainingMethod
DEFAULT_UNCONSTRAINING_METHOD;
00141
00143     extern const PreOptimisationMethod
DEFAULT_PREOPTIMISATION_METHOD;
00144
00146     extern const OptimisationMethod DEFAULT_OPTIMISATION_METHOD;
00147
00149     extern const PartnershipTechnique
DEFAULT_PARTNERSHIP_TECHNIQUE;
00150
00151 }
00152 #endif // __STDAIR_BAS_BASCONST_INVENTORY_HPP

```

### 33.35 stdair/basic/BasConst\_Period\_BOM.hpp File Reference

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



## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## Variables

- const DatePeriod\_T [stdair::BOOST\\_DEFAULT\\_DATE\\_PERIOD](#)
- const DOW\_String\_T [stdair::DEFAULT\\_DOW\\_STRING](#)
- const DateOffset\_T [stdair::DEFAULT\\_DATE\\_OFFSET](#)

### 33.36 BasConst\_Period\_BOM.hpp

```

00001 #ifndef __STDAIR_BAS_BASCONST_PERIOD_BOM_HPP
00002 #define __STDAIR_BAS_BASCONST_PERIOD_BOM_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_types.hpp>
00009
00010 namespace stdair {
00011
00012 // ////////// (Flight-)Period-related BOM //////////
00014 extern const DatePeriod_T BOOST_DEFAULT_DATE_PERIOD;
00015
00017 extern const std::string DOW_STR[];
00018
00020 extern const DOW_String_T DEFAULT_DOW_STRING;
00021
00023 extern const DateOffset_T DEFAULT_DATE_OFFSET;
00024
00026 extern const DayDuration_T DEFAULT_DAY_DURATION;
00027
00028 }
00029 #endif // __STDAIR_BAS_BASCONST_PERIOD_BOM_HPP

```

### 33.37 stdair/basic/BasConst\_Request.hpp File Reference

```

#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/stdair_date_time_types.hpp>

```

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## Variables

- const PartySize\_T [stdair::DEFAULT\\_PARTY\\_SIZE](#)
- const DayDuration\_T [stdair::DEFAULT\\_STAY\\_DURATION](#)
- const WTP\_T [stdair::DEFAULT\\_WTP](#)
- const CityCode\_T [stdair::DEFAULT\\_POS](#)
- const Date\_T [stdair::DEFAULT\\_PREFERRED\\_DEPARTURE\\_DATE](#)
- const Duration\_T [stdair::DEFAULT\\_PREFERRED\\_DEPARTURE\\_TIME](#)
- const DateOffset\_T [stdair::DEFAULT\\_ADVANCE\\_PURCHASE](#)
- const Date\_T [stdair::DEFAULT\\_REQUEST\\_DATE](#)
- const Duration\_T [stdair::DEFAULT\\_REQUEST\\_TIME](#)

- const DateTime\_T stdair::DEFAULT\_REQUEST\_DATE\_TIME
- const CabinCode\_T stdair::DEFAULT\_PREFERRED\_CABIN
- const ChannelLabel\_T stdair::CHANNEL\_DN
- const ChannelLabel\_T stdair::CHANNEL\_IN
- const TripType\_T stdair::TRIP\_TYPE\_ONE\_WAY
- const TripType\_T stdair::TRIP\_TYPE\_ROUND\_TRIP
- const TripType\_T stdair::TRIP\_TYPE\_INBOUND
- const TripType\_T stdair::TRIP\_TYPE\_OUTBOUND
- const FrequentFlyer\_T stdair::DEFAULT\_FF\_TIER
- const PriceValue\_T stdair::DEFAULT\_VALUE\_OF\_TIME
- const IntDuration\_T stdair::HOURL\_CONVERTED\_IN\_SECONDS

### 33.38 BasConst\_Request.hpp

```

00001 #ifndef __STDAIR_BAS_BASCONST_REQUEST_HPP
00002 #define __STDAIR_BAS_BASCONST_REQUEST_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009 #include <stdair/stdair_demand_types.hpp>
00010 #include <stdair/stdair_date_time_types.hpp>
00011
00012 namespace stdair {
00013
00015     extern const PartySize_T DEFAULT_PARTY_SIZE;
00016
00018     extern const DayDuration_T DEFAULT_STAY_DURATION;
00019
00021     extern const WTP_T DEFAULT_WTP;
00022
00024     extern const CityCode_T DEFAULT_POS;
00025
00027     extern const Date_T DEFAULT_PREFERRED_DEPARTURE_DATE;
00028
00030     extern const Duration_T DEFAULT_PREFERRED_DEPARTURE_TIME;
00031
00033     extern const DateOffset_T DEFAULT_ADVANCE_PURCHASE;
00034
00036     extern const Date_T DEFAULT_REQUEST_DATE;
00037
00039     extern const Duration_T DEFAULT_REQUEST_TIME;
00040
00042     extern const DateTime_T DEFAULT_REQUEST_DATE_TIME;
00043
00045     extern const CabinCode_T DEFAULT_PREFERRED_CABIN;
00046
00048     extern const ChannelLabel_T DEFAULT_CHANNEL;
00049
00051     extern const ChannelLabel_T CHANNEL_DN;
00052
00054     extern const ChannelLabel_T CHANNEL_IN;
00055
00057     extern const TripType_T TRIP_TYPE_ONE_WAY;
00058
00060     extern const TripType_T TRIP_TYPE_ROUND_TRIP;
00061
00063     extern const TripType_T TRIP_TYPE_INBOUND;
00064
00066     extern const TripType_T TRIP_TYPE_OUTBOUND;
00067
00069     extern const FrequentFlyer_T DEFAULT_FF_TIER;
00070
00072     extern const PriceValue_T DEFAULT_VALUE_OF_TIME;
00073
00075     extern const IntDuration_T HOURL_CONVERTED_IN_SECONDS;
00076
00077 }
00078 #endif // __STDAIR_BAS_BASCONST_REQUEST_HPP

```

### 33.39 stdair/basic/BasConst\_SellUpCurves.hpp File Reference

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

#### Classes

- struct [stdair::DefaultMap](#)

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.40 BasConst\_SellUpCurves.hpp

```
00001 #ifndef __STDAIR_BAS_BASCONST_SELLUPCURVES_HPP
00002 #define __STDAIR_BAS_BASCONST_SELLUPCURVES_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STDAIR
00008 #include <stdair/stdair_types.hpp>
00009
00010 namespace stdair {
00011
00012     extern const FRAT5Curve_T FRAT5_CURVE_A;
00013     extern const FRAT5Curve_T FRAT5_CURVE_B;
00014     extern const FRAT5Curve_T FRAT5_CURVE_C;
00015     extern const FRAT5Curve_T FRAT5_CURVE_D;
00016
00017     extern const FFDisutilityCurve_T FF_DISUTILITY_CURVE_A;
00018     extern const FFDisutilityCurve_T FF_DISUTILITY_CURVE_B;
00019     extern const FFDisutilityCurve_T FF_DISUTILITY_CURVE_C;
00020     extern const FFDisutilityCurve_T FF_DISUTILITY_CURVE_D;
00021     extern const FFDisutilityCurve_T FF_DISUTILITY_CURVE_E;
00022     extern const FFDisutilityCurve_T FF_DISUTILITY_CURVE_F;
00023
00024     struct DefaultMap {
00025         static FRAT5Curve_T createFRAT5CurveA();
00026         static FRAT5Curve_T createFRAT5CurveB();
00027         static FRAT5Curve_T createFRAT5CurveC();
00028         static FRAT5Curve_T createFRAT5CurveD();
00029         static FFDisutilityCurve_T createFFDisutilityCurveA();
00030         static FFDisutilityCurve_T createFFDisutilityCurveB();
00031         static FFDisutilityCurve_T createFFDisutilityCurveC();
00032         static FFDisutilityCurve_T createFFDisutilityCurveD();
00033         static FFDisutilityCurve_T createFFDisutilityCurveE();
00034         static FFDisutilityCurve_T createFFDisutilityCurveF();
00035     };
00036 }
00037 #endif // __STDAIR_BAS_BASCONST_SELLUPCURVES_HPP
```

### 33.41 stdair/basic/BasConst\_TravelSolution.hpp File Reference

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

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## Variables

- const Duration\_T stdair::DEFAULT\_MINIMAL\_CONNECTION\_TIME
- const Duration\_T stdair::DEFAULT\_MAXIMAL\_CONNECTION\_TIME
- const FlightPathCode\_T stdair::DEFAULT\_FLIGHTPATH\_CODE
- const Availability\_T stdair::DEFAULT\_CLASS\_AVAILABILITY
- const AvailabilityStatus\_T stdair::DEFAULT\_AVAILABILITY\_STATUS
- const unsigned short stdair::DEFAULT\_NUMBER\_OF\_REQUIRED\_SEATS
- const MatchingIndicator\_T stdair::DEFAULT\_MATCHING\_INDICATOR
- const AirlineCode\_T stdair::DEFAULT\_DICO\_STUDIED\_AIRLINE

## 33.42 BasConst\_TravelSolution.hpp

```

00001 #ifndef __STDAIR_BAS_BASCONST_TRAVELSOLUTION_HPP
00002 #define __STDAIR_BAS_BASCONST_TRAVELSOLUTION_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_types.hpp>
00009
00010 namespace stdair {
00011
00012     // ////////// Travel Solutions //////////
00014     extern const Distance_T DEFAULT_DISTANCE_VALUE;
00015
00017     extern const Duration_T DEFAULT_MINIMAL_CONNECTION_TIME;
00018
00020     extern const Duration_T DEFAULT_MAXIMAL_CONNECTION_TIME;
00021
00023     extern const Duration_T NULL_BOOST_TIME_DURATION;
00024
00026     extern const FlightPathCode_T DEFAULT_FLIGHTPATH_CODE;
00027
00029     extern const Availability_T DEFAULT_CLASS_AVAILABILITY;
00030
00032     extern const AvailabilityStatus_T
    DEFAULT_AVAILABILITY_STATUS;
00033
00035     extern const unsigned short DEFAULT_NUMBER_OF_REQUIRED_SEATS;
00036
00039     extern const MatchingIndicator_T DEFAULT_MATCHING_INDICATOR;
00040
00042     extern const Revenue_T DEFAULT_REVENUE_VALUE;
00043
00045     extern const AirlineCode_T DEFAULT_DICO_STUDIED_AIRLINE;
00046
00048     extern const Date_T DEFAULT_DICO_STUDIED_DATE;
00049
00050 }
00051 #endif // __STDAIR_BAS_BASCONST_TRAVELSOLUTION_HPP

```

## 33.43 stdair/basic/BasConst\_Yield.hpp File Reference

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

## Namespaces

- [stdair](#)  
*Handle on the StdAir library context.*

## Variables

- const Yield\_T stdair::DEFAULT\_YIELD\_VALUE
- const Yield\_T stdair::DEFAULT\_YIELD\_MAX\_VALUE

### 33.44 BasConst\_Yield.hpp

```

00001 #ifndef __STDAIR_BAS_BASCONST_YIELD_HPP
00002 #define __STDAIR_BAS_BASCONST_YIELD_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_types.hpp>
00009
00010 namespace stdair {
00011
00012 // ////////// (Leg-)Yield-related BOM //////////
00014 extern const Yield_T DEFAULT_YIELD_VALUE;
00015
00017 extern const Yield_T DEFAULT_YIELD_MAX_VALUE;
00018
00019 }
00020 #endif // __STDAIR_BAS_BASCONST_YIELD_HPP

```

### 33.45 stdair/basic/BasDBParams.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/basic/BasDBParams.hpp>

```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.46 BasDBParams.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasDBParams.hpp>
00009
00010 namespace stdair {
00011
00012 // //////////////////////////////////////
00013 BasDBParams::BasDBParams() {
00014 }
00015
00016 // //////////////////////////////////////
00017 BasDBParams::BasDBParams (const BasDBParams& iDBParams)
00018 : _user (iDBParams._user), _passwd (iDBParams._passwd),
00019   _host (iDBParams._host), _port (iDBParams._port),
00020   _dbname (iDBParams._dbname) {
00021 }
00022
00023 // //////////////////////////////////////
00024 BasDBParams::BasDBParams (const std::string& iDBUser,
00025                           const std::string& iDBPasswd,
00026                           const std::string& iDBHost,
00027                           const std::string& iDBPort,
00028                           const std::string& iDBName)
00029 : _user (iDBUser), _passwd (iDBPasswd), _host (iDBHost), _port (iDBPort),
00030   _dbname (iDBName) {
00031 }
00032
00033 // //////////////////////////////////////
00034 BasDBParams::~BasDBParams () {
00035 }
00036
00037 // //////////////////////////////////////
00038 const std::string BasDBParams::describe() const {
00039     return toString();
00040 }

```

```

00041
00042 // //////////////////////////////////////
00043 std::string BasDBParams::toShortString() const {
00044     std::ostringstream ostr;
00045     ostr << _dbname << "." << _user << "@" << _host << ":" << _port;
00046     return ostr.str();
00047 }
00048
00049 // //////////////////////////////////////
00050 std::string BasDBParams::toString() const {
00051     std::ostringstream ostr;
00052     ostr << _dbname << "." << _user << "@" << _host << ":" << _port;
00053     return ostr.str();
00054 }
00055
00056 // //////////////////////////////////////
00057 bool BasDBParams::check() const {
00058     if (_user.empty() == true || _passwd.empty() == true
00059         || _host.empty() == true || _port.empty()
00060         || _dbname.empty() == true) {
00061         return false;
00062     }
00063     return true;
00064 }
00065
00066 }

```

### 33.47 stdair/basic/BasDBParams.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/stdair_db.hpp>
#include <stdair/basic/StructAbstract.hpp>

```

#### Classes

- struct [stdair::BasDBParams](#)  
Structure holding the parameters for connection to a database.

#### Namespaces

- [stdair](#)  
Handle on the StdAir library context.

### 33.48 BasDBParams.hpp

```

00001 #ifndef __STDAIR_BAS_BASDBPARAMS_HPP
00002 #define __STDAIR_BAS_BASDBPARAMS_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // Stdair
00011 #include <stdair/stdair_db.hpp>
00012 #include <stdair/basic/StructAbstract.hpp>
00013
00014 namespace stdair {
00015
00016     struct BasDBParams : public StructAbstract {
00017     public:
00018         // ////////////////////////////////// Getters //
00019         const std::string& getUser() const {
00020             return _user;
00021         }
00022
00023         const std::string& getPassword() const {
00024             return _passwd;
00025         }
00026     };
00027 }

```

```

00030     }
00031
00033     const std::string& getHost() const {
00034         return _host;
00035     }
00036
00038     const std::string& getPort() const {
00039         return _port;
00040     }
00041
00043     const std::string& getDBName() const {
00044         return _dbname;
00045     }
00046
00047
00048     // ////////// Setters //////////
00050     void setUser (const std::string& iUser) {
00051         _user = iUser;
00052     }
00053
00055     void setPassword (const std::string& iPasswd) {
00056         _passwd = iPasswd;
00057     }
00058
00060     void setHost (const std::string& iHost) {
00061         _host = iHost;
00062     }
00063
00065     void setPort (const std::string& iPort) {
00066         _port = iPort;
00067     }
00068
00070     void setDBName (const std::string& iDBName) {
00071         _dbname = iDBName;
00072     }
00073
00074
00075 public:
00076     // ////////// Busines methods //////////
00080     bool check() const;
00081
00082
00083 public:
00084     // ////////// Display methods //////////
00088     const std::string describe() const;
00089
00093     std::string toShortString() const;
00094
00098     std::string toString() const;
00099
00100
00101 public:
00105     BasDBParams (const std::string& iDBUser, const std::string& iDBPasswd,
00106                 const std::string& iDBHost, const std::string& iDBPort,
00107                 const std::string& iDBName);
00108
00112     BasDBParams ();
00113
00117     BasDBParams (const BasDBParams&);
00118
00122     ~BasDBParams ();
00123
00124
00125 private:
00126     // ////////// Attributes //////////
00128     std::string _user;
00130     std::string _passwd;
00132     std::string _host;
00134     std::string _port;
00136     std::string _dbname;
00137 };
00138
00139 }
00140 #endif // __STDAIR_BAS_BASDBPARAMS_HPP

```

### 33.49 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

- [stdair](#)

*Handle on the StdAir library context.*

## 33.50 BasFileMgr.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // Boost (STL Extension)
00007 // Boost Filesystem (http://www.boost.org/doc/libs/1_41_0/libs/filesystem/doc/index.htm)
00008 #include <boost/version.hpp>
00009 #if BOOST_VERSION >= 103500
00010 #include <boost/filesystem.hpp>
00011 #else // BOOST_VERSION >= 103500
00012 #include <boost/filesystem/path.hpp>
00013 #include <boost/filesystem/operations.hpp>
00014 #endif // BOOST_VERSION >= 103500
00015 // StdAir
00016 #include <stdair/basic/BasFileMgr.hpp>
00017
00018 namespace boostfs = boost::filesystem;
00019
00020 namespace stdair {
00021
00022 // //////////////////////////////////////
00023 bool BasFileMgr::doesExistAndIsReadable (const std::string& iFilepath)
00024 {
00025     bool oFine = false;
00026     boostfs::path lPath (iFilepath);
00027
00028     if (boostfs::exists (lPath) == false) {
00029         return oFine;
00030     }
00031
00032     #if BOOST_VERSION >= 103500
00033     if (boostfs::is_regular (lPath) == true) {
00034         oFine = true;
00035     }
00036     #endif // BOOST_VERSION >= 103500
00037
00038     return oFine;
00039 }
00040
00041 }

```

## 33.51 stdair/basic/BasFileMgr.hpp File Reference

```
#include <string>
```

## Classes

- struct [stdair::BasFileMgr](#)

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*



### 33.52 BasFileMgr.hpp

```

00001 #ifndef __STDAIR_BAS_BASFILEMGR_HPP
00002 #define __STDAIR_BAS_BASFILEMGR_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009
00010 namespace stdair {
00011
00012     struct BasFileMgr {
00013     public:
00014
00015         // ////////////////////////////////// Functional Support Methods //////////////////////////////////
00016         static bool doesExistAndIsReadable (const std::string& iFilepath);
00017
00018     };
00019
00020 }
00021
00022 #endif // __STDAIR_BAS_BASFILEMGR_HPP

```

### 33.53 stdair/basic/BasLogParams.cpp File Reference

```

#include <cassert>
#include <iostream>
#include <sstream>
#include <stdair/basic/BasLogParams.hpp>

```

#### Namespaces

- **stdair**

*Handle on the StdAir library context.*

### 33.54 BasLogParams.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <iostream>
00007 #include <sstream>
00008 // StdAir
00009 #include <stdair/basic/BasLogParams.hpp>
00010
00011 namespace stdair {
00012
00013     // //////////////////////////////////
00014     BasLogParams::BasLogParams()
00015         : _logLevel (LOG::DEBUG), _logStream (std::cout),
00016         _forceMultipleInit (false) {
00017         assert (false);
00018     }
00019
00020     // //////////////////////////////////
00021     BasLogParams::BasLogParams (const BasLogParams& iLogParams)
00022         : _logLevel (iLogParams._logLevel), _logStream (iLogParams._logStream),
00023         _forceMultipleInit (iLogParams._forceMultipleInit) {
00024     }
00025
00026     // //////////////////////////////////
00027     BasLogParams::BasLogParams (const LOG::EN_LogLevel iLogLevel,
00028                                 std::ostream& ioLogOutputStream,
00029                                 const bool iForceMultipleInstance)
00030         : _logLevel (iLogLevel), _logStream (ioLogOutputStream),
00031         _forceMultipleInit (iForceMultipleInstance) {
00032     }
00033
00034     // //////////////////////////////////
00035     BasLogParams::~BasLogParams () {

```

```

00036     }
00037
00038     // //////////////////////////////////////
00039     const std::string BasLogParams::describe() const {
00040         return toString();
00041     }
00042
00043     // //////////////////////////////////////
00044     std::string BasLogParams::toShortString() const {
00045         const std::string isForcedStr = (_forceMultipleInit == true)? " (forced)":"";
00046         std::ostringstream oStr;
00047         oStr << LOG::_logLevels[_logLevel] << isForcedStr;
00048         return oStr.str();
00049     }
00050
00051     // //////////////////////////////////////
00052     std::string BasLogParams::toString() const {
00053         std::ostringstream oStr;
00054         oStr << LOG::_logLevels[_logLevel];
00055         return oStr.str();
00056     }
00057
00058 }

```

### 33.55 stdair/basic/BasLogParams.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/stdair_log.hpp>
#include <stdair/basic/StructAbstract.hpp>

```

#### Classes

- struct [stdair::BasLogParams](#)  
Structure holding parameters for logging.

#### Namespaces

- [stdair](#)  
Handle on the StdAir library context.

### 33.56 BasLogParams.hpp

```

00001 #ifndef __STDAIR_BAS_BASLOGPARAMS_HPP
00002 #define __STDAIR_BAS_BASLOGPARAMS_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // Stdair
00011 #include <stdair/stdair_log.hpp>
00012 #include <stdair/basic/StructAbstract.hpp>
00013
00014 namespace stdair {
00015
00016     struct BasLogParams : public StructAbstract {
00017         friend class Logger;
00018     public:
00019         // ////////////////////////////////// Getters //////////////////////////////////
00020         const LOG::EN_LogLevel& getLogLevel() const {
00021             return _logLevel;
00022         }
00023
00024         std::ostream& getLogStream() const {
00025             return _logStream;
00026         }
00027
00028     }
00029
00030 }
00031
00032
00033
00034
00035
00036

```

```

00040     const bool getForcedInitialisationFlag() const {
00041         return _forceMultipleInit;
00042     }
00043
00044
00045     // ////////// Setters //////////
00049     void setForcedInitialisationFlag (const bool iForceMultipleInstance) {
00050         _forceMultipleInit = iForceMultipleInstance;
00051     }
00052
00053
00054     public:
00055         // ////////// Busines methods //////////
00059         bool check() const;
00060
00061
00062     public:
00063         // ////////// Display methods //////////
00067         const std::string describe() const;
00068
00072         std::string toShortString() const;
00073
00077         std::string toString() const;
00078
00079
00080     public:
00089         BasLogParams (const LOG::EN_LogLevel iLogLevel,
00090                     std::ostream& ioLogOutputStream,
00091                     const bool iForceMultipleInstance = false);
00092
00096         BasLogParams (const BasLogParams&);
00097
00101         ~BasLogParams();
00102
00103     private:
00107         BasLogParams();
00108
00109
00110     private:
00111         // ////////// Attributes //////////
00115         const LOG::EN_LogLevel _logLevel;
00116
00120         std::ostream& _logStream;
00121
00135         bool _forceMultipleInit;
00136     };
00137
00138 }
00139 #endif // __STDAIR_BAS_BASLOGPARAMS_HPP

```

### 33.57 stdair/basic/BasParserHelperTypes.hpp File Reference

```

#include <string>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/service/Logger.hpp>

```

#### Classes

- struct [stdair::date\\_time\\_element< MIN, MAX >](#)

#### Namespaces

- [stdair](#)  
*Handle on the StdAir library context.*

#### Typedefs

- typedef date\_time\_element< 0, 23 > [stdair::hour\\_t](#)
- typedef date\_time\_element< 0, 59 > [stdair::minute\\_t](#)

- typedef date\_time\_element< 0, 59 > [stdair::second\\_t](#)
- typedef date\_time\_element< 1900, 2100 > [stdair::year\\_t](#)
- typedef date\_time\_element< 1, 12 > [stdair::month\\_t](#)
- typedef date\_time\_element< 1, 31 > [stdair::day\\_t](#)

## Functions

- template<int MIN, int MAX>  
date\_time\_element< MIN, MAX > [stdair::operator\\*](#) (const date\_time\_element< MIN, MAX > &o1, const date\_time\_element< MIN, MAX > &o2)
- template<int MIN, int MAX>  
date\_time\_element< MIN, MAX > [stdair::operator+](#) (const date\_time\_element< MIN, MAX > &o1, const date\_time\_element< MIN, MAX > &o2)

## 33.58 BasParserHelperTypes.hpp

```

00001 #ifndef __STDAIR_BAS_BASCOMPARSERHELPERTYPES_HPP
00002 #define __STDAIR_BAS_BASCOMPARSERHELPERTYPES_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <sstream>
00010 // StdAir
00011 #include <stdair/stdair_exceptions.hpp>
00012 #include <stdair/service/Logger.hpp>
00013
00014 namespace stdair {
00015
00016 // //////////////////////////////////////
00017 //
00018 // Parser structure helper
00019 //
00020 // //////////////////////////////////////
00022 template <int MIN = 0, int MAX = 0>
00023 struct date_time_element {
00024     unsigned int _value;
00025
00026     // ////////////////////////////////// Constructors //////////////////////////////////
00028     date_time_element () { }
00030     date_time_element (const date_time_element& t) : _value (t._value) {
00032     }
00032     date_time_element (int i) : _value (i) { }
00034     void check () const {
00035         if (_value < MIN || _value > MAX) {
00036             std::ostringstream oMessage;
00037             oMessage << "The value: " << _value << " is out of range ("
00038                 << MIN << ", " << MAX << ")";
00039             throw stdair::ParserException (oMessage.str());
00040         }
00041     }
00042 };
00043
00045 template <int MIN, int MAX>
00046 inline date_time_element<MIN,
00047     MAX> operator*(const
00048 date_time_element<MIN, MAX>& o1,
00049     const date_time_element<MIN, MAX>& o2)
00050 {
00049     return date_time_element<MIN, MAX> (o1._value * o2.
00050 _value);
00051 }
00053 template <int MIN, int MAX>
00054 inline date_time_element<MIN,
00055     MAX> operator+(const
00056 date_time_element<MIN, MAX>& o1,
00057     const date_time_element<MIN, MAX>& o2)
00058 {
00057     return date_time_element<MIN, MAX> (o1._value + o2.
00058 _value);
00059 }
00061 typedef date_time_element<0, 23> hour_t;
00062 typedef date_time_element<0, 59> minute_t;

```

```

00063     typedef date_time_element<0, 59> second_t;
00064     typedef date_time_element<1900, 2100> year_t;
00065     typedef date_time_element<1, 12> month_t;
00066     typedef date_time_element<1, 31> day_t;
00067
00068 }
00069 #endif // __STDAIR_BASIC_BASCOMPARSERHELPERTYPES_HPP

```

### 33.59 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

- [stdair](#)

*Handle on the StdAir library context.*

#### Typedefs

- typedef std::istreambuf\_iterator< char > [stdair::base\\_iterator\\_t](#)
- typedef boost::spirit::multi\_pass< base\_iterator\_t > [stdair::iterator\\_t](#)
- typedef boost::spirit::qi::int\_parser< unsigned int, 10, 1, 1 > [stdair::int1\\_p\\_t](#)
- typedef boost::spirit::qi::uint\_parser< int, 10, 2, 2 > [stdair::uint2\\_p\\_t](#)
- typedef boost::spirit::qi::uint\_parser< int, 10, 4, 4 > [stdair::uint4\\_p\\_t](#)
- typedef boost::spirit::qi::uint\_parser< int, 10, 1, 4 > [stdair::uint1\\_4\\_p\\_t](#)
- typedef boost::spirit::qi::uint\_parser< hour\_t, 10, 2, 2 > [stdair::hour\\_p\\_t](#)
- typedef boost::spirit::qi::uint\_parser< minute\_t, 10, 2, 2 > [stdair::minute\\_p\\_t](#)
- typedef boost::spirit::qi::uint\_parser< second\_t, 10, 2, 2 > [stdair::second\\_p\\_t](#)
- typedef boost::spirit::qi::uint\_parser< year\_t, 10, 4, 4 > [stdair::year\\_p\\_t](#)
- typedef boost::spirit::qi::uint\_parser< month\_t, 10, 2, 2 > [stdair::month\\_p\\_t](#)
- typedef boost::spirit::qi::uint\_parser< day\_t, 10, 2, 2 > [stdair::day\\_p\\_t](#)

### 33.60 BasParserTypes.hpp

```

00001 #ifndef __STDAIR_BASIC_BASCOMPARSERTYPES_HPP
00002 #define __STDAIR_BASIC_BASCOMPARSERTYPES_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // Boost Spirit (Parsing)
00010 #include <boost/spirit/include/qi.hpp>
00011 #include <boost/spirit/include/phoenix_core.hpp>
00012 #include <boost/spirit/include/phoenix_operator.hpp>
00013 #include <boost/spirit/include/support_multi_pass.hpp>
00014 // STDAIR
00015 #include <stdair/basic/BasParserHelperTypes.hpp>
00016
00017 namespace stdair {
00018
00019 // //////////////////////////////////////
00020 //
00021 // Definition of Basic Types
00022 //
00023 // //////////////////////////////////////
00024 // The types of iterator, scanner and rule are then derived from

```

```

00025 // the parsing unit.
00026 typedef std::istreambuf_iterator<char> base_iterator_t;
00027 typedef boost::spirit::multi_pass<base_iterator_t> iterator_t;
00028
00029 // //////////////////////////////////////
00030 //
00031 // Parser related types
00032 //
00033 // //////////////////////////////////////
00035 typedef boost::spirit::qi::int_parser<unsigned int, 10, 1, 1> int1_p_t;
00036
00038 typedef boost::spirit::qi::uint_parser<int, 10, 2, 2> uint2_p_t;
00039
00041 typedef boost::spirit::qi::uint_parser<int, 10, 4, 4> uint4_p_t;
00042
00044 typedef boost::spirit::qi::uint_parser<int, 10, 1, 4> uint1_4_p_t;
00045
00047 typedef boost::spirit::qi::uint_parser<hour_t, 10, 2, 2> hour_p_t;
00048 typedef boost::spirit::qi::uint_parser<minute_t, 10, 2, 2> minute_p_t;
00049 typedef boost::spirit::qi::uint_parser<second_t, 10, 2, 2> second_p_t;
00050 typedef boost::spirit::qi::uint_parser<year_t, 10, 4, 4> year_p_t;
00051 typedef boost::spirit::qi::uint_parser<month_t, 10, 2, 2> month_p_t;
00052 typedef boost::spirit::qi::uint_parser<day_t, 10, 2, 2> day_p_t;
00053 }
00054 #endif // __STDAIR_BAS_BASCOMPARSERTYPES_HPP

```

### 33.61 stdair/basic/BasTypes.hpp File Reference

```
#include <string>
```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.62 BasTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BAS_BASTYPES_HPP
00003 #define __STDAIR_BAS_BASTYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <string>
00010
00011 namespace stdair {
00012
00013 }
00014 #endif // __STDAIR_BAS_BASTYPES_HPP

```

### 33.63 stdair/basic/ContinuousAttributeLite.hpp File Reference

```

#include <cassert>
#include <iosfwd>
#include <string>
#include <vector>
#include <map>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/DictionaryManager.hpp>

```

## Classes

- struct `stdair::ContinuousAttributeLite< T >`

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

## Namespaces

- `stdair`

*Handle on the StdAir library context.*

## 33.64 ContinuousAttributeLite.hpp

```

00001 #ifndef __STDAIR_BAS_CONTINUOUSATTRIBUTE_LITE_HPP
00002 #define __STDAIR_BAS_CONTINUOUSATTRIBUTE_LITE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <cassert>
00009 #include <iosfwd>
00010 #include <string>
00011 #include <vector>
00012 #include <map>
00013 // StdAir
00014 #include <stdair/stdair_basic_types.hpp>
00015 // TraDemGen
00016 #include <stdair/stdair_exceptions.hpp>
00017 #include <stdair/basic/DictionaryManager.hpp>
00018
00019 namespace stdair {
00020
00021     template <typename T>
00022     struct ContinuousAttributeLite {
00023     public:
00024         // ////////////////////////////////// Type definitions //////////////////////////////////
00025         typedef std::map<T, stdair::Probability_T> ContinuousDistribution_T;
00026
00027     public:
00028         // ////////////////////////////////// Business Methods //////////////////////////////////
00029         const T getValue(const stdair::Probability_T& iCumulativeProbability)
00030         const{
00031             const DictionaryKey_T& lKey =
00032                 DictionaryManager::valueToKey (iCumulativeProbability);
00033
00034             // Find the first cumulative probability value greater or equal to lKey.
00035             unsigned int idx = 0;
00036             for (; idx < _size; ++idx) {
00037                 if (_cumulativeDistribution.at(idx) > lKey) {
00038                     break;
00039                 }
00040             }
00041
00042             if (idx == 0) {
00043                 return _valueArray.at(idx);
00044             }
00045             if (idx == _size) {
00046                 return _valueArray.at(idx-1);
00047             }
00048
00049             //
00050             const stdair::Probability_T lCumulativeCurrentPoint =
00051                 DictionaryManager::keyToValue (_cumulativeDistribution.at(idx));
00052             const T& lValueCurrentPoint = _valueArray.at(idx);
00053
00054             //
00055             const stdair::Probability_T lCumulativePreviousPoint =
00056                 DictionaryManager::keyToValue (_cumulativeDistribution.at(idx-1));
00057             const T& lValuePreviousPoint = _valueArray.at(idx-1);
00058
00059             if (lCumulativePreviousPoint == lCumulativeCurrentPoint) {
00060                 return lValuePreviousPoint;
00061             }
00062
00063             T oValue= lValuePreviousPoint + (lValueCurrentPoint - lValuePreviousPoint)
00064                 * (iCumulativeProbability - lCumulativePreviousPoint)
00065                 / (lCumulativeCurrentPoint - lCumulativePreviousPoint);
00066
00067             return oValue;
00068

```

```

00077     }
00078
00079     public:
00080         // //////////////// Business Methods ////////////////
00084         const stdair::Probability_T getRemainingProportion(const T&
iValue) const {
00085
00086             // Find the first value greater than iValue.
00087             unsigned int idx = 0;
00088             for (; idx < _size; ++idx) {
00089                 if (_valueArray.at(idx) > iValue) {
00090                     break;
00091                 }
00092             }
00093             if (idx == 0) {
00094                 const stdair::Probability_T oCumulativeProbability =
DictionaryManager::keyToValue (_cumulativeDistribution.at(idx));
00095                 return 1 - oCumulativeProbability;
00096             }
00097             if (idx == _size) {
00098                 const stdair::Probability_T oCumulativeProbability =
DictionaryManager::keyToValue (_cumulativeDistribution.at(idx-1));
00099                 return 1 - oCumulativeProbability;
00100             }
00101             //
00102             const stdair::Probability_T lCumulativeCurrentPoint =
DictionaryManager::keyToValue (_cumulativeDistribution.at(idx));
00103             const T& lValueCurrentPoint = _valueArray.at(idx);
00104             //
00105             const stdair::Probability_T lCumulativePreviousPoint =
DictionaryManager::keyToValue (_cumulativeDistribution.at(idx-1));
00106             const T& lValuePreviousPoint = _valueArray.at(idx-1);
00107             if (lValuePreviousPoint == lValueCurrentPoint) {
00108                 return 1 - lCumulativePreviousPoint;
00109             }
00110             const stdair::Probability_T oCumulativeProbability =
lCumulativePreviousPoint + (lCumulativeCurrentPoint - lCumulativePreviousPoint)
00111             * (iValue - lValuePreviousPoint)
00112             / (lValueCurrentPoint - lValuePreviousPoint);
00113             return 1 - oCumulativeProbability;
00114         }
00115     public:
00116         // //////////////// Business Methods ////////////////
00131         const double getDerivativeValue(const T iKey) const{
00132
00133             // Find the first key value greater or equal to iKey.
00134             unsigned int idx = 0;
00135             for (; idx < _size; ++idx) {
00136                 if (_valueArray.at(idx) > iKey) {
00137                     break;
00138                 }
00139             }
00140             assert (idx != 0);
00141             assert (idx != _size);
00142             //
00143             const stdair::Probability_T& lCumulativeCurrentPoint =
DictionaryManager::keyToValue (_cumulativeDistribution.at(idx));
00144             const T& lValueCurrentPoint = _valueArray.at(idx);
00145             //
00146             const stdair::Probability_T& lCumulativePreviousPoint =
DictionaryManager::keyToValue (_cumulativeDistribution.at(idx-1));
00147             const T& lValuePreviousPoint = _valueArray.at(idx-1);
00148             assert (lValueCurrentPoint != lValuePreviousPoint);
00149             const double oValue= (lCumulativeCurrentPoint - lCumulativePreviousPoint)
00150             / (lValueCurrentPoint - lValuePreviousPoint);
00151             return oValue;
00152         }
00153     const T getUpperBound (const T iKey) const {
00154         // Find the first key value greater or equal to iKey.
00155         unsigned int idx = 0;
00156         for (; idx < _size; ++idx) {
00157             if (_valueArray.at(idx) > iKey) {
00158                 break;
00159             }
00160         }
00161         assert (idx != 0);

```



```

00172     assert (idx != _size);
00173
00174     return _valueArray.at (idx);
00175 }
00176
00177 public:
00178 // //////////// Display Support Methods ////////////
00182 const std::string displayCumulativeDistribution() const {
00183     std::ostream oStr;
00184
00185     for (unsigned int idx = 0; idx < _size; ++idx) {
00186         if (idx != 0) {
00187             oStr << ", ";
00188         }
00189
00190         const stdair::Probability_T& lProbability =
00191             DictionaryManager::keyToValue (_cumulativeDistribution.at(idx));
00192
00193         oStr << _valueArray.at(idx) << ":" << lProbability;
00194     }
00195     return oStr.str();
00196 }
00197
00198
00199 public:
00200 // //////////// Constructors and destructors ////////////
00204 ContinuousAttributeLite (const ContinuousDistribution_T& iValueMap)
00205     : _size (iValueMap.size()) {
00206     init (iValueMap);
00207 }
00208
00212 ContinuousAttributeLite (const
ContinuousAttributeLite& iCAL)
00213     : _size (iCAL._size),
00214       _cumulativeDistribution (iCAL._cumulativeDistribution),
00215       _valueArray (iCAL._valueArray) {
00216 }
00217
00221 ContinuousAttributeLite& operator= (const
ContinuousAttributeLite& iCAL) {
00222     _size = iCAL._size;
00223     _cumulativeDistribution = iCAL._cumulativeDistribution;
00224     _valueArray = iCAL._valueArray;
00225     return *this;
00226 }
00227
00231 virtual ~ContinuousAttributeLite() {
00232 }
00233
00234 private:
00238 ContinuousAttributeLite() : _size(1) {
00239 }
00240
00245 void init (const ContinuousDistribution_T& iValueMap) {
00246     //
00247     const unsigned int lSize = iValueMap.size();
00248     _cumulativeDistribution.reserve (lSize);
00249     _valueArray.reserve (lSize);
00250
00251     // Browse the map to retrieve the values and cumulative probabilities.
00252     for (typename ContinuousDistribution_T::const_iterator it =
00253         iValueMap.begin(); it != iValueMap.end(); ++it) {
00254
00255         const T& attributeValue = it->first;
00256         const DictionaryKey_T& lKey =
DictionaryManager::valueToKey (it->second);
00257
00258         // Build the two arrays.
00259         _cumulativeDistribution.push_back (lKey);
00260         _valueArray.push_back (attributeValue);
00261     }
00262 }
00263
00264
00265 private:
00266 // //////////// Attributes ////////////
00270 unsigned int _size;
00271
00275 std::vector<DictionaryKey_T> _cumulativeDistribution;
00276
00280 std::vector<T> _valueArray;
00281 };
00282
00283 }
00284 #endif // __STDAIR_BAS_CONTINUOUSATTRIBUTE_LITE_HPP

```

## 33.65 stdair/basic/DemandGenerationMethod.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/DemandGenerationMethod.hpp>
```

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.66 DemandGenerationMethod.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/stdair_exceptions.hpp>
00009 #include <stdair/basic/DemandGenerationMethod.hpp>
00010
00011 namespace stdair {
00012
00013 // //////////////////////////////////////
00014 const std::string DemandGenerationMethod::_labels[LAST_VALUE] =
00015     { "PoissonProcess", "StatisticsOrder" };
00016
00017 // //////////////////////////////////////
00018 const char DemandGenerationMethod::_methodLabels[LAST_VALUE] = { 'P', 'S' };
00019
00020
00021 // //////////////////////////////////////
00022 DemandGenerationMethod::DemandGenerationMethod() : _method (LAST_VALUE) {
00023     assert (false);
00024 }
00025
00026 // //////////////////////////////////////
00027 DemandGenerationMethod::
00028 DemandGenerationMethod (const DemandGenerationMethod& iDemandGenerationMethod)
00029     : _method (iDemandGenerationMethod._method) {
00030 }
00031
00032 // //////////////////////////////////////
00033 DemandGenerationMethod::
00034 DemandGenerationMethod (const EN_DemandGenerationMethod& iDemandGenerationMethod
00035 )
00036     : _method (iDemandGenerationMethod) {
00037 }
00038
00039 DemandGenerationMethod::EN_DemandGenerationMethod
00040 DemandGenerationMethod::getMethod (const char iMethodChar) {
00041     EN_DemandGenerationMethod oMethod;
00042     switch (iMethodChar) {
00043     case 'P': oMethod = POI_PRO; break;
00044     case 'S': oMethod = STA_ORD; break;
00045     default: oMethod = LAST_VALUE; break;
00046     }
00047
00048     if (oMethod == LAST_VALUE) {
00049         const std::string& lLabels = describeLabels();
00050         std::ostringstream oMessage;
00051         oMessage << "The demand (booking request) generation method '"
00052             << iMethodChar
00053             << "' is not known. Known demand (booking request) generation "
00054             << "methods: " << lLabels;
00055         throw CodeConversionException (oMessage.str());
00056     }
00057
00058     return oMethod;
00059 }
00060
00061 // //////////////////////////////////////
```

```

00062 DemandGenerationMethod::DemandGenerationMethod (const char iMethodChar)
00063 : _method (getMethod (iMethodChar)) {
00064 }
00065
00066 // //////////////////////////////////////
00067 DemandGenerationMethod::
00068 DemandGenerationMethod (const std::string& iMethodStr) {
00069 //
00070     const size_t lSize = iMethodStr.size();
00071     assert (lSize == 1);
00072     const char lMethodChar = iMethodStr[0];
00073     _method = getMethod (lMethodChar);
00074 }
00075
00076 // //////////////////////////////////////
00077 const std::string& DemandGenerationMethod::
00078 getLabel (const EN_DemandGenerationMethod& iMethod) {
00079     return _labels[iMethod];
00080 }
00081
00082 // //////////////////////////////////////
00083 char DemandGenerationMethod::
00084 getMethodLabel (const EN_DemandGenerationMethod& iMethod) {
00085     return _methodLabels[iMethod];
00086 }
00087
00088 // //////////////////////////////////////
00089 std::string DemandGenerationMethod::
00090 getMethodLabelAsString (const
00091 EN_DemandGenerationMethod& iMethod) {
00092     std::ostringstream ostr;
00093     ostr << _methodLabels[iMethod];
00094     return ostr.str();
00095 }
00096
00097 // //////////////////////////////////////
00098 std::string DemandGenerationMethod::describeLabels() {
00099     std::ostringstream ostr;
00100     for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
00101         if (idx != 0) {
00102             ostr << ", ";
00103         }
00104         ostr << _labels[idx];
00105     }
00106     return ostr.str();
00107 }
00108
00109 // //////////////////////////////////////
00110 DemandGenerationMethod::EN_DemandGenerationMethod
00111 DemandGenerationMethod::getMethod() const {
00112     return _method;
00113 }
00114
00115 // //////////////////////////////////////
00116 char DemandGenerationMethod::getMethodAsChar() const {
00117     const char oMethodChar = _methodLabels[_method];
00118     return oMethodChar;
00119 }
00120
00121 // //////////////////////////////////////
00122 std::string DemandGenerationMethod::getMethodAsString() const {
00123     std::ostringstream ostr;
00124     ostr << _methodLabels[_method];
00125     return ostr.str();
00126 }
00127
00128 // //////////////////////////////////////
00129 const std::string DemandGenerationMethod::describe() const {
00130     std::ostringstream ostr;
00131     ostr << _labels[_method];
00132     return ostr.str();
00133 }
00134
00135 // //////////////////////////////////////
00136 bool DemandGenerationMethod::
00137 operator== (const EN_DemandGenerationMethod& iMethod) const {
00138     return (_method == iMethod);
00139 }
00140 }

```

### 33.67 stdair/basic/DemandGenerationMethod.hpp File Reference

```
#include <string>
```

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

## Classes

- struct [stdair::DemandGenerationMethod](#)  
*Enumeration of demand (booking request) generation methods.*

## Namespaces

- [stdair](#)  
*Handle on the StdAir library context.*

## 33.68 DemandGenerationMethod.hpp

```
00001 #ifndef __STDAIR_BAS_DEMANDGENERATIONMETHOD_HPP
00002 #define __STDAIR_BAS_DEMANDGENERATIONMETHOD_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/basic/StructAbstract.hpp>
00011
00012 namespace stdair {
00013
00017     struct DemandGenerationMethod : public StructAbstract {
00018     public:
00019         typedef enum {
00020             POI_PRO = 0,
00021             STA_ORD,
00022             LAST_VALUE
00023         } EN_DemandGenerationMethod;
00024
00028         static const std::string& getLabel (const EN_DemandGenerationMethod&);
00029
00033         static EN_DemandGenerationMethod getMethod (const char);
00034
00038         static char getMethodLabel (const EN_DemandGenerationMethod&);
00039
00043         static std::string getMethodLabelAsString (const
00044 EN_DemandGenerationMethod&);
00048
00048         static std::string describeLabels();
00049
00053         EN_DemandGenerationMethod getMethod() const;
00054
00058         char getMethodAsChar() const;
00059
00063         std::string getMethodAsString() const;
00064
00069         const std::string describe() const;
00070
00071     public:
00075         bool operator== (const EN_DemandGenerationMethod&) const;
00076
00077     public:
00081         DemandGenerationMethod (const
00082 EN_DemandGenerationMethod&);
00085         DemandGenerationMethod (const char iMethod);
00089         DemandGenerationMethod (const std::string& iMethod);
00093         DemandGenerationMethod (const DemandGenerationMethod&);
00094
00095     private:
00099         DemandGenerationMethod();
00100
00101
00102     private:
00106         static const std::string _labels[LAST_VALUE];
00110         static const char _methodLabels[LAST_VALUE];
00111
00112     private:
00113         // ////////// Attributes //////////
00117         EN_DemandGenerationMethod _method;
```

```

00118     };
00119
00120 }
00121 #endif // __STDAIR_BAS_DEMANDGENERATIONMETHOD_HPP

```

### 33.69 stdair/basic/DictionaryManager.cpp File Reference

```

#include <stdair/basic/DictionaryManager.hpp>
#include <stdair/basic/BasConst_General.hpp>

```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.70 DictionaryManager.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // StdAir
00005 #include <stdair/basic/DictionaryManager.hpp>
00006 #include <stdair/basic/BasConst_General.hpp>
00007
00008 namespace stdair {
00009
00010 // //////////////////////////////////////
00011 const stdair::Probability_T DictionaryManager::
00012 keyToValue (const DictionaryKey_T iKey) {
00013     const float lValue =
00014         static_cast<float> (iKey) / DEFAULT_NUMBER_OF_SUBDIVISIONS;
00015     const stdair::Probability_T lProbability (lValue);
00016     return lProbability;
00017 }
00018
00019 // //////////////////////////////////////
00020 const DictionaryKey_T DictionaryManager::
00021 valueToKey (const stdair::Probability_T iValue) {
00022     const unsigned short lValueMultipliedByThousand =
00023         static_cast<unsigned short> (iValue) * DEFAULT_NUMBER_OF_SUBDIVISIONS;
00024     const DictionaryKey_T lDictionaryKey (lValueMultipliedByThousand);
00025     return lDictionaryKey;
00026 }
00027
00028 }

```

### 33.71 stdair/basic/DictionaryManager.hpp File Reference

```

#include <stdair/stdair_maths_types.hpp>

```

#### Classes

- class [stdair::DictionaryManager](#)

*Class wrapper of dictionary business methods.*

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## Typedefs

- typedef unsigned short [stdair::DictionaryKey\\_T](#)

## 33.72 DictionaryManager.hpp

```

00001 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00002 #ifndef __STDAIR_BASIC_DICTIONARYMANAGER_HPP
00003 #define __STDAIR_BASIC_DICTIONARYMANAGER_HPP
00004
00005 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00006 // Import section
00007 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00008 // StdAir
00009 #include <stdair/stdair_maths_types.hpp>
00010
00011 namespace stdair {
00012
00013     ////////////////////////////////////////////////////////////////// Type definitions //////////////////////////////////////////////////////////////////
00017     typedef unsigned short DictionaryKey_T;
00018
00022     class DictionaryManager {
00023     public:
00024         ////////////////////////////////////////////////////////////////// Business methods //////////////////////////////////////////////////////////////////
00028         static const stdair::Probability_T keyToValue (const DictionaryKey_T);
00029
00033         static const DictionaryKey_T valueToKey (const
stdair::Probability_T);
00034     };
00035 }
00036 #endif // __STDAIR_BASIC_DICTIONARYMANAGER_HPP

```

## 33.73 stdair/basic/EventType.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/EventType.hpp>

```

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.74 EventType.cpp

```

00001 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00002 // Import section
00003 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/stdair_exceptions.hpp>
00009 #include <stdair/basic/EventType.hpp>
00010
00011 namespace stdair {
00012
00013     ////////////////////////////////////////////////////////////////// Type definitions //////////////////////////////////////////////////////////////////
00014     const std::string EventType::_labels[LAST_VALUE] =
00015         { "BookingRequest", "Cancellation", "OptimisationNotificationForFlightDate",
00016           "OptimisationNotificationForNetwork", "ScheduleChange", "Snapshot",
00017           "RevenueManagement", "BreakPoint" };
00018
00019     ////////////////////////////////////////////////////////////////// Business methods //////////////////////////////////////////////////////////////////
00020     const char EventType::_typeLabels[LAST_VALUE] = { 'B', 'X', 'F', 'N', 'C', 'S', 'R', 'P' };
00021
00022
00023
00024 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////

```

```

00025 EventType::EventType()
00026 : _type (LAST_VALUE) {
00027     assert (false);
00028 }
00029
00030 // //////////////////////////////////////
00031 EventType::EventType (const EventType& iEventType)
00032 : _type (iEventType._type) {
00033 }
00034
00035 // //////////////////////////////////////
00036 EventType::EventType (const EN_EventType& iEventType)
00037 : _type (iEventType) {
00038 }
00039
00040 // //////////////////////////////////////
00041 EventType::EventType (const char iType) {
00042     switch (iType) {
00043         case 'B': _type = BKG_REQ; break;
00044         case 'X': _type = CX; break;
00045         case 'F': _type = OPT_NOT_4_FD; break;
00046         case 'N': _type = OPT_NOT_4_NET; break;
00047         case 'C': _type = SKD_CHG; break;
00048         case 'S': _type = SNAPSHOT; break;
00049         case 'R': _type = RM; break;
00050         case 'P': _type = BRK_PT; break;
00051         default: _type = LAST_VALUE; break;
00052     }
00053
00054     if (_type == LAST_VALUE) {
00055         const std::string& lLabels = describeLabels();
00056         std::ostringstream oMessage;
00057         oMessage << "The event type '" << iType
00058             << "' is not known. Known event types: " << lLabels;
00059         throw CodeConversionException (oMessage.str());
00060     }
00061 }
00062
00063 // //////////////////////////////////////
00064 EventType::EventType (const std::string& iTypeStr) {
00065     for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
00066         if (iTypeStr.compare(_labels[idx]) == 0) {
00067             _type = static_cast<EN_EventType> (idx);
00068             break;
00069         } else {
00070             _type = LAST_VALUE;
00071         }
00072     }
00073     if (_type == LAST_VALUE) {
00074         const std::string& lLabels = describeLabels();
00075         std::ostringstream oMessage;
00076         oMessage << "The event type '" << iTypeStr
00077             << "' is not known. Known event types: " << lLabels;
00078         throw CodeConversionException (oMessage.str());
00079     }
00080 }
00081
00082 // //////////////////////////////////////
00083 const std::string& EventType::getLabel (const EN_EventType& iType) {
00084     return _labels[iType];
00085 }
00086
00087 // //////////////////////////////////////
00088 char EventType::getTypeLabel (const EN_EventType& iType) {
00089     return _typeLabels[iType];
00090 }
00091
00092 // //////////////////////////////////////
00093 std::string EventType::getTypeLabelAsString (const
EN_EventType& iType) {
00094     std::ostringstream ostr;
00095     ostr << _typeLabels[iType];
00096     return ostr.str();
00097 }
00098
00099 // //////////////////////////////////////
00100 std::string EventType::describeLabels() {
00101     std::ostringstream ostr;
00102     for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
00103         if (idx != 0) {
00104             ostr << ", ";
00105         }
00106         ostr << _labels[idx];
00107     }
00108     return ostr.str();
00109 }
00110

```

```

00111 // //////////////////////////////////////
00112 EventType::EN_EventType EventType::getType() const {
00113     return _type;
00114 }
00115
00116 // //////////////////////////////////////
00117 std::string EventType::getTypeAsString() const {
00118     std::ostringstream ostr;
00119     ostr << _typeLabels[_type];
00120     return ostr.str();
00121 }
00122
00123 // //////////////////////////////////////
00124 const std::string EventType::describe() const {
00125     std::ostringstream ostr;
00126     ostr << _labels[_type];
00127     return ostr.str();
00128 }
00129
00130 // //////////////////////////////////////
00131 bool EventType::operator== (const EN_EventType& iType) const {
00132     return (_type == iType);
00133 }
00134
00135 }

```

### 33.75 stdair/basic/EventType.hpp File Reference

```

#include <string>
#include <stdair/basic/StructAbstract.hpp>

```

#### Classes

- struct [stdair::EventType](#)

#### Namespaces

- [stdair](#)  
*Handle on the StdAir library context.*

### 33.76 EventType.hpp

```

00001 #ifndef __STDAIR_BAS_EVENTTYPE_HPP
00002 #define __STDAIR_BAS_EVENTTYPE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/basic/StructAbstract.hpp>
00011
00012 namespace stdair {
00013
00015     struct EventType : public StructAbstract {
00016     public:
00017         typedef enum {
00018             BKG_REQ = 0,
00019             CX,
00020             OPT_NOT_4_FD,
00021             OPT_NOT_4_NET,
00022             SKD_CHG,
00023             SNAPSHOT,
00024             RM,
00025             BRK_PT,
00026             LAST_VALUE
00027         } EN_EventType;
00028
00034         static const std::string& getLabel (const EN_EventType&);
00035
00039         static char getTypeLabel (const EN_EventType&);

```



```

00040
00044     static std::string getTypeLabelAsString (const
EN_EventType&);
00045
00047     static std::string describeLabels();
00048
00050     EN_EventType getType() const;
00051
00055     std::string getTypeAsString() const;
00056
00062     const std::string describe() const;
00063
00064 public:
00066     bool operator== (const EN_EventType&) const;
00067
00068 public:
00070     EventType (const EN_EventType&);
00072     EventType (const char iType);
00074     EventType (const std::string& iTypeStr);
00076     EventType (const EventType&);
00077
00078 private:
00080     EventType();
00081
00082
00083 private:
00085     static const std::string _labels[LAST_VALUE];
00087     static const char _typeLabels[LAST_VALUE];
00088
00089
00090 private:
00091     // ////////// Attributes //////////
00093     EN_EventType _type;
00094 };
00095
00096 }
00097 #endif // __STDAIR_BAS_EVENTTYPE_HPP

```

### 33.77 stdair/basic/float\_utils.hpp File Reference

```
#include <stdair/basic/float_utils_google.hpp>
```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.78 float\_utils.hpp

```

00001 #ifndef __STDAIR_BAS_FLOAT_UTILS_HPP
00002 #define __STDAIR_BAS_FLOAT_UTILS_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/basic/float_utils_google.hpp>
00009
00010 namespace stdair {
00011
00023 }
00024 #endif // __STDAIR_BAS_FLOAT_UTILS_HPP

```

### 33.79 stdair/basic/float\_utils\_google.hpp File Reference

#### Classes

- class [TypeWithSize< size >](#)
- class [TypeWithSize< 4 >](#)
- class [TypeWithSize< 8 >](#)

- class `FloatingPoint`< `RawType` >

### 33.80 float\_utils\_google.hpp

```

00001 #ifndef __STDAIR_BAS_FLOAT_UTILS_GOOGLE_HPP
00002 #define __STDAIR_BAS_FLOAT_UTILS_GOOGLE_HPP
00003
00004 // Redistribution and use in source and binary forms, with or without
00005 // modification, are permitted provided that the following conditions are
00006 // met:
00007 //
00008 //     * Redistributions of source code must retain the above copyright
00009 //     notice, this list of conditions and the following disclaimer.
00010 //     * Redistributions in binary form must reproduce the above
00011 //     copyright notice, this list of conditions and the following disclaimer
00012 //     in the documentation and/or other materials provided with the
00013 //     distribution.
00014 //     * Neither the name of Google Inc. nor the names of its
00015 //     contributors may be used to endorse or promote products derived from
00016 //     this software without specific prior written permission.
00017 //
00018 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00019 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00020 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00021 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
00022 // OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00023 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00024 // LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00025 // DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00026 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00027 // (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00028 // OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00029 //
00030 // Authors: wan@google.com (Zhanyong Wan), eefacm@gmail.com (Sean McAfee)
00031 //
00032 // The Google C++ Testing Framework (Google Test)
00033
00034
00035 // This template class serves as a compile-time function from size to
00036 // type. It maps a size in bytes to a primitive type with that
00037 // size. e.g.
00038 //
00039 //     TypeWithSize<4>::UInt
00040 //
00041 // is typedef-ed to be unsigned int (unsigned integer made up of 4
00042 // bytes).
00043 //
00044 // Such functionality should belong to STL, but I cannot find it
00045 // there.
00046 //
00047 // Google Test uses this class in the implementation of floating-point
00048 // comparison.
00049 //
00050 // For now it only handles UInt (unsigned int) as that's all Google Test
00051 // needs. Other types can be easily added in the future if need
00052 // arises.
00053 template <size_t size>
00054 class TypeWithSize {
00055 public:
00056     // This prevents the user from using TypeWithSize<N> with incorrect
00057     // values of N.
00058     typedef void UInt;
00059 };
00060
00061 // The specialization for size 4.
00062 template <>
00063 class TypeWithSize<4> {
00064 public:
00065     // unsigned int has size 4 in both gcc and MSVC.
00066     //
00067     // As base/basictypes.h doesn't compile on Windows, we cannot use
00068     // uint32, uint64, and etc here.
00069     typedef int Int;
00070     typedef unsigned int UInt;
00071 };
00072
00073 // The specialization for size 8.
00074 template <>
00075 class TypeWithSize<8> {
00076 public:
00077     #if GTEST_OS_WINDOWS
00078         typedef __int64 Int;
00079         typedef unsigned __int64 UInt;
00080     #else

```

```

00081     typedef long long Int; // NOLINT
00082     typedef unsigned long long UInt; // NOLINT
00083 #endif // GTEST_OS_WINDOWS
00084 };
00085
00086
00087 // This template class represents an IEEE floating-point number
00088 // (either single-precision or double-precision, depending on the
00089 // template parameters).
00090 //
00091 // The purpose of this class is to do more sophisticated number
00092 // comparison. (Due to round-off error, etc, it's very unlikely that
00093 // two floating-points will be equal exactly. Hence a naive
00094 // comparison by the == operation often doesn't work.)
00095 //
00096 // Format of IEEE floating-point:
00097 //
00098 //     The most-significant bit being the leftmost, an IEEE
00099 //     floating-point looks like
00100 //
00101 //         sign_bit exponent_bits fraction_bits
00102 //
00103 //     Here, sign_bit is a single bit that designates the sign of the
00104 //     number.
00105 //
00106 //     For float, there are 8 exponent bits and 23 fraction bits.
00107 //
00108 //     For double, there are 11 exponent bits and 52 fraction bits.
00109 //
00110 //     More details can be found at
00111 //     http://en.wikipedia.org/wiki/IEEE\_floating-point\_standard.
00112 //
00113 // Template parameter:
00114 //
00115 //     RawType: the raw floating-point type (either float or double)
00116 template <typename RawType>
00117 class FloatingPoint {
00118 public:
00119     // Defines the unsigned integer type that has the same size as the
00120     // floating point number.
00121     typedef typename TypeWithSize<sizeof(RawType)>::UInt
Bits;
00122
00123     // Constants.
00124
00125     // # of bits in a number.
00126     static const size_t kBitCount = 8*sizeof(RawType);
00127
00128     // # of fraction bits in a number.
00129     static const size_t kFractionBitCount =
00130         std::numeric_limits<RawType>::digits - 1;
00131
00132     // # of exponent bits in a number.
00133     static const size_t kExponentBitCount = kBitCount - 1 -
kFractionBitCount;
00134
00135     // The mask for the sign bit.
00136     static const Bits kSignBitMask = static_cast<Bits>(1) << (kBitCount - 1);
00137
00138     // The mask for the fraction bits.
00139     static const Bits kFractionBitMask =
00140         ~static_cast<Bits>(0) >> (kExponentBitCount + 1);
00141
00142     // The mask for the exponent bits.
00143     static const Bits kExponentBitMask = ~(kSignBitMask |
kFractionBitMask);
00144
00145     // How many ULP's (Units in the Last Place) we want to tolerate when
00146     // comparing two numbers. The larger the value, the more error we
00147     // allow. A 0 value means that two numbers must be exactly the same
00148     // to be considered equal.
00149     //
00150     // The maximum error of a single floating-point operation is 0.5
00151     // units in the last place. On Intel CPU's, all floating-point
00152     // calculations are done with 80-bit precision, while double has 64
00153     // bits. Therefore, 4 should be enough for ordinary use.
00154     //
00155     // See the following article for more details on ULP:
00156     // http://www.cygnum-software.com/papers/comparingfloats/comparingfloats.htm.
00157     static const size_t kMaxUlp = 4;
00158
00159     // Constructs a FloatingPoint from a raw floating-point number.
00160     //
00161     // On an Intel CPU, passing a non-normalized NAN (Not a Number)
00162     // around may change its bits, although the new value is guaranteed
00163     // to be also a NAN. Therefore, don't expect this constructor to
00164     // preserve the bits in x when x is a NAN.

```

```

00165     explicit FloatingPoint(const RawType& x) { u_.value_ = x; }
00166
00167     // Static methods
00168
00169     // Reinterprets a bit pattern as a floating-point number.
00170     //
00171     // This function is needed to test the AlmostEquals() method.
00172     static RawType ReinterpretBits(const Bits bits) {
00173         FloatingPoint fp(0);
00174         fp.u_.bits_ = bits;
00175         return fp.u_.value_;
00176     }
00177
00178     // Returns the floating-point number that represent positive infinity.
00179     static RawType Infinity() {
00180         return ReinterpretBits(kExponentBitMask);
00181     }
00182
00183     // Non-static methods
00184
00185     // Returns the bits that represents this number.
00186     const Bits &bits() const { return u_.bits_; }
00187
00188     // Returns the exponent bits of this number.
00189     Bits exponent_bits() const { return kExponentBitMask & u_.bits_; }
00190
00191     // Returns the fraction bits of this number.
00192     Bits fraction_bits() const { return kFractionBitMask & u_.bits_; }
00193
00194     // Returns the sign bit of this number.
00195     Bits sign_bit() const { return kSignBitMask & u_.bits_; }
00196
00197     // Returns true iff this is NAN (not a number).
00198     bool is_nan() const {
00199         // It's a NAN if the exponent bits are all ones and the fraction
00200         // bits are not entirely zeros.
00201         return (exponent_bits() == kExponentBitMask) && (fraction_bits() != 0);
00202     }
00203
00204     // Returns true iff this number is at most kMaxUlp's away from
00205     // rhs. In particular, this function:
00206     //
00207     // - returns false if either number is (or both are) NAN.
00208     // - treats really large numbers as almost equal to infinity.
00209     // - thinks +0.0 and -0.0 are 0 DLP's apart.
00210     bool AlmostEquals(const FloatingPoint& rhs) const {
00211         // The IEEE standard says that any comparison operation involving
00212         // a NAN must return false.
00213         if (is_nan() || rhs.is_nan()) return false;
00214
00215         return DistanceBetweenSignAndMagnitudeNumbers(u_.bits_, rhs.u_.bits_)
00216             <= kMaxUlp's;
00217     }
00218
00219 private:
00220     // The data type used to store the actual floating-point number.
00221     union FloatingPointUnion {
00222         RawType value_; // The raw floating-point number.
00223         Bits bits_;     // The bits that represent the number.
00224     };
00225
00226     // Converts an integer from the sign-and-magnitude representation to
00227     // the biased representation. More precisely, let N be 2 to the
00228     // power of (kBitCount - 1), an integer x is represented by the
00229     // unsigned number x + N.
00230     //
00231     // For instance,
00232     //
00233     // -N + 1 (the most negative number representable using
00234     // sign-and-magnitude) is represented by 1;
00235     // 0 is represented by N; and
00236     // N - 1 (the biggest number representable using
00237     // sign-and-magnitude) is represented by 2N - 1.
00238     //
00239     // Read http://en.wikipedia.org/wiki/Signed\_number\_representations
00240     // for more details on signed number representations.
00241     static Bits SignAndMagnitudeToBiased(const Bits &sam) {
00242         if (kSignBitMask & sam) {
00243             // sam represents a negative number.
00244             return ~sam + 1;
00245         } else {
00246             // sam represents a positive number.
00247             return kSignBitMask | sam;
00248         }
00249     }
00250
00251     // Given two numbers in the sign-and-magnitude representation,

```

```

00252 // returns the distance between them as an unsigned number.
00253 static Bits DistanceBetweenSignAndMagnitudeNumbers(const Bits &sam1,
00254                                                    const Bits &sam2) {
00255     const Bits biased1 = SignAndMagnitudeToBiased(sam1);
00256     const Bits biased2 = SignAndMagnitudeToBiased(sam2);
00257     return (biased1 >= biased2) ? (biased1 - biased2) : (biased2 - biased1);
00258 }
00259
00260 FloatingPointUnion u_;
00261 };
00262
00263 #endif // __STDAIR_BASIC_FLOAT_UTILS_GOOGLE_HPP

```

### 33.81 stdair/basic/ForecastingMethod.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/ForecastingMethod.hpp>

```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.82 ForecastingMethod.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/stdair_exceptions.hpp>
00009 #include <stdair/basic/ForecastingMethod.hpp>
00010
00011 namespace stdair {
00012
00013 // //////////////////////////////////////
00014 const std::string ForecastingMethod::_labels[LAST_VALUE] =
00015     { "Q Forecasting", "Hybrid Forecasting", "Old QFF", "New QFF",
00016       "Based Forecasting" };
00017
00018 // //////////////////////////////////////
00019 const char ForecastingMethod::_methodLabels[LAST_VALUE] = { 'Q', 'H', 'O', 'N', 'B' };
00020
00021
00022 // //////////////////////////////////////
00023 ForecastingMethod::ForecastingMethod()
00024     : _method (LAST_VALUE) {
00025     assert (false);
00026 }
00027
00028 // //////////////////////////////////////
00029 ForecastingMethod::
00030 ForecastingMethod (const ForecastingMethod& iForecastingMethod)
00031     : _method (iForecastingMethod._method) {
00032 }
00033
00034 // //////////////////////////////////////
00035 ForecastingMethod::
00036 ForecastingMethod (const EN_ForecastingMethod& iForecastingMethod)
00037     : _method (iForecastingMethod) {
00038 }
00039
00040 // //////////////////////////////////////
00041 ForecastingMethod::ForecastingMethod (const char iMethod) {
00042     switch (iMethod) {
00043     case 'Q': _method = Q_FORECASTING; break;
00044     case 'H': _method = HYBRID_FORECASTING; break;
00045     case 'O': _method = OLD_QFF; break;
00046     case 'N': _method = NEW_QFF; break;
00047

```

```

00048     case 'B': _method = BASED_FORECASTING; break;
00049     default: _method = LAST_VALUE; break;
00050 }
00051
00052     if (_method == LAST_VALUE) {
00053         const std::string& lLabels = describeLabels();
00054         std::ostringstream oMessage;
00055         oMessage << "The forecasting method '" << iMethod
00056             << "' is not known. Known forecasting methods: " << lLabels;
00057         throw CodeConversionException (oMessage.str());
00058     }
00059 }
00060
00061 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00062 const std::string& ForecastingMethod::
00063 getLabel (const EN_ForecastingMethod& iMethod) {
00064     return _labels[iMethod];
00065 }
00066
00067 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00068 char ForecastingMethod::getMethodLabel (const
EN_ForecastingMethod& iMethod) {
00069     return _methodLabels[iMethod];
00070 }
00071
00072 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00073 std::string ForecastingMethod::
00074 getMethodLabelAsString (const EN_ForecastingMethod& iMethod)
00075 {
00076     std::ostringstream oStr;
00077     oStr << _methodLabels[iMethod];
00078     return oStr.str();
00079 }
00080
00081 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00082 std::string ForecastingMethod::describeLabels() {
00083     std::ostringstream ostr;
00084     for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
00085         if (idx != 0) {
00086             ostr << ", ";
00087         }
00088         ostr << _labels[idx] << " (" << _methodLabels[idx] << ")";
00089     }
00090     return ostr.str();
00091 }
00092
00093 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00094 ForecastingMethod::EN_ForecastingMethod
ForecastingMethod::getMethod() const {
00095     return _method;
00096 }
00097
00098 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00099 std::string ForecastingMethod::getMethodAsString() const {
00100     std::ostringstream oStr;
00101     oStr << _methodLabels[_method];
00102     return oStr.str();
00103 }
00104
00105 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00106 const std::string ForecastingMethod::describe() const {
00107     std::ostringstream ostr;
00108     ostr << _labels[_method];
00109     return ostr.str();
00110 }
00111
00112 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00113 bool ForecastingMethod::
operator== (const EN_ForecastingMethod& iMethod) const {
00114     return (_method == iMethod);
00115 }
00116
00117 }

```

### 33.83 stdair/basic/ForecastingMethod.hpp File Reference

```

#include <string>
#include <stdair/basic/StructAbstract.hpp>

```

## Classes

- struct `stdair::ForecastingMethod`

## Namespaces

- `stdair`

*Handle on the StdAir library context.*

## 33.84 ForecastingMethod.hpp

```

00001 #ifndef __STDAIR_BAS_FORECASTINGMETHOD_HPP
00002 #define __STDAIR_BAS_FORECASTINGMETHOD_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/basic/StructAbstract.hpp>
00011
00012 namespace stdair {
00013
00014     struct ForecastingMethod : public StructAbstract {
00015     public:
00016         typedef enum {
00017             Q_FORECASTING = 0,
00018             HYBRID_FORECASTING,
00019             OLD_QFF,
00020             NEW_QFF,
00021             BASED_FORECASTING,
00022             LAST_VALUE
00023         } EN_ForecastingMethod;
00024
00025         static const std::string& getLabel (const EN_ForecastingMethod&);
00026
00027         static char getMethodLabel (const EN_ForecastingMethod&);
00028
00029         static std::string getMethodLabelAsString (const
00030             EN_ForecastingMethod&);
00031
00032         static std::string describeLabels();
00033
00034         EN_ForecastingMethod getMethod() const;
00035
00036         std::string getMethodAsString() const;
00037
00038         const std::string describe() const;
00039
00040     public:
00041         bool operator== (const EN_ForecastingMethod&) const;
00042
00043     public:
00044         ForecastingMethod (const EN_ForecastingMethod&);
00045         ForecastingMethod (const char iMethod);
00046         ForecastingMethod (const ForecastingMethod&);
00047
00048     private:
00049         ForecastingMethod();
00050
00051     private:
00052         static const std::string _labels[LAST_VALUE];
00053         static const char _methodLabels[LAST_VALUE];
00054
00055     private:
00056         // ////////// Attributes //////////
00057         EN_ForecastingMethod _method;
00058     };
00059
00060 }
00061 #endif // __STDAIR_BAS_FORECASTINGMETHOD_HPP

```

## 33.85 stdair/basic/JSonCommand.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/JSonCommand.hpp>
```

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.86 JSonCommand.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/stdair_exceptions.hpp>
00009 #include <stdair/basic/JSonCommand.hpp>
00010
00011 namespace stdair {
00012
00013 // //////////////////////////////////////
00014 const std::string JSonCommand::_labels[LAST_VALUE] =
00015     { "list", "flight_date", "event_list", "break_point", "run", "reset", "status", "config"};
00016
00017 // //////////////////////////////////////
00018 JSonCommand::JSonCommand()
00019     : _command (LAST_VALUE) {
00020     assert (false);
00021 }
00022
00023 // //////////////////////////////////////
00024 JSonCommand::
00025 JSonCommand (const JSonCommand& iJSonCommand)
00026     : _command (iJSonCommand._command) {
00027 }
00028
00029 // //////////////////////////////////////
00030 JSonCommand::EN_JSonCommand
00031 JSonCommand::getCommand (const std::string& iCommandStr) {
00032
00033     EN_JSonCommand oJSonCommand;
00034     if (iCommandStr == "list") {
00035         oJSonCommand = LIST;
00036     } else if (iCommandStr == "flight_date") {
00037         oJSonCommand = FLIGHT_DATE;
00038     } else if (iCommandStr == "event_list") {
00039         oJSonCommand = EVENT_LIST;
00040     } else if (iCommandStr == "break_point") {
00041         oJSonCommand = BREAK_POINT;
00042     } else if (iCommandStr == "run") {
00043         oJSonCommand = RUN;
00044     } else if (iCommandStr == "reset") {
00045         oJSonCommand = RESET;
00046     } else if (iCommandStr == "status") {
00047         oJSonCommand = STATUS;
00048     } else if (iCommandStr == "config") {
00049         oJSonCommand = CONFIG;
00050     } else {
00051         oJSonCommand = LAST_VALUE;
00052     }
00053
00054     if (oJSonCommand == LAST_VALUE) {
00055         const std::string& lLabels = describeLabels();
00056         std::ostringstream oMessage;
00057         oMessage << "The JSON command '" << iCommandStr
00058             << "' is not known. Known forecasting commands: " << lLabels;
00059         throw CodeConversionException (oMessage.str());
00060     }
00061
00062     return oJSonCommand;
00063 }
```



```

00063     }
00064
00065     // //////////////////////////////////////
00066     std::string JSonCommand::getLabel(const EN_JSonCommand& iCommand) {
00067         return _labels[iCommand];
00068     }
00069
00070     // //////////////////////////////////////
00071     JSonCommand::JSonCommand (const std::string& iCommandStr) {
00072         //
00073         _command = getCommand (iCommandStr);
00074     }
00075
00076     // //////////////////////////////////////
00077     std::string JSonCommand::describeLabels() {
00078         std::ostringstream ostr;
00079         for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
00080             if (idx != 0) {
00081                 ostr << ", ";
00082             }
00083             ostr << _labels[idx] << " "; // " << _commandLabels[idx] << " ";
00084         }
00085         return ostr.str();
00086     }
00087
00088     // //////////////////////////////////////
00089     JSonCommand::EN_JSonCommand JSonCommand::getCommand()
00090     const {
00091         return _command;
00092     }
00093
00094     // //////////////////////////////////////
00095     const std::string JSonCommand::describe() const {
00096         std::ostringstream ostr;
00097         ostr << _labels[_command];
00098         return ostr.str();
00099     }
00100
00101     // //////////////////////////////////////
00102     bool JSonCommand::
00103     operator== (const EN_JSonCommand& iCommand) const {
00104         return (_command == iCommand);
00105     }
00106 }

```

### 33.87 stdair/basic/JSonCommand.hpp File Reference

```

#include <string>
#include <stdair/basic/StructAbstract.hpp>

```

#### Classes

- struct [stdair::JSonCommand](#)  
*Enumeration of json commands.*

#### Namespaces

- [stdair](#)  
*Handle on the StdAir library context.*

### 33.88 JSonCommand.hpp

```

00001 #ifndef __STDAIR_BAS_JSONCOMMAND_HPP
00002 #define __STDAIR_BAS_JSONCOMMAND_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>

```

```

00009 // StdAir
00010 #include <stdair/basic/StructAbstract.hpp>
00011
00012 namespace stdair {
00013
00017     struct JSonCommand : public StructAbstract {
00018     public:
00019         typedef enum {
00020             LIST = 0,
00021             FLIGHT_DATE,
00022             EVENT_LIST,
00023             BREAK_POINT,
00024             RUN,
00025             RESET,
00026             STATUS,
00027             CONFIG,
00028             LAST_VALUE
00029         } EN_JSonCommand;
00030
00036         static EN_JSonCommand getCommand (const std::string& iCommandStr);
00037
00041         static std::string getLabel(const EN_JSonCommand&);
00042
00046         static std::string describeLabels();
00047
00051         EN_JSonCommand getCommand() const;
00052
00057         const std::string describe() const;
00058
00059     public:
00063         bool operator== (const EN_JSonCommand&) const;
00064
00065     public:
00069         JSonCommand (const EN_JSonCommand&);
00070
00074         JSonCommand (const std::string&);
00075
00079         JSonCommand (const JSonCommand&);
00080
00081     private:
00085         JSonCommand();
00086
00087
00088     private:
00092         static const std::string _labels[LAST_VALUE];
00093
00094     private:
00095         // ////////// Attributes //////////
00099         EN_JSonCommand _command;
00100     };
00101
00102 }
00103 #endif // __STDAIR_BAS_JSONCOMMAND_HPP

```

### 33.89 stdair/basic/OptimisationMethod.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/OptimisationMethod.hpp>

```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.90 OptimisationMethod.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>

```

```

00007 // StdAir
00008 #include <stdair/stdair_exceptions.hpp>
00009 #include <stdair/basic/OptimisationMethod.hpp>
00010
00011 namespace stdair {
00012
00013 // //////////////////////////////////////
00014 const std::string OptimisationMethod::_labels[LAST_VALUE] =
00015     { "Leg based Monte Carlo", "Leg based EMSRb" };
00016
00017 // //////////////////////////////////////
00018 const char OptimisationMethod::
00019     _methodLabels[LAST_VALUE] = { 'M', 'E' };
00020
00021
00022 // //////////////////////////////////////
00023 OptimisationMethod::OptimisationMethod()
00024     : _method (LAST_VALUE) {
00025     assert (false);
00026 }
00027
00028 // //////////////////////////////////////
00029 OptimisationMethod::
00030 OptimisationMethod (const OptimisationMethod& iOptimisationMethod)
00031     : _method (iOptimisationMethod._method) {
00032 }
00033
00034 // //////////////////////////////////////
00035 OptimisationMethod::
00036 OptimisationMethod (const EN_OptimisationMethod& iOptimisationMethod)
00037     : _method (iOptimisationMethod) {
00038 }
00039
00040 // //////////////////////////////////////
00041 OptimisationMethod::OptimisationMethod (const char iMethod) {
00042     switch (iMethod) {
00043     case 'M': _method = LEG_BASED_MC; break;
00044     case 'E': _method = LEG_BASED_EMSR_B; break;
00045     default: _method = LAST_VALUE; break;
00046     }
00047
00048     if (_method == LAST_VALUE) {
00049         const std::string& lLabels = describeLabels();
00050         std::ostringstream oMessage;
00051         oMessage << "The optimisation method '" << iMethod
00052             << "' is not known. Known optimisation methods: " << lLabels;
00053         throw CodeConversionException (oMessage.str());
00054     }
00055 }
00056
00057 // //////////////////////////////////////
00058 const std::string& OptimisationMethod::
00059 getLabel (const EN_OptimisationMethod& iMethod) {
00060     return _labels[iMethod];
00061 }
00062
00063 // //////////////////////////////////////
00064 char OptimisationMethod::getMethodLabel (const
EN_OptimisationMethod& iMethod) {
00065     return _methodLabels[iMethod];
00066 }
00067
00068 // //////////////////////////////////////
00069 std::string OptimisationMethod::
00070 getMethodLabelAsString (const EN_OptimisationMethod& iMethod
) {
00071     std::ostringstream oStr;
00072     oStr << _methodLabels[iMethod];
00073     return oStr.str();
00074 }
00075
00076 // //////////////////////////////////////
00077 std::string OptimisationMethod::describeLabels() {
00078     std::ostringstream ostr;
00079     for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
00080         if (idx != 0) {
00081             ostr << ", ";
00082         }
00083         ostr << _labels[idx] << " (" << _methodLabels[idx] << ")";
00084     }
00085     return ostr.str();
00086 }
00087
00088 // //////////////////////////////////////
00089 OptimisationMethod::EN_OptimisationMethod
OptimisationMethod::getMethod() const {
00090     return _method;

```

```

00091     }
00092
00093     // //////////////////////////////////////
00094     std::string OptimisationMethod::getMethodAsString() const {
00095         std::ostringstream ostr;
00096         ostr << _methodLabels[_method];
00097         return ostr.str();
00098     }
00099
00100     // //////////////////////////////////////
00101     const std::string OptimisationMethod::describe() const {
00102         std::ostringstream ostr;
00103         ostr << _labels[_method];
00104         return ostr.str();
00105     }
00106
00107     // //////////////////////////////////////
00108     bool OptimisationMethod::
00109     operator== (const EN_OptimisationMethod& iMethod) const {
00110         return (_method == iMethod);
00111     }
00112
00113 }

```

### 33.91 stdair/basic/OptimisationMethod.hpp File Reference

```

#include <string>
#include <stdair/basic/StructAbstract.hpp>

```

#### Classes

- struct [stdair::OptimisationMethod](#)

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.92 OptimisationMethod.hpp

```

00001 #ifndef __STDAIR_BAS_OPTIMISATIONMETHOD_HPP
00002 #define __STDAIR_BAS_OPTIMISATIONMETHOD_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/basic/StructAbstract.hpp>
00011
00012 namespace stdair {
00013
00014     struct OptimisationMethod : public StructAbstract {
00015     public:
00016         typedef enum {
00017             LEG_BASED_MC = 0,
00018             LEG_BASED_EMSR_B,
00019             LAST_VALUE
00020         } EN_OptimisationMethod;
00021
00022         static const std::string& getLabel (const EN_OptimisationMethod&);
00023
00024         static char getMethodLabel (const EN_OptimisationMethod&);
00025
00026         static std::string getMethodLabelAsString (const
00027             EN_OptimisationMethod&);
00028
00029         static std::string describeLabels ();
00030
00031         EN_OptimisationMethod getMethod() const;
00032
00033     };
00034
00035 }
00036
00037 #endif

```

```

00040     std::string getMethodAsString() const;
00041
00044     const std::string describe() const;
00045
00046 public:
00048     bool operator== (const EN_OptimisationMethod&) const;
00049
00050 public:
00052     OptimisationMethod (const EN_OptimisationMethod&);
00054     OptimisationMethod (const char iMethod);
00056     OptimisationMethod (const OptimisationMethod&);
00057
00058 private:
00060     OptimisationMethod();
00061
00062
00063 private:
00065     static const std::string _labels[LAST_VALUE];
00067     static const char _methodLabels[LAST_VALUE];
00068
00069
00070 private:
00071     // ////////// Attributes //////////
00073     EN_OptimisationMethod _method;
00074 };
00075
00076 }
00077 #endif // __STDAIR_BAS_OPTIMISATIONMETHOD_HPP

```

### 33.93 stdair/basic/PartnershipTechnique.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/PartnershipTechnique.hpp>

```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.94 PartnershipTechnique.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/stdair_exceptions.hpp>
00009 #include <stdair/basic/PartnershipTechnique.hpp>
00010
00011 namespace stdair {
00012
00013     // //////////////////////////////////////
00014     const std::string PartnershipTechnique::_labels[LAST_VALUE] =
00015     {
00016         "None",
00017         "RevenueAvailabilityExchangeDemandAggregation",
00018         "RevenueAvailabilityExchangeYieldProration",
00019         "InterlineBidPriceDemandAggregation",
00020         "InterlineBidPriceYieldProration",
00021         "NonProtectionistInterlineBidPriceYieldProration",
00022         "RevenueManagementCooperation",
00023         "AdvancedRevenueManagementCooperation"};
00024
00025     // //////////////////////////////////////
00026     const char PartnershipTechnique::_techniqueLabels[LAST_VALUE] = { 'N',
00027                                                                           'R',
00028                                                                           'I',
00029                                                                           'U',
00030                                                                           'C',

```

```

00032                                     'A'};
00033
00034
00035 // //////////////////////////////////////
00036 PartnershipTechnique::PartnershipTechnique() : _technique (LAST_VALUE) {
00037     assert (false);
00038 }
00039
00040 // //////////////////////////////////////
00041 PartnershipTechnique::
00042 PartnershipTechnique (const PartnershipTechnique& iPartnershipTechnique)
00043     : _technique (iPartnershipTechnique._technique) {
00044 }
00045
00046 // //////////////////////////////////////
00047 PartnershipTechnique::
00048 PartnershipTechnique (const EN_PartnershipTechnique& iPartnershipTechnique)
00049     : _technique (iPartnershipTechnique) {
00050 }
00051
00052 // //////////////////////////////////////
00053 PartnershipTechnique::EN_PartnershipTechnique
00054 PartnershipTechnique::getTechnique (const char iTechniqueChar) {
00055     EN_PartnershipTechnique oTechnique;
00056     switch (iTechniqueChar) {
00057     case 'N': oTechnique = NONE; break;
00058     case 'r': oTechnique = RAE_DA; break;
00059     case 'R': oTechnique = RAE_YP; break;
00060     case 'i': oTechnique = IBP_DA; break;
00061     case 'I': oTechnique = IBP_YP; break;
00062     case 'U': oTechnique = IBP_YP_U; break;
00063     case 'C': oTechnique = RMC; break;
00064     case 'A': oTechnique = A_RMC; break;
00065     default: oTechnique = LAST_VALUE; break;
00066     }
00067
00068     if (oTechnique == LAST_VALUE) {
00069         const std::string& lLabels = describeLabels();
00070         std::ostringstream oMessage;
00071         oMessage << "The partnership technique '"
00072                 << iTechniqueChar
00073                 << "' is not known. Known partnership techniques: "
00074                 << lLabels;
00075         throw CodeConversionException (oMessage.str());
00076     }
00077
00078     return oTechnique;
00079 }
00080
00081 // //////////////////////////////////////
00082 PartnershipTechnique::PartnershipTechnique (const char iTechniqueChar)
00083     : _technique (getTechnique (iTechniqueChar)) {
00084 }
00085
00086 // //////////////////////////////////////
00087 PartnershipTechnique::
00088 PartnershipTechnique (const std::string& iTechniqueStr) {
00089     //
00090     const size_t lSize = iTechniqueStr.size();
00091     assert (lSize == 1);
00092     const char lTechniqueChar = iTechniqueStr[0];
00093     _technique = getTechnique (lTechniqueChar);
00094 }
00095
00096 // //////////////////////////////////////
00097 const std::string& PartnershipTechnique::
00098 getLabel (const EN_PartnershipTechnique& iTechnique) {
00099     return _labels[iTechnique];
00100 }
00101
00102 // //////////////////////////////////////
00103 char PartnershipTechnique::
00104 getTechniqueLabel (const EN_PartnershipTechnique& iTechnique) {
00105     return _techniqueLabels[iTechnique];
00106 }
00107
00108 // //////////////////////////////////////
00109 std::string PartnershipTechnique::
00110 getTechniqueLabelAsString (const
00111 EN_PartnershipTechnique& iTechnique) {
00112     std::ostringstream oStr;
00113     oStr << _techniqueLabels[iTechnique];
00114     return oStr.str();
00115 }
00116
00117 // //////////////////////////////////////
00117 std::string PartnershipTechnique::describeLabels() {

```

```

00118     std::ostringstream ostr;
00119     for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
00120         if (idx != 0) {
00121             ostr << ", ";
00122         }
00123         ostr << _labels[idx] << " (" << _techniqueLabels[idx] << ")";
00124     }
00125     return ostr.str();
00126 }
00127
00128 // //////////////////////////////////////
00129 PartnershipTechnique::EN_PartnershipTechnique
00130 PartnershipTechnique::getTechnique() const {
00131     return _technique;
00132 }
00133
00134 // //////////////////////////////////////
00135 char PartnershipTechnique::getTechniqueAsChar() const {
00136     const char oTechniqueChar = _techniqueLabels[_technique];
00137     return oTechniqueChar;
00138 }
00139
00140 // //////////////////////////////////////
00141 std::string PartnershipTechnique::getTechniqueAsString() const
00142 {
00143     std::ostringstream ostr;
00144     ostr << _techniqueLabels[_technique];
00145     return ostr.str();
00146 }
00147
00148 // //////////////////////////////////////
00149 const std::string PartnershipTechnique::describe() const {
00150     std::ostringstream ostr;
00151     ostr << _labels[_technique];
00152     return ostr.str();
00153 }
00154
00155 // //////////////////////////////////////
00156 bool PartnershipTechnique::
00157 operator== (const EN_PartnershipTechnique& iTechnique) const {
00158     return (_technique == iTechnique);
00159 }
00160 }

```

### 33.95 stdair/basic/PartnershipTechnique.hpp File Reference

```

#include <string>
#include <stdair/basic/StructAbstract.hpp>

```

#### Classes

- struct [stdair::PartnershipTechnique](#)  
*Enumeration of partnership techniques.*

#### Namespaces

- [stdair](#)  
*Handle on the StdAir library context.*

### 33.96 PartnershipTechnique.hpp

```

00001 #ifndef __STDAIR_BAS_PARTNERSHIPTECHNIQUE_HPP
00002 #define __STDAIR_BAS_PARTNERSHIPTECHNIQUE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir

```

```

00010 #include <stdair/basic/StructAbstract.hpp>
00011
00012 namespace stdair {
00013
00017     struct PartnershipTechnique : public StructAbstract {
00018     public:
00019         typedef enum {
00020             NONE = 0,
00021             RAE_DA,
00022             RAE_YP,
00023             IBP_DA,
00024             IBP_YP,
00025             IBP_YP_U,
00026             RMC,
00027             A_RMC,
00028             LAST_VALUE
00029         } EN_PartnershipTechnique;
00030
00034         static const std::string& getLabel (const EN_PartnershipTechnique&);
00035
00039         static EN_PartnershipTechnique getTechnique (const char);
00040
00044         static char getTechniqueLabel (const
EN_PartnershipTechnique&);
00045
00049         static std::string getTechniqueLabelAsString (const
EN_PartnershipTechnique&);
00050
00054         static std::string describeLabels();
00055
00059         EN_PartnershipTechnique getTechnique() const;
00060
00064         char getTechniqueAsChar() const;
00065
00069         std::string getTechniqueAsString() const;
00070
00075         const std::string describe() const;
00076
00077     public:
00081         bool operator== (const EN_PartnershipTechnique&) const;
00082
00083     public:
00087         PartnershipTechnique (const EN_PartnershipTechnique&);
00091         PartnershipTechnique (const char iTechnique);
00095         PartnershipTechnique (const std::string& iTechnique);
00096
00100         PartnershipTechnique (const PartnershipTechnique&);
00101
00102     private:
00106         PartnershipTechnique();
00107
00108
00109     private:
00113         static const std::string _labels[LAST_VALUE];
00117         static const char _techniqueLabels[LAST_VALUE];
00118
00119     private:
00120         // ////////// Attributes //////////
00124         EN_PartnershipTechnique _technique;
00125     };
00126
00127 }
00128 #endif // __STDAIR_BAS_PARTNERSHIPTECHNIQUE_HPP

```

### 33.97 stdair/basic/PassengerChoiceModel.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/PassengerChoiceModel.hpp>

```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*



### 33.98 PassengerChoiceModel.cpp

```

00001 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00002 // Import section
00003 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/stdair_exceptions.hpp>
00009 #include <stdair/basic/PassengerChoiceModel.hpp>
00010
00011 namespace stdair {
00012
00013 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00014 const std::string PassengerChoiceModel::_labels[LAST_VALUE] =
00015     { "HardRestrictionModel", "PriceOrientedModel", "HybridModel" };
00016
00017 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00018 const char PassengerChoiceModel::_modelLabels[LAST_VALUE] = { 'R', 'P', 'H' };
00019
00020
00021 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00022 PassengerChoiceModel::PassengerChoiceModel()
00023     : _model (LAST_VALUE) {
00024     assert (false);
00025 }
00026
00027 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00028 PassengerChoiceModel::
00029 PassengerChoiceModel (const PassengerChoiceModel& iPassengerChoiceModel)
00030     : _model (iPassengerChoiceModel._model) {
00031 }
00032
00033 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00034 PassengerChoiceModel::
00035 PassengerChoiceModel (const EN_PassengerChoiceModel& iPassengerChoiceModel)
00036     : _model (iPassengerChoiceModel) {
00037 }
00038
00039 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00040 PassengerChoiceModel::PassengerChoiceModel (const char iModel) {
00041     switch (iModel) {
00042     case 'R': _model = HARD_RESTRICTION; break;
00043     case 'P': _model = PRICE_ORIENTED; break;
00044     case 'H': _model = HYBRID; break;
00045     default: _model = LAST_VALUE; break;
00046     }
00047
00048     if (_model == LAST_VALUE) {
00049         const std::string& lLabels = describeLabels();
00050         std::ostringstream oMessage;
00051         oMessage << "The passenger choice model '"
00052             << " is not known. Known passenger choice models " << lLabels;
00053         throw stdair::CodeConversionException (oMessage.str());
00054     }
00055 }
00056
00057 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00058 const std::string& PassengerChoiceModel::
00059 getLabel (const EN_PassengerChoiceModel& iModel) {
00060     return _labels[iModel];
00061 }
00062
00063 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00064 char PassengerChoiceModel::getModelLabel (const
00065 EN_PassengerChoiceModel& iModel) {
00066     return _modelLabels[iModel];
00067 }
00068
00069 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00070 std::string PassengerChoiceModel::
00071 getModelLabelAsString (const EN_PassengerChoiceModel&
00072 iModel) {
00073     std::ostringstream oStr;
00074     oStr << _modelLabels[iModel];
00075     return oStr.str();
00076 }
00077
00078 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00079 std::string PassengerChoiceModel::describeLabels() {
00080     std::ostringstream ostr;
00081     for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
00082         if (idx != 0) {
00083             ostr << ", ";

```

```

00084         ostr << _labels[idx] << " (" << _modelLabels[idx] << ")";
00085     }
00086     return ostr.str();
00087 }
00088
00089 // //////////////////////////////////////
00090 PassengerChoiceModel::EN_PassengerChoiceModel
PassengerChoiceModel::getModel() const {
00091     return _model;
00092 }
00093
00094 // //////////////////////////////////////
00095 std::string PassengerChoiceModel::getModelAsString() const {
00096     std::ostringstream ostr;
00097     ostr << _modelLabels[_model];
00098     return ostr.str();
00099 }
00100
00101 // //////////////////////////////////////
00102 const std::string PassengerChoiceModel::describe() const {
00103     std::ostringstream ostr;
00104     ostr << _labels[_model];
00105     return ostr.str();
00106 }
00107
00108 // //////////////////////////////////////
00109 bool PassengerChoiceModel::
00110 operator== (const EN_PassengerChoiceModel& iModel) const {
00111     return (_model == iModel);
00112 }
00113
00114 }

```

### 33.99 stdair/basic/PassengerChoiceModel.hpp File Reference

```

#include <string>
#include <stdair/basic/StructAbstract.hpp>

```

#### Classes

- struct [stdair::PassengerChoiceModel](#)

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.100 PassengerChoiceModel.hpp

```

00001 #ifndef __STDAIR_BAS_PASSENGERCHOICEMODEL_HPP
00002 #define __STDAIR_BAS_PASSENGERCHOICEMODEL_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/basic/StructAbstract.hpp>
00011
00012 namespace stdair {
00013
00014     struct PassengerChoiceModel : public
stdair::StructAbstract {
00016     public:
00017         typedef enum {
00018             HARD_RESTRICTION = 0,
00019             PRICE_ORIENTED,
00020             HYBRID,
00021             LAST_VALUE
00022         } EN_PassengerChoiceModel;
00023

```

```

00026     static const std::string& getLabel (const EN_PassengerChoiceModel&);
00027
00029     static char getModelLabel (const EN_PassengerChoiceModel&);
00030
00032     static std::string getModelLabelAsString (const
EN_PassengerChoiceModel&);
00033
00035     static std::string describeLabels();
00036
00038     EN_PassengerChoiceModel getModel() const;
00039
00041     std::string getModelAsString() const;
00042
00045     const std::string describe() const;
00046
00047 public:
00049     bool operator== (const EN_PassengerChoiceModel&) const;
00050
00051 public:
00053     PassengerChoiceModel (const EN_PassengerChoiceModel&);
00055     PassengerChoiceModel (const char iModel);
00057     PassengerChoiceModel (const PassengerChoiceModel&);
00058
00059 private:
00061     PassengerChoiceModel();
00062
00063
00064 private:
00066     static const std::string _labels[LAST_VALUE];
00068     static const char _modelLabels[LAST_VALUE];
00069
00070
00071 private:
00072     // ////////// Attributes //////////
00074     EN_PassengerChoiceModel _model;
00075 };
00076 }
00077
00078 #endif // __STDAIR_BAS_PASSENGERCHOICEMODEL_HPP

```

### 33.101 stdair/basic/PassengerType.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/PassengerType.hpp>

```

#### Namespaces

- **stdair**

*Handle on the StdAir library context.*

### 33.102 PassengerType.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/stdair_exceptions.hpp>
00009 #include <stdair/basic/PassengerType.hpp>
00010
00011 namespace stdair {
00012
00013     // //////////////////////////////////////
00014     const std::string PassengerType::_labels[LAST_VALUE] =
00015         { "Leisure", "Business", "First" };
00016
00017     const char PassengerType::_typeLabels[LAST_VALUE] = { 'L', 'B', 'F' };
00018
00019
00020     // //////////////////////////////////////

```

```

00021 PassengerType::PassengerType (const
EN_PassengerType& iPassengerType)
00022 : _type (iPassengerType) {
00023 }
00024
00025 // //////////////////////////////////////
00026 PassengerType::PassengerType (const char iType) {
00027     switch (iType) {
00028         case 'L': _type = LEISURE; break;
00029         case 'B': _type = BUSINESS; break;
00030         case 'F': _type = FIRST; break;
00031         default: _type = LAST_VALUE; break;
00032     }
00033
00034     if (_type == LAST_VALUE) {
00035         const std::string& lLabels = describeLabels();
00036         std::ostringstream oMessage;
00037         oMessage << "The passenger type '" << iType
00038             << "' is not known. Known passenger types: " << lLabels;
00039         throw CodeConversionException (oMessage.str());
00040     }
00041 }
00042
00043 // //////////////////////////////////////
00044 const std::string& PassengerType::getLabel (const
EN_PassengerType& iType) {
00045     return _labels[iType];
00046 }
00047
00048 // //////////////////////////////////////
00049 char PassengerType::getTypeLabel (const
EN_PassengerType& iType) {
00050     return _typeLabels[iType];
00051 }
00052
00053 // //////////////////////////////////////
00054 std::string PassengerType::
00055 getTypeLabelAsString (const EN_PassengerType& iType) {
00056     std::ostringstream oStr;
00057     oStr << _typeLabels[iType];
00058     return oStr.str();
00059 }
00060
00061 // //////////////////////////////////////
00062 std::string PassengerType::describeLabels() {
00063     std::ostringstream ostr;
00064     for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
00065         if (idx != 0) {
00066             ostr << ", ";
00067         }
00068         ostr << _labels[idx];
00069     }
00070     return ostr.str();
00071 }
00072
00073 // //////////////////////////////////////
00074 PassengerType::EN_PassengerType
PassengerType::getType() const {
00075     return _type;
00076 }
00077
00078 // //////////////////////////////////////
00079 std::string PassengerType::getTypeAsString() const {
00080     std::ostringstream oStr;
00081     oStr << _typeLabels[_type];
00082     return oStr.str();
00083 }
00084
00085 // //////////////////////////////////////
00086 const std::string PassengerType::describe() const {
00087     std::ostringstream ostr;
00088     ostr << _labels[_type];
00089     return ostr.str();
00090 }
00091
00092 // //////////////////////////////////////
00093 bool PassengerType::operator== (const
EN_PassengerType& iType) const {
00094     return (_type == iType);
00095 }
00096
00097 }

```

### 33.103 stdair/basic/PassengerType.hpp File Reference

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

#### Classes

- struct [stdair::PassengerType](#)

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.104 PassengerType.hpp

```
00001 #ifndef __STDAIR_BAS_PASSENGERTYPE_HPP
00002 #define __STDAIR_BAS_PASSENGERTYPE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/basic/StructAbstract.hpp>
00011
00012 namespace stdair {
00013
00014     struct PassengerType : public StructAbstract {
00015     public:
00016         typedef enum {
00017             LEISURE = 0,
00018             BUSINESS,
00019             FIRST,
00020             LAST_VALUE
00021         } EN_PassengerType;
00022
00023         static const std::string& getLabel (const EN_PassengerType&);
00024
00025         static char getTypeLabel (const EN_PassengerType&);
00026
00027         static std::string getTypeLabelAsString (const
00028             EN_PassengerType&);
00029
00030         static std::string describeLabels();
00031
00032         EN_PassengerType getType() const;
00033
00034         std::string getTypeAsString() const;
00035
00036         const std::string describe() const;
00037
00038     public:
00039         bool operator== (const EN_PassengerType&) const;
00040
00041     public:
00042         PassengerType (const EN_PassengerType&);
00043         PassengerType (const char iType);
00044
00045     private:
00046         static const std::string _labels[LAST_VALUE];
00047         static const char _typeLabels[LAST_VALUE];
00048
00049     private:
00050         // ////////// Attributes //////////
00051         EN_PassengerType _type;
00052     };
00053 }
00054 #endif // __STDAIR_BAS_PASSENGERTYPE_HPP
```

## 33.105 stdair/basic/PreOptimisationMethod.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/PreOptimisationMethod.hpp>
```

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.106 PreOptimisationMethod.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/stdair_exceptions.hpp>
00009 #include <stdair/basic/PreOptimisationMethod.hpp>
00010
00011 namespace stdair {
00012
00013 // //////////////////////////////////////
00014 const std::string PreOptimisationMethod::_labels[LAST_VALUE] =
00015     {"None", "Fare Adjustment", "Marginal Revenue Transformation"};
00016
00017 // //////////////////////////////////////
00018 const char PreOptimisationMethod::_methodLabels[LAST_VALUE] = {'N', 'F', 'M'};
00019
00020
00021
00022 // //////////////////////////////////////
00023 PreOptimisationMethod::PreOptimisationMethod()
00024     : _method (LAST_VALUE) {
00025     assert (false);
00026 }
00027
00028 // //////////////////////////////////////
00029 PreOptimisationMethod::
00030 PreOptimisationMethod (const PreOptimisationMethod& iPreOptimisationMethod)
00031     : _method (iPreOptimisationMethod._method) {
00032 }
00033
00034 // //////////////////////////////////////
00035 PreOptimisationMethod::
00036 PreOptimisationMethod (const EN_PreOptimisationMethod& iPreOptimisationMethod)
00037     : _method (iPreOptimisationMethod) {
00038 }
00039
00040 // //////////////////////////////////////
00041 PreOptimisationMethod::PreOptimisationMethod (const char iMethod) {
00042     switch (iMethod) {
00043     case 'N': _method = NONE; break;
00044     case 'F': _method = FA; break;
00045     case 'M': _method = MRT; break;
00046     default: _method = LAST_VALUE; break;
00047     }
00048
00049     if (_method == LAST_VALUE) {
00050         const std::string& lLabels = describeLabels();
00051         std::ostringstream oMessage;
00052         oMessage << "The pre-optimisation method '" << iMethod
00053             << "' is not known. Known pre-optimisation methods: " << lLabels;
00054         throw CodeConversionException (oMessage.str());
00055     }
00056 }
00057
00058 // //////////////////////////////////////
00059 const std::string& PreOptimisationMethod::
00060 getLabel (const EN_PreOptimisationMethod& iMethod) {
00061     return _labels[iMethod];
00062 }
```

```

00063
00064 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00065 char PreOptimisationMethod::getMethodLabel (const
EN_PreOptimisationMethod& iMethod) {
00066     return _methodLabels[iMethod];
00067 }
00068
00069 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00070 std::string PreOptimisationMethod::
00071 getMethodLabelAsString (const EN_PreOptimisationMethod&
iMethod) {
00072     std::ostringstream ostr;
00073     ostr << _methodLabels[iMethod];
00074     return ostr.str();
00075 }
00076
00077 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00078 std::string PreOptimisationMethod::describeLabels() {
00079     std::ostringstream ostr;
00080     for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
00081         if (idx != 0) {
00082             ostr << ", ";
00083         }
00084         ostr << _labels[idx] << " (" << _methodLabels[idx] << ")";
00085     }
00086     return ostr.str();
00087 }
00088
00089 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00090 PreOptimisationMethod::EN_PreOptimisationMethod
PreOptimisationMethod::getMethod() const {
00091     return _method;
00092 }
00093
00094 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00095 std::string PreOptimisationMethod::getMethodAsString() const {
00096     std::ostringstream ostr;
00097     ostr << _methodLabels[_method];
00098     return ostr.str();
00099 }
00100
00101 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00102 const std::string PreOptimisationMethod::describe() const {
00103     std::ostringstream ostr;
00104     ostr << _labels[_method];
00105     return ostr.str();
00106 }
00107
00108 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00109 bool PreOptimisationMethod::
00110 operator== (const EN_PreOptimisationMethod& iMethod) const {
00111     return (_method == iMethod);
00112 }
00113
00114 }

```

### 33.107 stdair/basic/PreOptimisationMethod.hpp File Reference

```

#include <string>
#include <stdair/basic/StructAbstract.hpp>

```

#### Classes

- struct [stdair::PreOptimisationMethod](#)

#### Namespaces

- [stdair](#)  
*Handle on the StdAir library context.*

### 33.108 PreOptimisationMethod.hpp

```

00001 #ifndef __STDAIR_BAS_PREOPTIMISATIONMETHOD_HPP

```

```

00002 #define __STDAIR_BAS_PREOPTIMISATIONMETHOD_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/basic/StructAbstract.hpp>
00011
00012 namespace stdair {
00013
00014     struct PreOptimisationMethod : public StructAbstract {
00015     public:
00016         typedef enum {
00017             NONE = 0,
00018             FA,
00019             MRT,
00020             LAST_VALUE
00021         } EN_PreOptimisationMethod;
00022
00023         static const std::string& getLabel (const EN_PreOptimisationMethod&);
00024
00025         static char getMethodLabel (const EN_PreOptimisationMethod&);
00026
00027         static std::string getMethodLabelAsString (const
00028             EN_PreOptimisationMethod&);
00029
00030         static std::string describeLabels();
00031
00032         EN_PreOptimisationMethod getMethod() const;
00033
00034         std::string getMethodAsString() const;
00035
00036         const std::string describe() const;
00037
00038     public:
00039         bool operator== (const EN_PreOptimisationMethod&) const;
00040
00041     public:
00042         PreOptimisationMethod (const EN_PreOptimisationMethod&);
00043         PreOptimisationMethod (const char iMethod);
00044         PreOptimisationMethod (const PreOptimisationMethod&);
00045
00046     private:
00047         PreOptimisationMethod();
00048
00049     private:
00050         static const std::string _labels[LAST_VALUE];
00051         static const char _methodLabels[LAST_VALUE];
00052
00053     private:
00054         // ////////// Attributes //////////
00055         EN_PreOptimisationMethod _method;
00056     };
00057 }
00058 #endif // __STDAIR_BAS_PREOPTIMISATIONMETHOD_HPP

```

### 33.109 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

- [stdair](#)

*Handle on the StdAir library context.*



### 33.110 ProgressStatus.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/stdair_exceptions.hpp>
00009 #include <stdair/basic/BasConst_Event.hpp>
00010 #include <stdair/basic/ProgressStatus.hpp>
00011
00012 namespace stdair {
00013
00014 // //////////////////////////////////////
00015 ProgressStatus::ProgressStatus (const Count_T& iCurrentNb,
00016                                const Count_T& iExpectedNb,
00017                                const Count_T& iActualNb)
00018     : _currentNb (iCurrentNb),
00019       _expectedNb (iExpectedNb), _actualNb (iActualNb) {
00020 }
00021
00022 // //////////////////////////////////////
00023 ProgressStatus::ProgressStatus (const Count_T& iExpectedNb,
00024                                const Count_T& iActualNb)
00025     : _currentNb (DEFAULT_PROGRESS_STATUS),
00026       _expectedNb (iExpectedNb), _actualNb (iActualNb) {
00027 }
00028
00029 // //////////////////////////////////////
00030 ProgressStatus::ProgressStatus (const Count_T& iExpectedNb)
00031     : _currentNb (DEFAULT_PROGRESS_STATUS),
00032       _expectedNb (iExpectedNb), _actualNb (iExpectedNb) {
00033 }
00034
00035 // //////////////////////////////////////
00036 ProgressStatus::ProgressStatus ()
00037     : _currentNb (DEFAULT_PROGRESS_STATUS),
00038       _expectedNb (DEFAULT_PROGRESS_STATUS),
00039       _actualNb (DEFAULT_PROGRESS_STATUS) {
00040 }
00041
00042 // //////////////////////////////////////
00043 ProgressStatus::ProgressStatus (const
ProgressStatus& iProgressStatus)
00044     : _currentNb (iProgressStatus._currentNb),
00045       _expectedNb (iProgressStatus._expectedNb),
00046       _actualNb (iProgressStatus._actualNb) {
00047 }
00048
00049 // //////////////////////////////////////
00050 void ProgressStatus::reset () {
00051     _currentNb = DEFAULT_PROGRESS_STATUS;
00052     _actualNb = DEFAULT_PROGRESS_STATUS;
00053 }
00054
00055 // //////////////////////////////////////
00056 const std::string ProgressStatus::describe () const {
00057     std::ostringstream oStr;
00058     oStr << _currentNb << " / {" << _expectedNb << ", " << _actualNb << "}";
00059     return oStr.str();
00060 }
00061
00062 // //////////////////////////////////////
00063 const std::string ProgressStatus::toString () const {
00064     std::ostringstream oStr;
00065     oStr << std::setprecision (3) << progress()
00066         << "% (" << _currentNb << "/" << _actualNb << ")";
00067     return oStr.str();
00068 }
00069
00070 }

```

### 33.111 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>
```

## Classes

- struct [stdair::ProgressStatus](#)

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.112 ProgressStatus.hpp

```
00001 #ifndef __STDAIR_BAS_PROGRESSSTATUS_HPP
00002 #define __STDAIR_BAS_PROGRESSSTATUS_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // Boost Progress
00010 #include <boost/progress.hpp>
00011 // StdAir
00012 #include <stdair/basic/BasConst_Event.hpp>
00013 #include <stdair/stdair_basic_types.hpp>
00014 #include <stdair/basic/StructAbstract.hpp>
00015 #include <stdair/service/Logger.hpp>
00016
00017 namespace stdair {
00018
00027     struct ProgressStatus : public StructAbstract {
00028     public:
00029         // ////////////////////////////////////// Getters //////////////////////////////////////
00031         const Count_T& count() const {
00032             return _currentNb;
00033         }
00034
00036         const Count_T& getCurrentNb() const {
00037             return _currentNb;
00038         }
00039
00041         const Count_T& getExpectedNb() const {
00042             return _expectedNb;
00043         }
00044
00046         const Count_T& getActualNb() const {
00047             return _actualNb;
00048         }
00049
00051         const ProgressPercentage_T progress() const {
00052             if (_actualNb == 0) {
00053                 return 0;
00054             }
00055             Percentage_T lPercentage =
00056                 (static_cast<Percentage_T> (_currentNb)
00057                  / static_cast<Percentage_T> (_actualNb));
00058             lPercentage *= MAXIMUM_PROGRESS_STATUS;
00059             return lPercentage;
00060         }
00061
00062         // ////////////////////////////////////// Setters //////////////////////////////////////
00065         void setCurrentNb (const Count_T& iCurrentNb) {
00066             _currentNb = iCurrentNb;
00067         }
00068
00070         void setExpectedNb (const Count_T& iExpectedNb) {
00071             _expectedNb = iExpectedNb;
00072         }
00073     };
00074 }
```

```

00073
00075     void setActualNb (const Count_T& iActualNb) {
00076         _actualNb = iActualNb;
00077     }
00078
00080     void reset();
00081
00083     Count_T operator+= (Count_T iIncrement) {
00084         _currentNb += iIncrement;
00085         return _currentNb;
00086     }
00087
00089     Count_T operator++() {
00090         ++_currentNb;
00091         return _currentNb;
00092     }
00093
00094
00095 public:
00096     // //////////// Display Support Methods ////////////
00098     const std::string describe() const;
00099
00101     const std::string toString() const;
00102
00103
00104 public:
00112     ProgressStatus (const Count_T& iCurrentNb, const
Count_T& iExpectedNb,
00113                     const Count_T& iActualNb);
00114
00123     ProgressStatus (const Count_T& iExpectedNb, const
Count_T& iActualNb);
00124
00133     ProgressStatus (const Count_T& iActualNb);
00134
00140     ProgressStatus();
00141
00145     ProgressStatus (const ProgressStatus&);
00146
00147 private:
00148     // //////////// Attributes ////////////
00150     Count_T _currentNb;
00151
00153     Count_T _expectedNb;
00154
00156     Count_T _actualNb;
00157 };
00158
00159 }
00160 #endif // __STDAIR_BAS_PROGRESSSTATUS_HPP

```

### 33.113 stdair/basic/ProgressStatusSet.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/basic/ProgressStatusSet.hpp>

```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.114 ProgressStatusSet.cpp

```

00001 // //////////// Import section ////////////
00002 // Import section
00003 // //////////// STL ////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/ProgressStatusSet.hpp>
00009
00010 namespace stdair {

```

```

00011
00012 ////////////////////////////////////////////////////////////////////
00013 ProgressStatusSet::ProgressStatusSet()
00014 : _eventType (EventType::LAST_VALUE), _typeSpecificProgressStatus(),
00015   _generatorProgressStatus(), _overallProgressStatus(), _generatorKey ("") {
00016     assert (false);
00017 }
00018
00019 ////////////////////////////////////////////////////////////////////
00020 ProgressStatusSet::ProgressStatusSet (const EventType::EN_EventType& iType)
00021 : _eventType (iType), _typeSpecificProgressStatus(),
00022   _generatorProgressStatus(), _overallProgressStatus(), _generatorKey ("") {
00023 }
00024
00025 ////////////////////////////////////////////////////////////////////
00026 ProgressStatusSet::
00027 ProgressStatusSet (const ProgressStatusSet& iProgressStatusSet)
00028 : _eventType (iProgressStatusSet._eventType),
00029   _typeSpecificProgressStatus(iProgressStatusSet._typeSpecificProgressStatus),
00030   _generatorProgressStatus (iProgressStatusSet._generatorProgressStatus),
00031   _overallProgressStatus (iProgressStatusSet._overallProgressStatus),
00032   _generatorKey (iProgressStatusSet._generatorKey) {
00033 }
00034
00035 ////////////////////////////////////////////////////////////////////
00036 ProgressStatusSet::~ProgressStatusSet () {
00037 }
00038
00039 ////////////////////////////////////////////////////////////////////
00040 void ProgressStatusSet::fromStream (std::istream& ioIn) {
00041 }
00042
00043 ////////////////////////////////////////////////////////////////////
00044 const std::string ProgressStatusSet::describe() const {
00045     std::ostringstream oStr;
00046
00047     oStr << "-[Overall]"
00048     << "[" << _overallProgressStatus.getCurrentNb()
00049     << "/" << _overallProgressStatus.getExpectedNb()
00050     << "," << _overallProgressStatus.getActualNb()
00051     << "]" ";";
00052
00053     oStr << "[" << EventType (_eventType) << "]"
00054     << "[" << _typeSpecificProgressStatus.getCurrentNb()
00055     << "/" << _typeSpecificProgressStatus.getExpectedNb()
00056     << "," << _typeSpecificProgressStatus.getActualNb()
00057     << "]" ";";
00058
00059     oStr << " [Specific generator: " << _generatorKey << "]"
00060     << "[" << _generatorProgressStatus.getCurrentNb()
00061     << "/" << _generatorProgressStatus.getExpectedNb()
00062     << "," << _generatorProgressStatus.getActualNb()
00063     << "]" ";";
00064
00065     return oStr.str();
00066 }
00067
00068 }

```

### 33.115 stdair/basic/ProgressStatusSet.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_event_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/basic/EventType.hpp>
#include <stdair/basic/ProgressStatus.hpp>

```

#### Classes

- struct [stdair::ProgressStatusSet](#)

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.116 ProgressStatusSet.hpp

```

00001 #ifndef __STDAIR_BAS_PROGRESSSTATUSSET_HPP
00002 #define __STDAIR_BAS_PROGRESSSTATUSSET_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_basic_types.hpp>
00012 #include <stdair/stdair_event_types.hpp>
00013 #include <stdair/basic/StructAbstract.hpp>
00014 #include <stdair/basic/EventType.hpp>
00015 #include <stdair/basic/ProgressStatus.hpp>
00016
00017 namespace stdair {
00018
00019     struct ProgressStatusSet : public StructAbstract {
00020     // ////////////////////////////////////// Getters //////////////////////////////////////
00021     const ProgressStatus& getTypeSpecificStatus() const {
00022         return _typeSpecificProgressStatus;
00023     }
00024
00025     const ProgressStatus& getSpecificGeneratorStatus() const {
00026         return _generatorProgressStatus;
00027     }
00028
00029     const ProgressStatus& getOverallStatus() const {
00030         return _overallProgressStatus;
00031     }
00032
00033     // ////////////////////////////////////// Setters //////////////////////////////////////
00034 public:
00035     void setTypeSpecificStatus (const ProgressStatus& iProgressStatus) {
00036         _typeSpecificProgressStatus = iProgressStatus;
00037     }
00038
00039     void setSpecificGeneratorStatus (const
00040 ProgressStatus& iProgressStatus,
00041                                     const EventGeneratorKey_T& iKey) {
00042         _generatorProgressStatus = iProgressStatus;
00043         _generatorKey = iKey;
00044     }
00045
00046     void setOverallStatus (const ProgressStatus& iProgressStatus) {
00047         _overallProgressStatus = iProgressStatus;
00048     }
00049
00050     // ////////////////////////////////////// Display methods //////////////////////////////////////
00051 public:
00052     void fromStream (std::istream& ioIn);
00053
00054     const std::string describe() const;
00055
00056     // ////////////////////////////////////// Constructors and destructors //////////////////////////////////////
00057 public:
00058     ProgressStatusSet (const EventType::EN_EventType&);
00059     ProgressStatusSet (const ProgressStatusSet&);
00060     ~ProgressStatusSet();
00061
00062 private:
00063     ProgressStatusSet ();
00064
00065     // ////////////////////////////////////// Attributes //////////////////////////////////////
00066 private:
00067     const EventType::EN_EventType _eventType;
00068
00069     ProgressStatus _typeSpecificProgressStatus;
00070
00071     ProgressStatus _generatorProgressStatus;
00072
00073     ProgressStatus _overallProgressStatus;

```

```

00125
00129     EventGeneratorKey_T _generatorKey;
00130 };
00131
00132 }
00133 #endif // __STDAIR_BASIC_PROGRESSSTATUSSET_HPP

```

### 33.117 stdair/basic/RandomGeneration.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <boost/version.hpp>
#include <stdair/basic/RandomGeneration.hpp>

```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.118 RandomGeneration.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost
00008 #include <boost/version.hpp>
00009 #if BOOST_VERSION >= 103500
00010 #include <boost/math/distributions/normal.hpp>
00011 #endif // BOOST_VERSION >= 103500
00012 // StdAir
00013 #include <stdair/basic/RandomGeneration.hpp>
00014
00015 namespace stdair {
00016
00022 // //////////////////////////////////////
00023 RandomGeneration::RandomGeneration() : _generator (1) {
00024 }
00025
00026 // //////////////////////////////////////
00027 RandomGeneration::RandomGeneration (const
RandomSeed_T& iSeed)
00028 : _generator (iSeed) {
00029 }
00030
00031 // //////////////////////////////////////
00032 RandomGeneration::RandomGeneration (const
RandomGeneration& iRandomGeneration)
00033 : _generator (iRandomGeneration._generator) {
00034 }
00035
00036 // //////////////////////////////////////
00037 RandomGeneration::~~RandomGeneration() {
00038 }
00039
00040 // //////////////////////////////////////
00041 void RandomGeneration::init (const RandomSeed_T& iSeed) {
00042     _generator.seed (iSeed);
00043 }
00044
00045 // //////////////////////////////////////
00046 const std::string RandomGeneration::describe() const {
00047     std::ostringstream oStr;
00048     oStr << _generator;
00049     return oStr.str();
00050 }
00051
00052 // //////////////////////////////////////
00053 RealNumber_T RandomGeneration::generateUniform01() {
00054     UniformGenerator_T lGenerator (_generator, boost::uniform_real<>(0, 1));
00055     return lGenerator();
00056 }

```

```

00057
00058 ///////////////////////////////////////////////////////////////////
00059 RealNumber_T RandomGeneration::generateUniform(const
RealNumber_T& iMinValue,
00060                                         const RealNumber_T& iMaxValue) {
00061     const Probability_T lVariateUnif01 = generateUniform01();
00062     const RealNumber_T lVariateUnif =
00063         iMinValue + lVariateUnif01 * (iMaxValue - iMinValue);
00064     return lVariateUnif;
00065 }
00066
00067 ///////////////////////////////////////////////////////////////////
00068 RealNumber_T RandomGeneration::generateNormal (const
RealNumber_T& mu,
00069                                         const RealNumber_T& sigma) {
00070
00071 #if BOOST_VERSION >= 103500
00072     const Probability_T lVariateUnif = generateUniform01();
00073     const boost::math::normal lNormal (mu, sigma);
00074     const RealNumber_T lRealNumberOfRequestsToBeGenerated =
00075         boost::math::quantile (lNormal, lVariateUnif);
00076 #else // BOOST_VERSION >= 103500
00077     // TODO: rely on GSL when Boost version smaller than 1.35
00078     const RealNumber_T lRealNumberOfRequestsToBeGenerated = 0.0;
00079 #endif // BOOST_VERSION >= 103500
00080
00081     return lRealNumberOfRequestsToBeGenerated;
00082 }
00083
00084
00085 ///////////////////////////////////////////////////////////////////
00086 RealNumber_T RandomGeneration::generateExponential (
const RealNumber_T& lambda) {
00091     ExponentialDistribution_T lExponentialDistribution (lambda);
00092
00094     ExponentialGenerator_T lExponentialDistributionGenerator (
_generator,
00095                                         lExponentialDistribution);
00096
00097     // Generate a random variate, expressed in (fractional) day
00098     const RealNumber_T lExponentialVariateInDays =
00099         lExponentialDistributionGenerator();
00100
00101     return lExponentialVariateInDays;
00102 }
00103
00104 }

```

### 33.119 stdair/basic/RandomGeneration.hpp File Reference

```

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

```

#### Classes

- struct [stdair::RandomGeneration](#)  
Class holding a random generator.

#### Namespaces

- [stdair](#)  
Handle on the StdAir library context.

### 33.120 RandomGeneration.hpp

```

00001 #ifndef __STDAIR_BAS_RANDOMGENERATION_HPP
00002 #define __STDAIR_BAS_RANDOMGENERATION_HPP
00003
00004 ///////////////////////////////////////////////////////////////////
00005 // Import section

```

```

00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009 #include <stdair/stdair_maths_types.hpp>
00010 #include <stdair/basic/StructAbstract.hpp>
00011
00012 namespace stdair {
00013
00017     struct RandomGeneration : public StructAbstract {
00018     public:
00019         // ////////////////////////////////// Business Methods //////////////////////////////////
00024         RealNumber_T generateUniform01();
00025
00030         RealNumber_T operator() () {
00031             return generateUniform01();
00032         }
00033
00039         RealNumber_T generateUniform (const RealNumber_T&, const
RealNumber_T&);
00040
00045         RealNumber_T generateNormal (const RealNumber_T&, const
RealNumber_T&);
00046
00051         RealNumber_T generateExponential (const
RealNumber_T&);
00052
00056         BaseGenerator_T& getBaseGenerator () { return
_generator; }
00057
00058
00059     public:
00060         // ////////////////////////////////// Display Support Methods //////////////////////////////////
00064         const std::string describe() const;
00065
00066     public:
00067         // ////////////////////////////////// Constructors and destructors //////////////////////////////////
00072         RandomGeneration (const RandomSeed_T&);
00076         RandomGeneration();
00077
00078     private:
00082         RandomGeneration (const RandomGeneration&);
00086         RandomGeneration& operator= (const RandomGeneration& iRandomGeneration)
{
00087             _generator = iRandomGeneration._generator;
00088             return *this;
00089         }
00090     public:
00094         ~RandomGeneration();
00095
00103         void init (const RandomSeed_T&);
00104
00105         // ////////////////////////////////// Attributes //////////////////////////////////
00112         BaseGenerator_T _generator;
00113     };
00114
00115 }
00116 #endif // __STDAIR_BAS_RANDOMGENERATION_HPP

```

### 33.121 stdair/basic/SampleType.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/SampleType.hpp>

```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.122 SampleType.cpp

```

00001 // //////////////////////////////////////

```



```

00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/stdair_exceptions.hpp>
00009 #include <stdair/basic/SampleType.hpp>
00010
00011 namespace stdair {
00012
00013 // //////////////////////////////////////
00014 const std::string SampleType::_labels[LAST_VALUE] =
00015 { "All", "AllForPartnerships", "RevenueManagement", "Inventory", "Schedule",
00016   "RevenueAccounting", "FareQuote", "CRS", "DemandGeneration", "EventManager",
00017   "CustomerChoice" };
00018
00019 // //////////////////////////////////////
00020 const char SampleType::_typeLabels[LAST_VALUE] = { 'A', 'P', 'R', 'I', 'S', 'T', 'F', 'C', 'D', 'E', 'M' };
00021
00022
00023 // //////////////////////////////////////
00024 SampleType::SampleType()
00025 : _type (LAST_VALUE) {
00026   assert (false);
00027 }
00028
00029 // //////////////////////////////////////
00030 SampleType::SampleType (const SampleType& iSampleType)
00031 : _type (iSampleType._type) {
00032 }
00033
00034 // //////////////////////////////////////
00035 SampleType::SampleType (const EN_SampleType& iSampleType)
00036 : _type (iSampleType) {
00037 }
00038
00039 // //////////////////////////////////////
00040 SampleType::SampleType (const char iType) {
00041   switch (iType) {
00042     case 'A': _type = ALL; break;
00043     case 'P': _type = A4P; break;
00044     case 'R': _type = RMS; break;
00045     case 'I': _type = INV; break;
00046     case 'S': _type = SCH; break;
00047     case 'T': _type = RAC; break;
00048     case 'F': _type = FQT; break;
00049     case 'C': _type = CRS; break;
00050     case 'D': _type = DEM; break;
00051     case 'E': _type = EVT; break;
00052     case 'M': _type = CCM; break;
00053     default: _type = LAST_VALUE; break;
00054   }
00055
00056   if (_type == LAST_VALUE) {
00057     const std::string& lLabels = describeLabels();
00058     std::ostringstream oMessage;
00059     oMessage << "The sample type '" << iType
00060       << "' is not known. Known sample types: " << lLabels;
00061     throw CodeConversionException (oMessage.str());
00062   }
00063 }
00064
00065 // //////////////////////////////////////
00066 const std::string& SampleType::getLabel (const
00067 EN_SampleType& iType) {
00068   return _labels[iType];
00069 }
00070
00071 // //////////////////////////////////////
00072 char SampleType::getTypeLabel (const EN_SampleType& iType) {
00073   return _typeLabels[iType];
00074 }
00075
00076 // //////////////////////////////////////
00077 std::string SampleType::getTypeLabelAsString (const
00078 EN_SampleType& iType) {
00079   std::ostringstream oStr;
00080   oStr << _typeLabels[iType];
00081   return oStr.str();
00082 }
00083
00084 // //////////////////////////////////////
00085 std::string SampleType::describeLabels() {
00086   std::ostringstream ostr;
00087   for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {

```

```

00087         if (idx != 0) {
00088             ostr << ", ";
00089         }
00090         ostr << _labels[idx];
00091     }
00092     return ostr.str();
00093 }
00094
00095 // //////////////////////////////////////
00096 SampleType::EN_SampleType SampleType::getType() const {
00097     return _type;
00098 }
00099
00100 // //////////////////////////////////////
00101 std::string SampleType::getTypeAsString() const {
00102     std::ostringstream ostr;
00103     ostr << _typeLabels[_type];
00104     return ostr.str();
00105 }
00106
00107 // //////////////////////////////////////
00108 const std::string SampleType::describe() const {
00109     std::ostringstream ostr;
00110     ostr << _labels[_type];
00111     return ostr.str();
00112 }
00113
00114 // //////////////////////////////////////
00115 bool SampleType::operator== (const EN_SampleType& iType) const {
00116     return (_type == iType);
00117 }
00118
00119 }

```

### 33.123 stdair/basic/SampleType.hpp File Reference

```

#include <string>
#include <stdair/basic/StructAbstract.hpp>

```

#### Classes

- struct [stdair::SampleType](#)  
*Enumeration of BOM sample types.*

#### Namespaces

- [stdair](#)  
*Handle on the StdAir library context.*

### 33.124 SampleType.hpp

```

00001 #ifndef __STDAIR_BAS_SAMPLETYPE_HPP
00002 #define __STDAIR_BAS_SAMPLETYPE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/basic/StructAbstract.hpp>
00011
00012 namespace stdair {
00013
00025     struct SampleType : public StructAbstract {
00026     public:
00027         typedef enum {
00028             ALL = 0,
00029             A4P,
00030             RMS,
00031             INV,

```

```

00032     SCH,
00033     RAC,
00034     FQT,
00035     CRS,
00036     DEM,
00037     EVT,
00038     CCM,
00039     LAST_VALUE
00040 } EN_SampleType;
00041
00045     static const std::string& getLabel (const EN_SampleType&);
00046
00050     static char getTypeLabel (const EN_SampleType&);
00051
00055     static std::string getTypeLabelAsString (const
EN_SampleType&);
00056
00060     static std::string describeLabels();
00061
00065     EN_SampleType getType() const;
00066
00070     std::string getTypeAsString() const;
00071
00075     const std::string describe() const;
00076
00077 public:
00081     bool operator== (const EN_SampleType&) const;
00082
00083 public:
00087     SampleType (const EN_SampleType&);
00091     SampleType (const char iType);
00095     SampleType (const SampleType&);
00096
00097 private:
00101     SampleType();
00102
00103
00104 private:
00108     static const std::string _labels[LAST_VALUE];
00109
00113     static const char _typeLabels[LAST_VALUE];
00114
00115
00116 private:
00117     // ////////// Attributes //////////
00121     EN_SampleType _type;
00122 };
00123
00124 }
00125 #endif // __STDAIR_BASIC_SAMPLETYPE_HPP

```

### 33.125 stdair/basic/ServiceInitialisationType.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/ServiceInitialisationType.hpp>

```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.126 ServiceInitialisationType.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/stdair_exceptions.hpp>
00009 #include <stdair/basic/ServiceInitialisationType.hpp>

```

```

00010
00011 namespace stdair {
00012
00013 // //////////////////////////////////////
00014 const std::string ServiceInitialisationType::_labels[LAST_VALUE] =
00015     { "Not yet initialised", "File parsing", "Built-in sample BOM" };
00016
00017 // //////////////////////////////////////
00018 const char ServiceInitialisationType::_typeLabels[LAST_VALUE] =
00019     { 'N', 'F', 'B' };
00020
00021
00022 // //////////////////////////////////////
00023 ServiceInitialisationType::ServiceInitialisationType()
00024     : _type (LAST_VALUE) {
00025     assert (false);
00026 }
00027
00028 // //////////////////////////////////////
00029 ServiceInitialisationType::
00030 ServiceInitialisationType (const ServiceInitialisationType&
iServiceInitialisationType)
00031     : _type (iServiceInitialisationType._type) {
00032 }
00033
00034 // //////////////////////////////////////
00035 ServiceInitialisationType::
00036 ServiceInitialisationType (const EN_ServiceInitialisationType&
iServiceInitialisationType)
00037     : _type (iServiceInitialisationType) {
00038 }
00039
00040 // //////////////////////////////////////
00041 ServiceInitialisationType::EN_ServiceInitialisationType
00042 ServiceInitialisationType::getType (const char iTypeChar) {
00043     EN_ServiceInitialisationType oType;
00044     switch (iTypeChar) {
00045     case 'N': oType = NOT_YET_INITIALISED; break;
00046     case 'F': oType = FILE_PARSING; break;
00047     case 'B': oType = BUILTIN_SAMPLE; break;
00048     default: oType = LAST_VALUE; break;
00049     }
00050
00051     if (oType == LAST_VALUE) {
00052         const std::string& lLabels = describeLabels();
00053         std::ostringstream oMessage;
00054         oMessage << "The service initialisation type '" << iTypeChar
00055             << "' is not known. "
00056             << "Known service initialisation types: " << lLabels;
00057         throw CodeConversionException (oMessage.str());
00058     }
00059
00060     return oType;
00061 }
00062
00063 // //////////////////////////////////////
00064 ServiceInitialisationType::
00065 ServiceInitialisationType (const char iTypeChar)
00066     : _type (getType (iTypeChar)) {
00067 }
00068
00069 // //////////////////////////////////////
00070 ServiceInitialisationType::
00071 ServiceInitialisationType (const std::string& iTypeStr) {
00072     //
00073     const size_t lSize = iTypeStr.size();
00074     assert (lSize == 1);
00075     const char lTypeChar = iTypeStr[0];
00076     _type = getType (lTypeChar);
00077 }
00078
00079 // //////////////////////////////////////
00080 const std::string& ServiceInitialisationType::
00081 getLabel (const EN_ServiceInitialisationType& iType) {
00082     return _labels[iType];
00083 }
00084
00085 // //////////////////////////////////////
00086 char ServiceInitialisationType::
00087 getTypeLabel (const EN_ServiceInitialisationType& iType) {
00088     return _typeLabels[iType];
00089 }
00090
00091 // //////////////////////////////////////
00092 std::string ServiceInitialisationType::
00093 getTypeLabelAsString (const EN_ServiceInitialisationType
& iType) {

```

```

00094     std::ostringstream ostr;
00095     ostr << _typeLabels[iType];
00096     return ostr.str();
00097 }
00098
00099 // //////////////////////////////////////
00100 std::string ServiceInitialisationType::describeLabels() {
00101     std::ostringstream ostr;
00102     for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
00103         if (idx != 0) {
00104             ostr << ", ";
00105         }
00106         ostr << _labels[idx];
00107     }
00108     return ostr.str();
00109 }
00110
00111 // //////////////////////////////////////
00112 ServiceInitialisationType::EN_ServiceInitialisationType
00113 ServiceInitialisationType::getType() const {
00114     return _type;
00115 }
00116
00117 // //////////////////////////////////////
00118 char ServiceInitialisationType::getTypeAsChar() const {
00119     const char oTypeChar = _typeLabels[_type];
00120     return oTypeChar;
00121 }
00122
00123 // //////////////////////////////////////
00124 std::string ServiceInitialisationType::getTypeAsString() const
00125 {
00126     std::ostringstream ostr;
00127     ostr << _typeLabels[_type];
00128     return ostr.str();
00129 }
00130
00131 // //////////////////////////////////////
00131 const std::string ServiceInitialisationType::describe() const {
00132     std::ostringstream ostr;
00133     ostr << _labels[_type];
00134     return ostr.str();
00135 }
00136
00137 // //////////////////////////////////////
00138 bool ServiceInitialisationType::
00139 operator== (const EN_ServiceInitialisationType& iType) const {
00140     return (_type == iType);
00141 }
00142
00143 }

```

### 33.127 stdair/basic/ServiceInitialisationType.hpp File Reference

```

#include <string>
#include <stdair/basic/StructAbstract.hpp>

```

#### Classes

- struct [stdair::ServiceInitialisationType](#)  
*Enumeration of service initialisation types.*

#### Namespaces

- [stdair](#)  
*Handle on the StdAir library context.*

### 33.128 ServiceInitialisationType.hpp

```

00001 #ifndef __STDAIR_BAS_SERVICEINITIALISATIONTYPE_HPP
00002 #define __STDAIR_BAS_SERVICEINITIALISATIONTYPE_HPP

```

```

00003
00004 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00005 // Import section
00006 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/basic/StructAbstract.hpp>
00011
00012 namespace stdair {
00013
00017     struct ServiceInitialisationType : public
StructAbstract {
00018     public:
00019         typedef enum {
00020             NOT_YET_INITIALISED = 0,
00021             FILE_PARSING,
00022             BUILTIN_SAMPLE,
00023             LAST_VALUE
00024         } EN_ServiceInitialisationType;
00025
00030         static const std::string& getLabel (const
EN_ServiceInitialisationType&);
00031
00035         static EN_ServiceInitialisationType getType (const char);
00036
00040         static char getTypeLabel (const EN_ServiceInitialisationType&);
00041
00045         static std::string
getTypeLabelAsString (const EN_ServiceInitialisationType
00046         &);
00047
00051         static std::string describeLabels();
00052
00056         EN_ServiceInitialisationType getType() const;
00057
00061         char getTypeAsChar() const;
00062
00066         std::string getTypeAsString() const;
00067
00072         const std::string describe() const;
00073
00074     public:
00078         bool operator== (const EN_ServiceInitialisationType&) const;
00079
00080     public:
00084         ServiceInitialisationType (const
EN_ServiceInitialisationType&);
00088         ServiceInitialisationType (const char iType);
00092         ServiceInitialisationType (const std::string& iType);
00096         ServiceInitialisationType (const
ServiceInitialisationType&);
00097
00098     private:
00102         ServiceInitialisationType();
00103
00104
00105     private:
00109         static const std::string _labels[LAST_VALUE];
00113         static const char _typeLabels[LAST_VALUE];
00114
00115     private:
00116         // ////////// Attributes //////////
00120         EN_ServiceInitialisationType _type;
00121     };
00122 }
00123
00124 #endif // __STDAIR_BAS_SERVICEINITIALISATIONTYPE_HPP

```

### 33.129 stdair/basic/StructAbstract.hpp File Reference

```

#include <iosfwd>
#include <string>

```

#### Classes

- struct [stdair::StructAbstract](#)  
Base class for the light structures.

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## Functions

- `template<class charT , class traits >  
std::basic_ostream< charT, traits > & operator<< (std::basic_ostream< charT, traits > &ioOut, const  
stdair::StructAbstract &iStruct)`
- `template<class charT , class traits >  
std::basic_istream< charT, traits > & operator>> (std::basic_istream< charT, traits > &ioIn, stdair::StructAbstract &iStruct)`

### 33.129.1 Function Documentation

**33.129.1.1** `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](#).

**33.129.1.2** `template<class charT , class traits > std::basic_istream<charT, traits>& operator>> ( std::basic_istream< charT, traits > &ioIn, stdair::StructAbstract &iStruct ) [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\(\)](#).

### 33.130 StructAbstract.hpp

```
00001 #ifndef __STDAIR_BAS_STRUCTABSTRACT_HPP
00002 #define __STDAIR_BAS_STRUCTABSTRACT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010
00011 namespace stdair {
00012
00016     struct StructAbstract {
00017     public:
00018
00022         virtual ~StructAbstract() {}
00023
00029         void toStream (std::ostream& ioOut) const {
00030             ioOut << describe();
00031         }
00032
00038         virtual void fromStream (std::istream& ioIn) {}
00039
00043         virtual const std::string describe() const = 0;
00044
00045     protected:
00049         StructAbstract() {}
00050     };
00051 }
00052
00058 template <class charT, class traits>
00059 inline
00060 std::basic_ostream<charT, traits>&
```

```

00061 operator<< (std::basic_ostream<charT, traits>& ioOut,
00062             const stdair::StructAbstract& iStruct) {
00068     std::basic_ostringstream<charT,traits> ostr;
00069     ostr.copyfmt (ioOut);
00070     ostr.width (0);
00071
00072     // Fill string stream
00073     iStruct.toStream (ostr);
00074
00075     // Print string stream
00076     ioOut << ostr.str();
00077
00078     return ioOut;
00079 }
00080
00086 template <class charT, class traits>
00087 inline
00088 std::basic_istream<charT, traits>&
00089 operator>> (std::basic_istream<charT, traits>& ioIn,
00090             stdair::StructAbstract& ioStruct) {
00091     // Fill the Structure object with the input stream.
00092     ioStruct.fromStream (ioIn);
00093     return ioIn;
00094 }
00095 }
00096 #endif // __STDAIR_BAS_STRUCTABSTRACT_HPP

```

### 33.131 stdair/basic/UnconstrainingMethod.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/UnconstrainingMethod.hpp>

```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.132 UnconstrainingMethod.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/stdair_exceptions.hpp>
00009 #include <stdair/basic/UnconstrainingMethod.hpp>
00010
00011 namespace stdair {
00012
00013     // //////////////////////////////////////
00014     const std::string UnconstrainingMethod::_labels[LAST_VALUE] =
00015         { "Expectation-Maximisation" };
00016
00017     // //////////////////////////////////////
00018     const char UnconstrainingMethod::_methodLabels[LAST_VALUE] = { 'E' };
00019
00020
00021
00022     // //////////////////////////////////////
00023     UnconstrainingMethod::UnconstrainingMethod()
00024         : _method (LAST_VALUE) {
00025         assert (false);
00026     }
00027
00028     // //////////////////////////////////////
00029     UnconstrainingMethod::
00030     UnconstrainingMethod (const UnconstrainingMethod& iUnconstrainingMethod)
00031         : _method (iUnconstrainingMethod._method) {
00032     }
00033

```



```

00034 // //////////////////////////////////////
00035 UnconstrainingMethod:
00036 UnconstrainingMethod(const EN_UnconstrainingMethod& iUnconstrainingMethod)
00037 : _method(iUnconstrainingMethod) {
00038 }
00039
00040 // //////////////////////////////////////
00041 UnconstrainingMethod::UnconstrainingMethod(const char iMethod) {
00042     switch(iMethod) {
00043         case 'E': _method = EM; break;
00044         default: _method = LAST_VALUE; break;
00045     }
00046
00047     if(_method == LAST_VALUE) {
00048         const std::string& lLabels = describeLabels();
00049         std::ostringstream oMessage;
00050         oMessage << "The unconstraining method '" << iMethod
00051             << "' is not known. Known unconstraining methods: " << lLabels;
00052         throw CodeConversionException(oMessage.str());
00053     }
00054 }
00055
00056 // //////////////////////////////////////
00057 const std::string& UnconstrainingMethod::
00058 getLabel(const EN_UnconstrainingMethod& iMethod) {
00059     return _labels[iMethod];
00060 }
00061
00062 // //////////////////////////////////////
00063 char UnconstrainingMethod::getMethodLabel(const
EN_UnconstrainingMethod& iMethod) {
00064     return _methodLabels[iMethod];
00065 }
00066
00067 // //////////////////////////////////////
00068 std::string UnconstrainingMethod::
00069 getMethodLabelAsString(const EN_UnconstrainingMethod&
iMethod) {
00070     std::ostringstream oStr;
00071     oStr << _methodLabels[iMethod];
00072     return oStr.str();
00073 }
00074
00075 // //////////////////////////////////////
00076 std::string UnconstrainingMethod::describeLabels() {
00077     std::ostringstream ostr;
00078     for(unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
00079         if(idx != 0) {
00080             ostr << ", ";
00081         }
00082         ostr << _labels[idx] << " (" << _methodLabels[idx] << ")";
00083     }
00084     return ostr.str();
00085 }
00086
00087 // //////////////////////////////////////
00088 UnconstrainingMethod::EN_UnconstrainingMethod
UnconstrainingMethod::getMethod() const {
00089     return _method;
00090 }
00091
00092 // //////////////////////////////////////
00093 std::string UnconstrainingMethod::getMethodAsString() const {
00094     std::ostringstream oStr;
00095     oStr << _methodLabels[_method];
00096     return oStr.str();
00097 }
00098
00099 // //////////////////////////////////////
00100 const std::string UnconstrainingMethod::describe() const {
00101     std::ostringstream ostr;
00102     ostr << _labels[_method];
00103     return ostr.str();
00104 }
00105
00106 // //////////////////////////////////////
00107 bool UnconstrainingMethod::
00108 operator==(const EN_UnconstrainingMethod& iMethod) const {
00109     return (_method == iMethod);
00110 }
00111
00112 }

```

## 33.133 stdair/basic/UnconstrainingMethod.hpp File Reference

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

## Classes

- struct [stdair::UnconstrainingMethod](#)

## Namespaces

- [stdair](#)  
*Handle on the StdAir library context.*

## 33.134 UnconstrainingMethod.hpp

```
00001 #ifndef __STDAIR_BAS_UNCONSTRAININGMETHOD_HPP
00002 #define __STDAIR_BAS_UNCONSTRAININGMETHOD_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/basic/StructAbstract.hpp>
00011
00012 namespace stdair {
00013
00015     struct UnconstrainingMethod : public StructAbstract {
00016     public:
00017         typedef enum {
00018             EM = 0,
00019             LAST_VALUE
00020         } EN_UnconstrainingMethod;
00021
00023         static const std::string& getLabel (const EN_UnconstrainingMethod&);
00024
00026         static char getMethodLabel (const EN_UnconstrainingMethod&);
00027
00029         static std::string getMethodLabelAsString (const
00030             EN_UnconstrainingMethod&);
00032         static std::string describeLabels ();
00033
00035         EN_UnconstrainingMethod getMethod() const;
00036
00038         std::string getMethodAsString() const;
00039
00042         const std::string describe() const;
00043
00044     public:
00046         bool operator== (const EN_UnconstrainingMethod&) const;
00047
00048     public:
00050         UnconstrainingMethod (const EN_UnconstrainingMethod&);
00052         UnconstrainingMethod (const char iMethod);
00054         UnconstrainingMethod (const UnconstrainingMethod&);
00055
00056     private:
00058         UnconstrainingMethod();
00059
00060
00061     private:
00063         static const std::string _labels[LAST_VALUE];
00065         static const char _methodLabels[LAST_VALUE];
00066
00067
00068     private:
00069         // ////////// Attributes //////////
00071         EN_UnconstrainingMethod _method;
00072     };
00073
00074 }
00075 #endif // __STDAIR_BAS_UNCONSTRAININGMETHOD_HPP
```

### 33.135 stdair/basic/YieldRange.cpp File Reference

```
#include <limits>
#include <sstream>
#include <stdair/basic/YieldRange.hpp>
```

#### Namespaces

- **stdair**

*Handle on the StdAir library context.*

### 33.136 YieldRange.cpp

```
00001 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00002 // Import section
00003 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <limits>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/YieldRange.hpp>
00009
00010 namespace stdair {
00011
00012 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00013 YieldRange::YieldRange() :
00014     _upperYield (std::numeric_limits<Yield_T>::max()),
00015     _averageYield (std::numeric_limits<Yield_T>::max()),
00016     _lowerYield (std::numeric_limits<Yield_T>::min()) {
00017 }
00018
00019 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00020 YieldRange::YieldRange (const YieldRange& iYieldRange) :
00021     _upperYield (iYieldRange.getUpperYield()),
00022     _averageYield (iYieldRange.getAverageYield()),
00023     _lowerYield (std::numeric_limits<Yield_T>::min()) {
00024 }
00025
00026 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00027 YieldRange::YieldRange (const Yield_T iUpperYield) :
00028     _upperYield (iUpperYield), _averageYield (iUpperYield),
00029     _lowerYield (iUpperYield) {
00030 }
00031
00032 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00033 YieldRange::YieldRange (const Yield_T iUpperYield,
00034                         const Yield_T iAverageYield) :
00035     _upperYield (iUpperYield), _averageYield (iAverageYield),
00036     _lowerYield (std::numeric_limits<Yield_T>::min()) {
00037 }
00038
00039 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00040 YieldRange::YieldRange (const Yield_T iUpperYield,
00041                         const Yield_T iAverageYield,
00042                         const Yield_T iLowerYield) :
00043     _upperYield (iUpperYield), _averageYield (iAverageYield),
00044     _lowerYield (iLowerYield) {
00045 }
00046
00047 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00048 YieldRange::~YieldRange() {
00049 }
00050
00051 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00052 void YieldRange::toStream (std::ostream& ioOut) const {
00053     ioOut << _averageYield << "[" << _lowerYield << ", "
00054         << _upperYield << "]";
00055 }
00056
00057 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00058 void YieldRange::fromStream (std::istream& ioIn) {
00059 }
00060
00061 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00062 const std::string YieldRange::describe() const {
00063     std::ostringstream oStr;
00064 }
```

```

00065     return oStr.str();
00066 }
00067
00068 }

```

### 33.137 stdair/basic/YieldRange.hpp File Reference

```

#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/StructAbstract.hpp>

```

#### Classes

- class [stdair::YieldRange](#)

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.138 YieldRange.hpp

```

00001 #ifndef __STDAIR_BAS_YIELDRANGE_HPP
00002 #define __STDAIR_BAS_YIELDRANGE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STDAIR
00008 #include <stdair/stdair_inventory_types.hpp>
00009 #include <stdair/basic/StructAbstract.hpp>
00010
00011 namespace stdair {
00012
00023 class YieldRange : public StructAbstract {
00024 public:
00026     YieldRange ();
00027     YieldRange (const YieldRange&);
00028     YieldRange (const Yield_T iUpperYield);
00029     YieldRange (const Yield_T iUpperYield, const Yield_T iAverageYield);
00030     YieldRange (const Yield_T iUpperYield, const Yield_T iAverageYield,
00031               const Yield_T iLowerYield);
00032
00034     virtual ~YieldRange();
00035
00036
00037     // //////////// Getters ////////////
00039     Yield_T getUpperYield() const {
00040         return _upperYield;
00041     }
00043     Yield_T getAverageYield() const {
00044         return _averageYield;
00045     }
00047     Yield_T getLowerYield() const {
00048         return _lowerYield;
00049     }
00050
00051     // //////////// Setters ////////////
00053     void setUpperYield (const Yield_T iUpperYield) {
00054         _upperYield = iUpperYield;
00055     }
00057     void setAverageYield (const Yield_T iAverageYield) {
00058         _averageYield = iAverageYield;
00059     }
00061     void setLowerYield (const Yield_T iLowerYield) {
00062         _lowerYield = iLowerYield;
00063     }
00064
00065
00066     // //////////// Display methods ////////////
00069     void toStream (std::ostream&) const;
00070
00073     void fromStream (std::istream&);

```

```

00074
00076     const std::string describe() const;
00077
00078 private:
00079     // ////////// Attributes //////////
00081     Yield_T _upperYield;
00082
00084     Yield_T _averageYield;
00085
00087     Yield_T _lowerYield;
00088 };
00089 }
00090 #endif // __STDAIR_BAS_YIELDRANGE_HPP

```

### 33.139 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

- [stdair](#)

*Handle on the StdAir library context.*

### 33.140 AirlineClassList.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/access.hpp>
00011 // StdAir
00012 #include <stdair/basic/BasConst_Inventory.hpp>
00013 #include <stdair/service/Logger.hpp>
00014 #include <stdair/bom/AirlineClassList.hpp>
00015
00016 namespace stdair {
00017
00018     // //////////////////////////////////////
00019     AirlineClassList::AirlineClassList()
00020         : _key (DEFAULT_AIRLINE_CODE_LIST,
00021             DEFAULT_CLASS_CODE_LIST),
00022         _parent (NULL) {
00023         assert (false);
00024     }
00025
00026     // //////////////////////////////////////
00027     AirlineClassList::AirlineClassList (const AirlineClassList& iACL)
00028         : _key (iACL._key),
00029         _parent (NULL),
00030         _yield(iACL._yield),
00031         _fare(iACL._fare) {
00032     }
00033
00034     // //////////////////////////////////////
00035     AirlineClassList::AirlineClassList (const Key_T& iKey)
00036         : _key (iKey), _parent (NULL) {
00037     }
00038
00039     // //////////////////////////////////////
00039     AirlineClassList::~AirlineClassList () {

```

```

00040     }
00041
00042     // //////////////////////////////////////
00043     std::string AirlineClassList::toString() const {
00044         std::ostringstream ostr;
00045         ostr << describeKey() << ", " << _yield << ", " << _fare;
00046         return ostr.str();
00047     }
00048
00049     // //////////////////////////////////////
00050     void AirlineClassList::serialisationImplementationExport() const {
00051         std::ostringstream ostr;
00052         boost::archive::text_oarchive oa (ostr);
00053         oa << *this;
00054     }
00055
00056     // //////////////////////////////////////
00057     void AirlineClassList::serialisationImplementationImport() {
00058         std::istringstream istr;
00059         boost::archive::text_iarchive ia (istr);
00060         ia >> *this;
00061     }
00062
00063     // //////////////////////////////////////
00064     template<class Archive>
00065     void AirlineClassList::serialize (Archive& ioArchive,
00066                                     const unsigned int iFileVersion) {
00067         ioArchive & _key & _yield & _fare;
00068     }
00069
00070 }
00071
00072
00073

```

### 33.141 stdair/bom/AirlineClassList.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/AirlineClassListKey.hpp>
#include <stdair/bom/AirlineClassListTypes.hpp>

```

#### Classes

- class [stdair::AirlineClassList](#)  
*Class representing the actual attributes for a segment-features.*

#### Namespaces

- [boost](#)  
*Forward declarations.*
- [boost::serialization](#)
- [stdair](#)  
*Handle on the StdAir library context.*

### 33.142 AirlineClassList.hpp

```

00001 #ifndef __STDAIR_BOM_AIRLINECLASSLIST_HPP
00002 #define __STDAIR_BOM_AIRLINECLASSLIST_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir

```

```

00011 #include <stdair/bom/BomAbstract.hpp>
00012 #include <stdair/bom/AirlineClassListKey.hpp>
00013 #include <stdair/bom/AirlineClassListTypes.hpp>
00014
00016 namespace boost {
00017     namespace serialization {
00018         class access;
00019     }
00020 }
00021
00022 namespace stdair {
00023
00027     class AirlineClassList : public BomAbstract {
00028     public:
00028         template <typename BOM> friend class FacBom;
00029         template <typename BOM> friend class FacCloneBom;
00030         friend class FacBomManager;
00031         friend class boost::serialization::access;
00032
00033     public:
00034         // ////////// Type definitions //////////
00038         typedef AirlineClassListKey Key_T;
00039
00040
00041     public:
00042         // ////////// Getters //////////
00044         const Key_T& getKey() const {
00045             return _key;
00046         }
00047
00049         BomAbstract* const getParent() const {
00050             return _parent;
00051         }
00052
00054         const AirlineCodeList_T& getAirlineCodeList() const {
00055             return _key.getAirlineCodeList();
00056         }
00057
00059         const ClassList_StringList_T& getClassCodeList() const {
00060             return _key.getClassCodeList();
00061         }
00062
00064         const HolderMap_T& getHolderMap() const {
00065             return _holderMap;
00066         }
00067
00069         const stdair::Yield_T& getYield() const {
00070             return _yield;
00071         }
00072
00074         const stdair::Fare_T& getFare() const {
00075             return _fare;
00076         }
00077
00078     public:
00079         // ////////// Setters //////////
00080         void setYield (const Yield_T& iYield) {
00081             _yield = iYield;
00082         }
00083
00084         void setFare (const Fare_T& iFare) {
00085             _fare = iFare;
00086         }
00087
00088     public:
00089         // ////////// Display support methods //////////
00095         void toStream (std::ostream& ioOut) const {
00096             ioOut << toString();
00097         }
00098
00104         void fromStream (std::istream& ioIn) {
00105         }
00106
00110         std::string toString() const;
00111
00115         const std::string describeKey() const {
00116             return _key.toString();
00117         }
00118
00119
00120     public:
00121         // ////////// (Boost) Serialisation support methods //////////
00125         template<class Archive>
00126         void serialize (Archive& ar, const unsigned int iFileVersion);
00127
00128     private:
00133         void serialisationImplementationExport() const;
00134         void serialisationImplementationImport();

```

```

00135
00136
00137 protected:
00138     // ////////// Constructors and destructors //////////
00142     AirlineClassList (const Key_T&);
00146     virtual ~AirlineClassList();
00147
00148 private:
00152     AirlineClassList();
00153
00157     AirlineClassList (const AirlineClassList&);
00158
00159
00160 protected:
00161     // ////////// Attributes //////////
00165     Key_T _key;
00166
00170     BomAbstract* _parent;
00171
00175     HolderMap_T _holderMap;
00176
00177     /*
00178      * Yield value.
00179      */
00180     Yield_T _yield;
00181
00182     /*
00183      * Fare value.
00184      */
00185     Fare_T _fare;
00186 };
00187
00188 }
00189 #endif // __STDAIR_BOM_AIRLINECLASSLIST_HPP
00190

```

### 33.143 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

- [stdair](#)  
Handle on the StdAir library context.

#### Functions

- template void [stdair::AirlineClassListKey::serialize< ba::text\\_oarchive >](#) (ba::text\_oarchive &, unsigned int)
- template void [stdair::AirlineClassListKey::serialize< ba::text\\_iarchive >](#) (ba::text\_iarchive &, unsigned int)

### 33.144 AirlineClassListKey.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/access.hpp>
00011 // StdAir

```



```

00012 #include <stdair/basic/BasConst_BomDisplay.hpp>
00013 #include <stdair/bom/AirlineClassListKey.hpp>
00014
00015 namespace stdair {
00016
00017 // //////////////////////////////////////
00018 AirlineClassListKey::AirlineClassListKey() {
00019     assert (false);
00020 }
00021
00022 // //////////////////////////////////////
00023 AirlineClassListKey::
00024 AirlineClassListKey (const AirlineCodeList_T& iAirlineCodeList,
00025                     const ClassList_StringList_T& iClassCodeList)
00026 : _airlineCodeList (iAirlineCodeList), _classCodeList (iClassCodeList) {
00027 }
00028
00029 // //////////////////////////////////////
00030 AirlineClassListKey::AirlineClassListKey (const AirlineClassListKey& iKey)
00031 : _airlineCodeList (iKey._airlineCodeList),
00032   _classCodeList (iKey._classCodeList) {
00033 }
00034
00035 // //////////////////////////////////////
00036 AirlineClassListKey::~AirlineClassListKey() {
00037 }
00038
00039 // //////////////////////////////////////
00040 void AirlineClassListKey::toStream (std::ostream& ioOut) const {
00041     ioOut << "AirlineClassListKey: " << toString() << std::endl;
00042 }
00043
00044 // //////////////////////////////////////
00045 void AirlineClassListKey::fromStream (std::istream& ioIn) {
00046 }
00047
00048 // //////////////////////////////////////
00049 const std::string AirlineClassListKey::toString() const {
00050     std::ostringstream oStr;
00051     assert (_airlineCodeList.size() == _classCodeList.size());
00052
00053     unsigned short idx = 0;
00054     AirlineCodeList_T::const_iterator itAirlineCode = _airlineCodeList.begin();
00055     for (ClassList_StringList_T::const_iterator itClassCode =
00056          _classCodeList.begin(); itClassCode != _classCodeList.end();
00057          ++itClassCode, ++itAirlineCode, ++idx) {
00058         if (idx != 0) {
00059             oStr << DEFAULT_KEY_SUB_FLD_DELIMITER << " ";
00060         }
00061
00062         const AirlineCode_T& lAirlineCode = *itAirlineCode;
00063         const ClassCode_T& lClassCode = *itClassCode;
00064         oStr << lAirlineCode << " " << lClassCode;
00065     }
00066
00067     return oStr.str();
00068 }
00069
00070 // //////////////////////////////////////
00071 void AirlineClassListKey::serialisationImplementationExport() const {
00072     std::ostringstream oStr;
00073     boost::archive::text_oarchive oa (oStr);
00074     oa << *this;
00075 }
00076
00077 // //////////////////////////////////////
00078 void AirlineClassListKey::serialisationImplementationImport() {
00079     std::istringstream iStr;
00080     boost::archive::text_iarchive ia (iStr);
00081     ia >> *this;
00082 }
00083
00084 // //////////////////////////////////////
00085 template<class Archive>
00086 void AirlineClassListKey::serialize (Archive& ioArchive,
00087                                     const unsigned int iFileVersion) {
00088     AirlineCodeList_T::const_iterator itAirlineCode = _airlineCodeList.begin();
00089     for (ClassList_StringList_T::const_iterator itClassCode =
00090          _classCodeList.begin(); itClassCode != _classCodeList.end();
00091          ++itClassCode, ++itAirlineCode) {
00092         AirlineCode_T lAirlineCode = *itAirlineCode;
00093         ClassCode_T lClassCode = *itClassCode;
00094
00095         ioArchive & lAirlineCode & lClassCode;
00096     }
00097 }
00098
00099 }
00100
00101
00102

```

```

00107 // //////////////////////////////////////
00108 // Explicit template instantiation
00109 namespace ba = boost::archive;
00110 template void AirlineClassListKey::
00111     serialize<ba::text_oarchive> (ba::text_oarchive&, unsigned int);
00112 template void AirlineClassListKey::
00113     serialize<ba::text_iarchive> (ba::text_iarchive&, unsigned int);
00114 // //////////////////////////////////////
00115
00116 }

```

### 33.145 stdair/bom/AirlineClassListKey.hpp File Reference

```

#include <iosfwd>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>

```

#### Classes

- struct [stdair::AirlineClassListKey](#)

*Key of airport-pair.*

#### Namespaces

- [boost](#)

*Forward declarations.*

- [boost::serialization](#)
- [stdair](#)

*Handle on the StdAir library context.*

### 33.146 AirlineClassListKey.hpp

```

00001 #ifndef __STDAIR_BOM_AIRLINECLASSLISTKEY_HPP
00002 #define __STDAIR_BOM_AIRLINECLASSLISTKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 // StdAir
00010 #include <stdair/stdair_inventory_types.hpp>
00011 #include <stdair/bom/KeyAbstract.hpp>
00012
00014 namespace boost {
00015     namespace serialization {
00016         class access;
00017     }
00018 }
00019
00020 namespace stdair {
00021
00025     struct AirlineClassListKey : public KeyAbstract {
00026         friend class boost::serialization::access;
00027
00028         // ////////////////////////////////// Constructors and destructors //////////////////////////////////
00029     private:
00033         AirlineClassListKey();
00034
00035     public:
00039         AirlineClassListKey (const AirlineCodeList_T&,
00040                             const ClassList_StringList_T&);
00041
00045         AirlineClassListKey (const AirlineClassListKey&);
00046
00050         ~AirlineClassListKey();
00051
00052
00053     public:

```

```

00054 // //////////// Getters ////////////
00055 const AirlineCodeList_T& getAirlineCodeList() const {
00056     return _airlineCodeList;
00057 }
00058
00059 const ClassList_StringList_T& getClassCodeList() const {
00060     return _classCodeList;
00061 }
00062
00063 public:
00064 // //////////// Display support methods ////////////
00065 void toStream (std::ostream& ioOut) const;
00066
00067 void fromStream (std::istream& ioIn);
00068
00069 const std::string toString() const;
00070
00071 public:
00072 // //////////// (Boost) Serialisation support methods ////////////
00073 template<class Archive>
00074 void serialize (Archive& ar, const unsigned int iFileVersion);
00075
00076 private:
00077 void serialisationImplementationExport() const;
00078 void serialisationImplementationImport();
00079
00080 private:
00081 // //////////// Attributes ////////////
00082 AirlineCodeList_T _airlineCodeList;
00083
00084 ClassList_StringList_T _classCodeList;
00085 };
00086
00087 }
00088
00089 #endif // __STDAIR_BOM_AIRLINECLASSLISTKEY_HPP

```

### 33.147 stdair/bom/AirlineClassListTypes.hpp File Reference

```

#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>

```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

#### Typedefs

- typedef std::list< AirlineClassList \* > [stdair::AirlineClassListList\\_T](#)
- typedef std::map< const MapKey\_T, AirlineClassList \* > [stdair::AirlineClassListMap\\_T](#)
- typedef std::pair< MapKey\_T, AirlineClassList \* > [stdair::AirlineClassListWithKey\\_T](#)
- typedef std::list< AirlineClassListWithKey\_T > [stdair::AirlineClassListDetailedList\\_T](#)

### 33.148 AirlineClassListTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_AIRLINECLASSLISTTYPES_HPP
00003 #define __STDAIR_BOM_AIRLINECLASSLISTTYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // STDAIR

```

```

00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016     // Forward declarations.
00017     class AirlineClassList;
00018
00020     typedef std::list<AirlineClassList*> AirlineClassListList_T;
00021
00023     typedef std::map<const MapKey_T, AirlineClassList*> AirlineClassListMap_T;
00024
00026     typedef std::pair<MapKey_T, AirlineClassList*> AirlineClassListWithKey_T;
00027     typedef std::list<AirlineClassListWithKey_T> AirlineClassListDetailedList_T;
00028 }
00029 #endif // __STDAIR_BOM_AIRLINECLASSLISTTYPES_HPP

```

### 33.149 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

- [stdair](#)

*Handle on the StdAir library context.*

### 33.150 AirlineFeature.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // StdAir
00007 #include <stdair/stdair_types.hpp>
00008 #include <stdair/basic/BasConst_Inventory.hpp>
00009 #include <stdair/bom/AirlineFeature.hpp>
00010
00011 namespace stdair {
00012
00013     // //////////////////////////////////////
00014     AirlineFeature::AirlineFeature (const Key_T& iKey) :
00015         _key (iKey),
00016         _forecastingMethod (DEFAULT_FORECASTING_METHOD),
00017         _unconstrainingMethod (DEFAULT_UNCONSTRAINING_METHOD),
00018         _preOptimisationMethod (DEFAULT_PREOPTIMISATION_METHOD),
00019         _optimisationMethod (DEFAULT_OPTIMISATION_METHOD),
00020         _partnershipTechnique (DEFAULT_PARTNERSHIP_TECHNIQUE) {
00021     }
00022
00023     // //////////////////////////////////////
00024     AirlineFeature::AirlineFeature (const AirlineFeature& iAirlineFeature) :
00025         _key (iAirlineFeature._key),
00026         _forecastingMethod (iAirlineFeature._forecastingMethod),
00027         _unconstrainingMethod (iAirlineFeature._unconstrainingMethod),
00028         _preOptimisationMethod (iAirlineFeature._preOptimisationMethod),
00029         _optimisationMethod (iAirlineFeature._optimisationMethod),
00030         _partnershipTechnique (iAirlineFeature._partnershipTechnique) {
00031     }
00032
00033     // //////////////////////////////////////
00034     AirlineFeature::~AirlineFeature () {
00035     }
00036
00037     // //////////////////////////////////////
00038     void AirlineFeature::init (const ForecastingMethod&
00039                               iForecastingMethod,
00040                               const UnconstrainingMethod& iUnconstrainingMethod,
00041                               const PreOptimisationMethod& iPreOptimisationMethod,
00042                               const OptimisationMethod& iOptimisationMethod,
00043                               const HistoricalDataLimit_T& iHistoricalDataLimit,

```

```

00043         const ControlMode_T& iControlMode,
00044         const PartnershipTechnique& iPartnershipTechnique) {
00045     _forecastingMethod = iForecastingMethod;
00046     _unconstrainingMethod = iUnconstrainingMethod;
00047     _preOptimisationMethod = iPreOptimisationMethod;
00048     _optimisationMethod = iOptimisationMethod;
00049     _historicalDataLimit = iHistoricalDataLimit;
00050     _controlMode = iControlMode;
00051     _partnershipTechnique = iPartnershipTechnique;
00052 }
00053
00054 // //////////////////////////////////////
00055 std::string AirlineFeature::toString() const {
00056     std::ostringstream ostr;
00057     ostr << describeKey()
00058         << ", " << _forecastingMethod
00059         << ", " << _unconstrainingMethod
00060         << ", " << _preOptimisationMethod
00061         << ", " << _optimisationMethod
00062         << ", " << _historicalDataLimit
00063         //<< ", " << _controlMode
00064         << ", " << _partnershipTechnique;
00065     return ostr.str();
00066 }
00067
00068 }
00069

```

### 33.151 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>

```

#### 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.*

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.152 AirlineFeature.hpp

```

00001 #ifndef __STDAIR_BOM_AIRLINEFEATURE_HPP
00002 #define __STDAIR_BOM_AIRLINEFEATURE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_rm_types.hpp>
00009 #include <stdair/basic/UnconstrainingMethod.hpp>
00010 #include <stdair/basic/ForecastingMethod.hpp>
00011 #include <stdair/basic/PreOptimisationMethod.hpp>
00012 #include <stdair/basic/OptimisationMethod.hpp>
00013 #include <stdair/basic/PartnershipTechnique.hpp>
00014 #include <stdair/bom/BomAbstract.hpp>

```

```

00015 #include <stdair/bom/AirlineFeatureKey.hpp>
00016 #include <stdair/bom/AirlineFeatureTypes.hpp>
00017
00018 namespace stdair {
00019
00025     class AirlineFeature : public BomAbstract {
00026     template <typename BOM> friend class FacBom;
00027     template <typename BOM> friend class FacCloneBom;
00028     friend class FacBomManager;
00029
00030     public:
00031         // //////////// Type definitions ////////////
00035         typedef AirlineFeatureKey Key_T;
00036
00037     public:
00038         // //////////// Display support methods ////////////
00044         void toStream (std::ostream& ioOut) const {
00045             ioOut << toString();
00046         }
00047
00053         void fromStream (std::istream& ioIn) {
00054             }
00055
00059         std::string toString() const;
00060
00064         const std::string describeKey() const {
00065             return _key.toString();
00066         }
00067
00068     public:
00069         // //////////// Getters ////////////
00073         const Key_T& getKey() const {
00074             return _key;
00075         }
00076
00080         BomAbstract* const getParent() const {
00081             return _parent;
00082         }
00083
00087         const HolderMap_T& getHolderMap() const {
00088             return _holderMap;
00089         }
00090
00094         ForecastingMethod::EN_ForecastingMethod
getForecastingMethod() const {
00095             return _forecastingMethod.getMethod();
00096         }
00097
00101         UnconstrainingMethod::EN_UnconstrainingMethod
getUnconstrainingMethod() const {
00102             return _unconstrainingMethod.getMethod();
00103         }
00104
00108         PartnershipTechnique::EN_PartnershipTechnique
getPartnershipTechnique() const {
00109             return _partnershipTechnique.getTechnique();
00110         }
00111
00115         PreOptimisationMethod::EN_PreOptimisationMethod
getPreOptimisationMethod() const {
00116             return _preOptimisationMethod.getMethod();
00117         }
00118
00122         OptimisationMethod::EN_OptimisationMethod
getOptimisationMethod() const {
00123             return _optimisationMethod.getMethod();
00124         }
00125
00126     public:
00127         // //////////// Setters ////////////
00139         void init (const ForecastingMethod&,
00140                 const UnconstrainingMethod&,
00141                 const PreOptimisationMethod&,
00142                 const OptimisationMethod&,
00143                 const HistoricalDataLimit_T&,
00144                 const ControlMode_T&,
00145                 const PartnershipTechnique&);
00146
00150         void setForecastingMethod (const ForecastingMethod&
iForecastingMethod) {
00151             _forecastingMethod = iForecastingMethod;
00152         }
00153
00157         void setUnconstrainingMethod(const
UnconstrainingMethod& iUnconstrainingMethod) {
00158             _unconstrainingMethod = iUnconstrainingMethod;

```

```

00159     }
00160
00164     void setPartnershipTechnique(const
PartnershipTechnique& iPartnershipTechnique) {
00165         _partnershipTechnique = iPartnershipTechnique;
00166     }
00167
00171     void setPreOptimisationMethod(const
PreOptimisationMethod& iPreOptimisationMethod) {
00172         _preOptimisationMethod = iPreOptimisationMethod;
00173     }
00174
00178     void setOptimisationMethod(const OptimisationMethod&
iOptimisationMethod) {
00179         _optimisationMethod = iOptimisationMethod;
00180     }
00181
00182
00183 protected:
00184     // ////////// Constructors and destructors //////////
00188     AirlineFeature (const Key_T&);
00192     virtual ~AirlineFeature ();
00193
00194 private:
00198     AirlineFeature ();
00202     AirlineFeature (const AirlineFeature&);
00203
00204 protected:
00205     // ////////// Attributes //////////
00209     Key_T _key;
00210
00214     BomAbstract* _parent;
00215
00219     HolderMap_T _holderMap;
00220
00224     ForecastingMethod _forecastingMethod;
00225
00229     HistoricalDataLimit_T _historicalDataLimit;
00230
00234     ControlMode_T _controlMode;
00235
00239     UnconstrainingMethod _unconstrainingMethod;
00240
00244     PreOptimisationMethod _preOptimisationMethod;
00245
00249     OptimisationMethod _optimisationMethod;
00250
00254     PartnershipTechnique _partnershipTechnique;
00255
00256 };
00257
00258 }
00259 #endif // __STDAIR_BOM_AIRLINEFEATURE_HPP
00260

```

### 33.153 stdair/bom/AirlineFeatureKey.cpp File Reference

```

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

```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.154 AirlineFeatureKey.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <sstream>
00006 // StdAir
00007 #include <stdair/bom/AirlineFeatureKey.hpp>
00008

```

```

00009 namespace stdair {
00010
00011 // //////////////////////////////////////
00012 AirlineFeatureKey::AirlineFeatureKey (const
AirlineCode_T& iAirlineCode)
00013 : _airlineCode (iAirlineCode) {
00014 }
00015
00016 // //////////////////////////////////////
00017 AirlineFeatureKey::~AirlineFeatureKey () {
00018 }
00019
00020 // //////////////////////////////////////
00021 void AirlineFeatureKey::toStream (std::ostream& ioOut) const {
00022     ioOut << "AirlineFeatureKey: " << toString() << std::endl;
00023 }
00024
00025 // //////////////////////////////////////
00026 void AirlineFeatureKey::fromStream (std::istream& ioIn) {
00027 }
00028
00029 // //////////////////////////////////////
00030 const std::string AirlineFeatureKey::toString() const {
00031     std::ostringstream oStr;
00032     oStr << _airlineCode;
00033     return oStr.str();
00034 }
00035
00036 }

```

### 33.155 stdair/bom/AirlineFeatureKey.hpp File Reference

```

#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>

```

#### Classes

- struct [stdair::AirlineFeatureKey](#)

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.156 AirlineFeatureKey.hpp

```

00001 #ifndef __STDAIR_BOM_AIRLINEFEATUREKEY_HPP
00002 #define __STDAIR_BOM_AIRLINEFEATUREKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/stdair_inventory_types.hpp>
00011 #include <stdair/bom/KeyAbstract.hpp>
00012
00013 namespace stdair {
00015     struct AirlineFeatureKey : public KeyAbstract {
00016
00017     public:
00018         // ////////////////////////////////// Construction //////////////////////////////////
00020         AirlineFeatureKey (const AirlineCode_T& iAirlineCode);
00021
00023         ~AirlineFeatureKey ();
00024
00025         // ////////////////////////////////// Getters //////////////////////////////////
00027         const AirlineCode_T& getAirlineCode() const { return _airlineCode; }
00028

```



```

00029 // ////////// Display support methods //////////
00032 void toStream (std::ostream& ioOut) const;
00033
00036 void fromStream (std::istream& ioIn);
00037
00043 const std::string toString() const;
00044
00045 private:
00046 // Attributes
00048 AirlineCode_T _airlineCode;
00049 };
00050
00051 }
00052 #endif // __STDAIR_BOM_AIRLINEFEATUREKEY_HPP

```

### 33.157 stdair/bom/AirlineFeatureTypes.hpp File Reference

```

#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>

```

#### Namespaces

- [stdair](#)  
Handle on the StdAir library context.

#### Typedefs

- typedef std::list< AirlineFeature \* > [stdair::AirlineFeatureList\\_T](#)
- typedef std::map< const MapKey\_T, AirlineFeature \* > [stdair::AirlineFeatureMap\\_T](#)

### 33.158 AirlineFeatureTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_AIRLINEFEATURETYPES_HPP
00003 #define __STDAIR_BOM_AIRLINEFEATURETYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // StdAir
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016 // Forward declarations.
00017 class AirlineFeature;
00018
00020 typedef std::list<AirlineFeature*> AirlineFeatureList_T;
00021
00023 typedef std::map<const MapKey_T, AirlineFeature*> AirlineFeatureMap_T;
00024
00025 }
00026 #endif // __STDAIR_BOM_AIRLINEFEATURETYPES_HPP
00027

```

### 33.159 stdair/bom/AirlineStruct.cpp File Reference

```

#include <cassert>
#include <istream>
#include <ostream>
#include <sstream>
#include <stdair/bom/AirlineStruct.hpp>

```

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.160 AirlineStruct.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <istream>
00007 #include <ostream>
00008 #include <sstream>
00009 // StdAir
00010 #include <stdair/bom/AirlineStruct.hpp>
00011
00012 namespace stdair {
00013
00014 // //////////////////////////////////////
00015 AirlineStruct::AirlineStruct () {
00016 }
00017
00018 // //////////////////////////////////////
00019 AirlineStruct::AirlineStruct (const AirlineStruct&
iAirlineStruct)
00020 : _code (iAirlineStruct._code), _name (iAirlineStruct._name) {
00021 }
00022
00023 // //////////////////////////////////////
00024 AirlineStruct::AirlineStruct (const AirlineCode_T& iAirlineCode,
const std::string& iAirlineName)
00025 : _code (iAirlineCode), _name (iAirlineName) {
00026 }
00027
00028 // //////////////////////////////////////
00029 AirlineStruct::~AirlineStruct () {
00030 }
00031
00032 // //////////////////////////////////////
00033 void AirlineStruct::toStream (std::ostream& ioOut) const {
00034 ioOut << describe();
00035 }
00036
00037 // //////////////////////////////////////
00038 void AirlineStruct::fromStream (std::istream& ioIn) {
00039 }
00040
00041 // //////////////////////////////////////
00042 const std::string AirlineStruct::describe() const {
00043 std::ostringstream oStr;
00044 oStr << _code << " " << _name;
00045 return oStr.str();
00046 }
00047
00048
00049 }

```

## 33.161 stdair/bom/AirlineStruct.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <vector>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/StructAbstract.hpp>

```

## Classes

- struct [stdair::AirlineStruct](#)

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.162 AirlineStruct.hpp

```

00001 #ifndef __STDAIR_BOM_AIRLINESTRUCT_HPP
00002 #define __STDAIR_BOM_AIRLINESTRUCT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 #include <vector>
00011 // StdAir
00012 #include <stdair/stdair_inventory_types.hpp>
00013 #include <stdair/basic/StructAbstract.hpp>
00014
00015 namespace stdair {
00016
00017     struct AirlineStruct : public StructAbstract {
00018     public:
00019         // ////////////////////////////////// Getters //////////////////////////////////
00020         const AirlineCode_T& getAirlineCode() const {
00021             return _code;
00022         }
00023
00024         const std::string& getAirlineName() const {
00025             return _name;
00026         }
00027
00028         // ////////////////////////////////// Setters //////////////////////////////////
00029         void setAirlineCode (const AirlineCode_T& iAirlineCode) {
00030             _code = iAirlineCode;
00031         }
00032
00033         void setAirlineName (const std::string& iAirlineName) {
00034             _name = iAirlineName;
00035         }
00036
00037     public:
00038         // ////////////////////////////////// Display support method //////////////////////////////////
00039         void toStream (std::ostream& ioOut) const;
00040
00041         void fromStream (std::istream& ioIn);
00042
00043         const std::string describe() const;
00044
00045     public:
00046         // ////////////////////////////////// Constructors & Destructor //////////////////////////////////
00047         AirlineStruct (const AirlineCode_T&, const std::string& iAirlineName);
00048         AirlineStruct ();
00049         AirlineStruct (const AirlineStruct&);
00050         ~AirlineStruct ();
00051
00052     private:
00053         // ////////////////////////////////// Attributes //////////////////////////////////
00054         AirlineCode_T _code;
00055
00056         std::string _name;
00057     };
00058
00059 #endif // __STDAIR_BOM_AIRLINESTRUCT_HPP

```

### 33.163 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

- `stdair`

*Handle on the StdAir library context.*

## 33.164 AirportPair.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_Inventory.hpp>
00009 #include <stdair/service/Logger.hpp>
00010 #include <stdair/bom/AirportPair.hpp>
00011
00012 namespace stdair {
00013
00014 // //////////////////////////////////////
00015 AirportPair::AirportPair()
00016 : _key (DEFAULT_ORIGIN, DEFAULT_DESTINATION),
00017   _parent (NULL) {
00018     // That constructor is used by the serialisation process
00019 }
00020
00021 // //////////////////////////////////////
00022 AirportPair::AirportPair (const AirportPair& iAirportPair)
00023 : _key (iAirportPair.getKey()), _parent (NULL) {
00024 }
00025
00026 // //////////////////////////////////////
00027 AirportPair::AirportPair (const Key_T& iKey)
00028 : _key (iKey), _parent (NULL) {
00029 }
00030
00031 // //////////////////////////////////////
00032 AirportPair::~AirportPair () {
00033 }
00034
00035 // //////////////////////////////////////
00036 std::string AirportPair::toString() const {
00037     std::ostringstream oStr;
00038     oStr << describeKey();
00039     return oStr.str();
00040 }
00041 }
00042
00043
```

## 33.165 stdair/bom/AirportPair.hpp File Reference

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

## Classes

- class `stdair::AirportPair`

*Class representing the actual attributes for an airport-pair.*

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.166 AirportPair.hpp

```

00001 #ifndef __STDAIR_BOM_AIRPORTPAIR_HPP
00002 #define __STDAIR_BOM_AIRPORTPAIR_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STDAIR
00008 #include <stdair/bom/BomAbstract.hpp>
00009 #include <stdair/bom/AirportPairKey.hpp>
00010 #include <stdair/bom/AirportPairTypes.hpp>
00011
00012 // Forward declaration
00013 namespace stdair {
00014
00015     class AirportPair : public BomAbstract {
00016     public:
00017         template <typename BOM> friend class FacBom;
00018         template <typename BOM> friend class FacCloneBom;
00019         friend class FacBomManager;
00020
00021     public:
00022         // ////////////////////////////////// Type definitions //////////////////////////////////
00023         typedef AirportPairKey Key_T;
00024
00025     public:
00026         // ////////////////////////////////// Display support methods //////////////////////////////////
00027         void toStream (std::ostream& ioOut) const {
00028             ioOut << toString();
00029         }
00030
00031         void fromStream (std::istream& ioIn) {
00032         }
00033
00034         std::string toString() const;
00035
00036         const std::string describeKey() const {
00037             return _key.toString();
00038         }
00039
00040     public:
00041         // ////////////////////////////////// Getters //////////////////////////////////
00042         const Key_T& getKey() const {
00043             return _key;
00044         }
00045
00046         const AirportCode_T& getBoardingPoint() const {
00047             return _key.getBoardingPoint();
00048         }
00049
00050         const AirportCode_T& getOffPoint() const {
00051             return _key.getOffPoint();
00052         }
00053
00054         BomAbstract* const getParent() const {
00055             return _parent;
00056         }
00057
00058         const HolderMap_T& getHolderMap() const {
00059             return _holderMap;
00060         }
00061
00062     protected:
00063         // ////////////////////////////////// Constructors and destructors //////////////////////////////////
00064         AirportPair (const Key_T&);
00065         virtual ~AirportPair();
00066
00067     private:
00068         AirportPair();
00069         AirportPair (const AirportPair&);
00070
00071     protected:
00072         // ////////////////////////////////// Attributes //////////////////////////////////
00073         Key_T _key;
00074
00075         BomAbstract* _parent;
00076
00077         HolderMap_T _holderMap;
00078

```

```

00135
00136     };
00137
00138 }
00139 #endif // __STDAIR_BOM_AIRPORTPAIR_HPP
00140

```

### 33.167 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

- [stdair](#)

*Handle on the StdAir library context.*

### 33.168 AirportPairKey.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <ostream>
00006 #include <sstream>
00007 // STDAIR
00008 #include <stdair/basic/BasConst_BomDisplay.hpp>
00009 #include <stdair/basic/BasConst_Inventory.hpp>
00010 #include <stdair/bom/AirportPairKey.hpp>
00011
00012 namespace stdair {
00013
00014 // //////////////////////////////////////
00015 AirportPairKey::AirportPairKey ()
00016     : _boardingPoint (DEFAULT_ORIGIN),
00017       _offPoint (DEFAULT_DESTINATION) {
00018     assert (false);
00019 }
00020
00021 // //////////////////////////////////////
00022 AirportPairKey::AirportPairKey (const AirportCode_T& iBoardingPoint,
00023                                 const AirportCode_T& iOffPoint)
00024     : _boardingPoint (iBoardingPoint), _offPoint (iOffPoint) {
00025 }
00026
00027 // //////////////////////////////////////
00028 AirportPairKey::AirportPairKey (const AirportPairKey& iKey)
00029     : _boardingPoint (iKey._boardingPoint),
00030       _offPoint (iKey._offPoint) {
00031 }
00032
00033 // //////////////////////////////////////
00034 AirportPairKey::~AirportPairKey () {
00035 }
00036
00037 // //////////////////////////////////////
00038 void AirportPairKey::toStream (std::ostream& ioOut) const {
00039     ioOut << "AirportPairKey: " << toString() << std::endl;
00040 }
00041
00042 // //////////////////////////////////////
00043 void AirportPairKey::fromStream (std::istream& ioIn) {
00044 }
00045
00046 // //////////////////////////////////////
00047 const std::string AirportPairKey::toString() const {
00048     std::ostringstream oStr;
00049     oStr << _boardingPoint << DEFAULT_KEY_SUB_FLD_DELIMITER
00050         << " " << _offPoint;
00051     return oStr.str();

```

```
00052     }
00053
00054 }
```

### 33.169 stdair/bom/AirportPairKey.hpp File Reference

```
#include <stdair/bom/KeyAbstract.hpp>
#include <stdair/stdair_basic_types.hpp>
```

#### Classes

- struct [stdair::AirportPairKey](#)

*Key of airport-pair.*

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.170 AirportPairKey.hpp

```
00001 #ifndef __STDAIR_BOM_AIRPORTPAIRKEY_HPP
00002 #define __STDAIR_BOM_AIRPORTPAIRKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STDAIR
00008 #include <stdair/bom/KeyAbstract.hpp>
00009 #include <stdair/stdair_basic_types.hpp>
00010
00011 namespace stdair {
00012
00016     struct AirportPairKey : public KeyAbstract {
00017
00018     public:
00019         // ////////////////////////////////// Construction //////////////////////////////////
00021         AirportPairKey (const stdair::AirportCode_T&,
00022             const stdair::AirportCode_T&);
00024         AirportPairKey (const AirportPairKey&);
00026         ~AirportPairKey ();
00027     private:
00029         AirportPairKey ();
00030
00031     public:
00032         // ////////////////////////////////// Getters //////////////////////////////////
00036         const stdair::AirportCode_T& getBoardingPoint() const {
00037             return _boardingPoint;
00038         }
00039
00043         const stdair::AirportCode_T& getOffPoint() const {
00044             return _offPoint;
00045         }
00046
00047         // ////////////////////////////////// Display support methods //////////////////////////////////
00053         void toStream (std::ostream& ioOut) const;
00054
00060         void fromStream (std::istream& ioIn);
00061
00067         const std::string toString() const;
00068
00069     private:
00070         // ////////////////////////////////// Attributes //////////////////////////////////
00074         AirportCode_T _boardingPoint;
00075
00079         AirportCode_T _offPoint;
00080     };
00081
00082 }
00083 #endif // __STDAIR_BOM_AIRPORTPAIRKEY_HPP
```

### 33.171 stdair/bom/AirportPairTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

#### Typedefs

- typedef std::list< AirportPair \* > [stdair::AirportPairList\\_T](#)
- typedef std::map< const MapKey\_T, AirportPair \* > [stdair::AirportPairMap\\_T](#)
- typedef std::pair< MapKey\_T, AirportPair \* > [stdair::AirportPairWithKey\\_T](#)
- typedef std::list< AirportPairWithKey\_T > [stdair::AirportPairDetailedList\\_T](#)

### 33.172 AirportPairTypes.hpp

```
00001 ///////////////////////////////////////////////////////////////////
00002 #ifndef __STDAIR_BOM_AIRPORTPAIRTYPES_HPP
00003 #define __STDAIR_BOM_AIRPORTPAIRTYPES_HPP
00004
00005 ///////////////////////////////////////////////////////////////////
00006 // Import section
00007 ///////////////////////////////////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // STDAIR
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016     // Forward declarations.
00017     class AirportPair;
00018
00020     typedef std::list<AirportPair*> AirportPairList_T;
00021
00023     typedef std::map<const MapKey_T, AirportPair*> AirportPairMap_T;
00024
00026     typedef std::pair<MapKey_T, AirportPair*> AirportPairWithKey_T;
00027     typedef std::list<AirportPairWithKey_T> AirportPairDetailedList_T;
00028 }
00029 #endif // __STDAIR_BOM_AIRPORTPAIRTYPES_HPP
00030
```

### 33.173 stdair/bom/BomAbstract.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <map>
#include <typeinfo>
```

#### Classes

- class [stdair::BomAbstract](#)

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



## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## Typedefs

- `typedef std::map< const std::type_info *, BomAbstract * > stdair::HolderMap\_T`

## Functions

- `template<class charT , class traits >  
std::basic_ostream< charT, traits > & operator<< (std::basic_ostream< charT, traits > &ioOut, const stdair::BomAbstract &iBom)`
- `template<class charT , class traits >  
std::basic_istream< charT, traits > & operator>> (std::basic_istream< charT, traits > &ioln, stdair::BomAbstract &iBom)`

### 33.173.1 Function Documentation

**33.173.1.1** `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](#).

**33.173.1.2** `template<class charT , class traits > std::basic_istream<charT, traits>& operator>> ( std::basic_istream< charT, traits > &ioln, stdair::BomAbstract &iBom ) [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\(\)](#).

### 33.174 BomAbstract.hpp

```
00001
00007 #ifndef __STDAIR_BOM_BOMABSTRACT_HPP
00008 #define __STDAIR_BOM_BOMABSTRACT_HPP
00009
00010 // //////////////////////////////////////
00011 // Import section
00012 // //////////////////////////////////////
00013 // STL
00014 #include <iosfwd>
00015 #include <string>
00016 #include <map>
00017 #include <typeinfo>
00018
00019 namespace stdair {
00020
00024 class BomAbstract {
00025 public:
00026     // ////////////////////////////////// Display support methods //////////////////////////////////
00032     virtual void toStream (std::ostream& ioOut) const = 0;
00033
00039     virtual void fromStream (std::istream& ioIn) = 0;
00040
00046     virtual std::string toString() const = 0;
00047
00048
00049 protected:
```

```

00053     BomAbstract() {}
00054     BomAbstract(const BomAbstract&) {}
00055 public:
00059     virtual ~BomAbstract() {}
00060 };
00061
00062 /* Define the map of object holder type. */
00063 typedef std::map<const std::type_info*, BomAbstract*> HolderMap_T;
00064 }
00065
00071 template <class charT, class traits>
00072 inline
00073 std::basic_ostream<charT, traits>&
00074 operator<< (std::basic_ostream<charT, traits>& ioOut,
00075           const stdair::BomAbstract& iBom) {
00081     std::basic_ostringstream<charT,traits> ostr;
00082     ostr.copyfmt (ioOut);
00083     ostr.width (0);
00084
00085     // Fill string stream
00086     iBom.toStream (ostr);
00087
00088     // Print string stream
00089     ioOut << ostr.str();
00090
00091     return ioOut;
00092 }
00093
00099 template <class charT, class traits>
00100 inline
00101 std::basic_istream<charT, traits>&
00102 operator>> (std::basic_istream<charT, traits>& ioIn,
00103           stdair::BomAbstract& ioBom) {
00104     // Fill Bom object with input stream
00105     ioBom.fromStream (ioIn);
00106     return ioIn;
00107 }
00108
00109 #endif // __STDAIR_BOM_BOMABSTRACT_HPP

```

### 33.175 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

- [stdair](#)

*Handle on the StdAir library context.*

### 33.176 BomArchive.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/tmpdir.hpp>
00009 #include <boost/archive/text_iarchive.hpp>
00010 #include <boost/archive/text_oarchive.hpp>
00011 #include <boost/serialization/base_object.hpp>
00012 #include <boost/serialization/utility.hpp>
00013 #include <boost/serialization/list.hpp>
00014 // #include <boost/serialization/assume_abstract.hpp>
00015 // StdAir
00016 #include <stdair/bom/BomRoot.hpp>
00017 #include <stdair/bom/Inventory.hpp>
00018 #include <stdair/bom/FlightDate.hpp>
00019 #include <stdair/bom/LegDate.hpp>
00020 #include <stdair/bom/SegmentDate.hpp>
00021 #include <stdair/bom/LegCabin.hpp>
00022 #include <stdair/bom/SegmentCabin.hpp>
00023 #include <stdair/bom/FareFamily.hpp>
00024 #include <stdair/bom/BookingClass.hpp>
00025 #include <stdair/bom/BookingRequestStruct.hpp>
00026 #include <stdair/bom/BomManager.hpp>
00027 #include <stdair/bom/BomArchive.hpp>
00028
00029 namespace stdair {
00030
00031 // //////////////////////////////////////
00032 void BomArchive::archive (const BomRoot& iBomRoot) {
00033 }
00034
00035 // //////////////////////////////////////
00036 std::string BomArchive::archive (const Inventory& iInventory) {
00037     std::ostringstream oStr;
00038     boost::archive::text_oarchive oa (oStr);
00039     oa << iInventory;
00040     return oStr.str();
00041 }
00042
00043 // //////////////////////////////////////
00044 void BomArchive::restore (const std::string& iArchive,
00045                          Inventory& ioInventory) {
00046     std::istringstream iStr;
00047     boost::archive::text_iarchive ia (iStr);
00048     ia >> ioInventory;
00049 }
00050
00051 // //////////////////////////////////////
00052 void BomArchive::archive (const FlightDate& iFlightDate) {
00053 }
00054
00055 }

```

### 33.177 stdair/bom/BomArchive.hpp File Reference

```
#include <iosfwd>
```

#### Classes

- class [stdair::BomArchive](#)

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

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.178 BomArchive.hpp

```

00001 #ifndef __STDAIR_BOM_BOMARCHIVE_HPP
00002 #define __STDAIR_BOM_BOMARCHIVE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009
00010 namespace stdair {
00011
00012     class BomRoot;
00013     class Inventory;
00014     class FlightDate;
00015     class LegDate;
00016     class SegmentDate;
00017     class LegCabin;
00018     class SegmentCabin;
00019     class FareFamily;
00020     class BookingClass;
00021     struct BookingRequestStruct;
00022
00023     class BomArchive {
00024     public:
00025         static void archive (const BomRoot&);
00026
00027         static std::string archive (const Inventory&);
00028
00029         static void restore (const std::string& iArchive, Inventory&);
00030
00031         static void archive (const FlightDate&);
00032     };
00033 }
00034
00035 #endif // __STDAIR_BOM_BOMARCHIVE_HPP

```

## 33.179 stdair/bom/BomDisplay.cpp File Reference

## 33.180 BomDisplay.cpp

```

00001
00002 // //////////////////////////////////////
00003 // Import section
00004 // //////////////////////////////////////
00005 // STL
00006 #include <cassert>
00007 #include <ostream>
00008 // StdAir
00009 #include <stdair/basic/BasConst_BomDisplay.hpp>
00010 #include <stdair/bom/BomManager.hpp>
00011 #include <stdair/bom/BomRoot.hpp>
00012 #include <stdair/bom/Inventory.hpp>
00013 #include <stdair/bom/FlightDate.hpp>
00014 #include <stdair/bom/LegDate.hpp>
00015 #include <stdair/bom/SegmentDate.hpp>
00016 #include <stdair/bom/LegCabin.hpp>
00017 #include <stdair/bom/SegmentCabin.hpp>
00018 #include <stdair/bom/FareFamily.hpp>
00019 #include <stdair/bom/BookingClass.hpp>
00020 #include <stdair/bom/AirportPair.hpp>
00021 #include <stdair/bom/PosChannel.hpp>
00022 #include <stdair/bom/DatePeriod.hpp>
00023 #include <stdair/bom/TimePeriod.hpp>
00024 #include <stdair/bom/FareFeatures.hpp>
00025 #include <stdair/bom/YieldFeatures.hpp>
00026 #include <stdair/bom/AirlineClassList.hpp>
00027 #include <stdair/bom/Bucket.hpp>
00028 #include <stdair/bom/TravelSolutionTypes.hpp>
00029 #include <stdair/bom/TravelSolutionStruct.hpp>
00030 #include <stdair/bom/BomDisplay.hpp>
00031 #include <stdair/bom/OnDDate.hpp>
00032
00033 namespace stdair {
00034
00035     struct FlagSaver {
00036     public:
00037         FlagSaver (std::ostream& oStream)
00038             : _oStream (oStream), _streamFlags (oStream.flags()) {
00039         }
00040     };
00041 }

```

```

00051     ~FlagSaver() {
00052         // Reset formatting flags of the given output stream
00053         _oStream.flags(_streamFlags);
00054     }
00055
00056 private:
00057     std::ostream& _oStream;
00060     std::ios::fmtflags _streamFlags;
00061 };
00062
00063 // //////////////////////////////////////
00064 void BomDisplay::list (std::ostream& oStream, const BomRoot& iBomRoot,
00065                      const AirlineCode_T& iAirlineCode,
00066                      const FlightNumber_T& iFlightNumber) {
00067     // Save the formatting flags for the given STL output stream
00068     FlagSaver flagSaver (oStream);
00069
00070     // Check whether there are Inventory objects
00071     if (BomManager::hasList<Inventory> (iBomRoot) == false) {
00072         return;
00073     }
00074
00075     // Browse the inventories
00076     unsigned short invIdx = 1;
00077     const InventoryList_T& lInventoryList =
00078         BomManager::getList<Inventory> (iBomRoot);
00079     for (InventoryList_T::const_iterator itInv = lInventoryList.begin();
00080          itInv != lInventoryList.end(); ++itInv, ++invIdx) {
00081         const Inventory* lInv_ptr = *itInv;
00082         assert (lInv_ptr != NULL);
00083
00084         // Retrieve the inventory key (airline code)
00085         const AirlineCode_T& lAirlineCode = lInv_ptr->getAirlineCode();
00086
00087         // Display only the requested inventories
00088         if (iAirlineCode == "all" || iAirlineCode == lAirlineCode) {
00089             // Get the list of flight-dates for that inventory
00090             list (oStream, *lInv_ptr, invIdx, iFlightNumber);
00091         }
00092     }
00093 }
00094
00095 // //////////////////////////////////////
00096 void BomDisplay::list (std::ostream& oStream, const Inventory& iInventory,
00097                      const unsigned short iInventoryIndex,
00098                      const FlightNumber_T& iFlightNumber) {
00099     // Save the formatting flags for the given STL output stream
00100     FlagSaver flagSaver (oStream);
00101
00102     // Check whether there are FlightDate objects
00103     if (BomManager::hasMap<FlightDate> (iInventory) == false) {
00104         return;
00105     }
00106
00107     //
00108     const AirlineCode_T& lAirlineCode = iInventory.getAirlineCode();
00109     oStream << iInventoryIndex << ". " << lAirlineCode << std::endl;
00110
00111     // Browse the flight-dates
00112     unsigned short lCurrentFlightNumber = 0;
00113     unsigned short flightNumberIdx = 0;
00114     unsigned short departureDateIdx = 1;
00115     const FlightDateMap_T& lFlightDateList =
00116         BomManager::getMap<FlightDate> (iInventory);
00117     for (FlightDateMap_T::const_iterator itFD = lFlightDateList.begin();
00118          itFD != lFlightDateList.end(); ++itFD, ++departureDateIdx) {
00119         const FlightDate* lFD_ptr = itFD->second;
00120         assert (lFD_ptr != NULL);
00121
00122         // Retrieve the key of the flight-date
00123         const FlightNumber_T& lFlightNumber = lFD_ptr->getFlightNumber();
00124         const Date_T& lFlightDateDate = lFD_ptr->getDepartureDate();
00125
00126         // Display only the requested flight number
00127         if (iFlightNumber == 0 || iFlightNumber == lFlightNumber) {
00128             //
00129             if (lCurrentFlightNumber != lFlightNumber) {
00130                 lCurrentFlightNumber = lFlightNumber;
00131                 ++flightNumberIdx; departureDateIdx = 1;
00132                 oStream << " " << iInventoryIndex << ". " << flightNumberIdx << ". "
00133                     << lAirlineCode << lFlightNumber << std::endl;
00134             }
00135
00136             oStream << " " << iInventoryIndex << ". " << flightNumberIdx
00137                 << ". " << departureDateIdx << ". "
00138                 << lAirlineCode << lFlightNumber << " / " << lFlightDateDate
00139                 << std::endl;
00140         }
00141     }
00142 }
00143
00144 oStream << " " << iInventoryIndex << ". " << flightNumberIdx
00145     << ". " << departureDateIdx << ". "
00146     << lAirlineCode << lFlightNumber << " / " << lFlightDateDate
00147     << std::endl;

```

```

00148     }
00149     }
00150 }
00151
00152 ///////////////////////////////////////////////////////////////////
00153 void BomDisplay::listAirportPairDateRange (std::ostream& oStream,
00154                                           const BomRoot& iBomRoot) {
00155     // Save the formatting flags for the given STL output stream
00156     FlagSaver flagSaver (oStream);
00157
00158     // Check whether there are AirportPair objects
00159     if (BomManager::hasList<AirportPair> (iBomRoot) == false) {
00160         return;
00161     }
00162
00163     const AirportPairList_T& lAirportPairList =
00164         BomManager::getList<AirportPair> (iBomRoot);
00165     for (AirportPairList_T::const_iterator itAir = lAirportPairList.begin();
00166          itAir != lAirportPairList.end(); ++itAir) {
00167         const AirportPair* lAir_ptr = *itAir;
00168         assert (lAir_ptr != NULL);
00169
00170         // Check whether there are date-period objects
00171         assert (BomManager::hasList<DatePeriod> (*lAir_ptr) == true);
00172
00173         // Browse the date-period objects
00174         const DatePeriodList_T& lDatePeriodList =
00175             BomManager::getList<DatePeriod> (*lAir_ptr);
00176
00177         for (DatePeriodList_T::const_iterator itDP = lDatePeriodList.begin();
00178              itDP != lDatePeriodList.end(); ++itDP) {
00179             const DatePeriod* lDP_ptr = *itDP;
00180             assert (lDP_ptr != NULL);
00181
00182             // Display the date-period object
00183             oStream << lAir_ptr->describeKey()
00184                 << " / " << lDP_ptr->describeKey() << std::endl;
00185         }
00186     }
00187 }
00188
00189 ///////////////////////////////////////////////////////////////////
00190 void BomDisplay::csvDisplay (std::ostream& oStream,
00191                             const BomRoot& iBomRoot) {
00192     // Save the formatting flags for the given STL output stream
00193     FlagSaver flagSaver (oStream);
00194
00195     oStream << std::endl;
00196     oStream << "===== "
00197         << std::endl;
00198     oStream << "BomRoot: " << iBomRoot.describeKey() << std::endl;
00199     oStream << "===== "
00200         << std::endl;
00201
00202     // Check whether there are Inventory objects
00203     if (BomManager::hasList<Inventory> (iBomRoot) == false) {
00204         return;
00205     }
00206
00207     // Browse the inventories
00208     const InventoryList_T& lInventoryList =
00209         BomManager::getList<Inventory> (iBomRoot);
00210     for (InventoryList_T::const_iterator itInv = lInventoryList.begin();
00211          itInv != lInventoryList.end(); ++itInv) {
00212         const Inventory* lInv_ptr = *itInv;
00213         assert (lInv_ptr != NULL);
00214
00215         // Display the inventory
00216         csvDisplay (oStream, *lInv_ptr);
00217     }
00218 }
00219
00220 ///////////////////////////////////////////////////////////////////
00221 void BomDisplay::csvDisplay (std::ostream& oStream,
00222                             const Inventory& iInventory) {
00223     // Save the formatting flags for the given STL output stream
00224     FlagSaver flagSaver (oStream);
00225
00226     oStream << "+++++" << std::endl;
00227     oStream << "Inventory: " << iInventory.describeKey() << std::endl;
00228     oStream << "+++++" << std::endl;
00229
00230     // Check whether there are FlightDate objects
00231     if (BomManager::hasList<FlightDate> (iInventory) == false) {
00232         return;
00233     }
00234 }

```

```

00241
00242 // Browse the flight-dates
00243 const FlightDateList_T& lFlightDateList =
00244     BomManager::getList<FlightDate> (iInventory);
00245 for (FlightDateList_T::const_iterator itFD = lFlightDateList.begin();
00246      itFD != lFlightDateList.end(); ++itFD) {
00247     const FlightDate* lFD_ptr = *itFD;
00248     assert (lFD_ptr != NULL);
00249
00250     // Display the flight-date
00251     csvDisplay (oStream, *lFD_ptr);
00252 }
00253
00254 // Check if the inventory contains a list of partners
00255
00256 if (BomManager::hasList<Inventory> (iInventory)){
00257
00258     // Browse the partner's inventories
00259     const InventoryList_T& lPartnerInventoryList =
00260         BomManager::getList<Inventory> (iInventory);
00261
00262     for (InventoryList_T::const_iterator itInv = lPartnerInventoryList.begin();
00263          itInv != lPartnerInventoryList.end(); ++itInv) {
00264
00265         oStream << "-----" << std::endl;
00266         oStream << "Partner inventory:" << std::endl;
00267         oStream << "-----" << std::endl;
00268         const Inventory* lInv_ptr = *itInv;
00269         assert (lInv_ptr != NULL);
00270
00271         // Display the inventory
00272         csvDisplay (oStream, *lInv_ptr);
00273     }
00274     oStream << "*****" << std::endl;
00275     oStream << std::endl;
00276 }
00277
00278 // Check if the inventory contains a list of O&D dates
00279
00280 if (BomManager::hasList<OnDDate> (iInventory)){
00281
00282     //Browse the O&Ds
00283     const OnDDateList_T& lOnDDateList =
00284         BomManager::getList<OnDDate> (iInventory);
00285
00286     for (OnDDateList_T::const_iterator itOnD = lOnDDateList.begin();
00287          itOnD != lOnDDateList.end(); ++itOnD) {
00288         oStream << "*****" << std::endl;
00289         oStream << "O&D-Date:" << std::endl;
00290         oStream << "-----" << std::endl;
00291         oStream << "Airline, Date, Origin-Destination, Segments, " << std::endl;
00292
00293         const OnDDate* lOnDDate_ptr = *itOnD;
00294         assert (lOnDDate_ptr != NULL);
00295
00296         // Display the O&D date
00297         csvDisplay (oStream, *lOnDDate_ptr);
00298     }
00299     oStream << "*****" << std::endl;
00300 }
00301 }
00302
00303 // //////////////////////////////////////
00304 void BomDisplay::csvDisplay (std::ostream& oStream,
00305                             const OnDDate& iOnDDate) {
00306     // Save the formatting flags for the given STL output stream
00307     FlagSaver flagSaver (oStream);
00308
00309     const AirlineCode_T& lAirlineCode = iOnDDate.getAirlineCode();
00310     const Date_T& lDate = iOnDDate.getDate();
00311     const AirportCode_T& lOrigin = iOnDDate.getOrigin();
00312     const AirportCode_T& lDestination = iOnDDate.getDestination();
00313
00314     oStream << lAirlineCode << ", " << lDate << ", " << lOrigin << "-"
00315         << lDestination << ", " << iOnDDate.describeKey() << ", "
00316         << std::endl;
00317
00318     const StringDemandStructMap_T& lDemandInfoMap =
00319         iOnDDate.getDemandInfoMap();
00320
00321     // Check if the map contains information.
00322     const bool isInfoMapEmpty = lDemandInfoMap.empty();
00323     if (isInfoMapEmpty) {
00324         return;
00325     }
00326     assert (lDemandInfoMap.empty() == false);
00327
00328 }
00329
00330

```

```

00331     ostream << "-----" << std::endl;
00332     ostream << "Cabin-Class path, Demand mean, Demand std dev, Yield, "
00333             << std::endl;
00334
00335     for (StringDemandStructMap_T::const_iterator itDI = lDemandInfoMap.begin();
00336          itDI != lDemandInfoMap.end(); ++itDI) {
00337         const std::string& lCabinClassPath = itDI->first;
00338         const YieldDemandPair_T lYieldDemandPair =
00339             itDI->second;
00340         const Yield_T lYield = lYieldDemandPair.first;
00341         const MeanStdDevPair_T lMeanStdDevPair =
00342             lYieldDemandPair.second;
00343         const MeanValue_T lDemandMean = lMeanStdDevPair.first;
00344         const StdDevValue_T lDemandStdDev = lMeanStdDevPair.second;
00345
00346         ostream << lCabinClassPath << ", "
00347                 << lDemandMean << ", "
00348                 << lDemandStdDev << ", "
00349                 << lYield << ", "
00350                 << std::endl;
00351     }
00352 }
00353
00354 }
00355
00356 // //////////////////////////////////////
00357 void BomDisplay::csvDisplay (std::ostream& ostream,
00358                             const FlightDate& iFlightDate) {
00359     // Save the formatting flags for the given STL output stream
00360     FlagSaver flagSaver (ostream);
00361
00362     const AirlineCode_T& lAirlineCode = iFlightDate.getAirlineCode();
00363     ostream << "*****" << std::endl;
00364     ostream << "FlightDate: " << lAirlineCode << iFlightDate.describeKey()
00365             << std::endl;
00366     ostream << "*****" << std::endl;
00367
00368     //
00369     csvSegmentDateDisplay (ostream, iFlightDate);
00370     //
00371     csvLegDateDisplay (ostream, iFlightDate);
00372     //
00373     csvLegCabinDisplay (ostream, iFlightDate);
00374     //
00375     csvBucketDisplay (ostream, iFlightDate);
00376     //
00377     csvFareFamilyDisplay (ostream, iFlightDate);
00378     //
00379     csvBookingClassDisplay (ostream, iFlightDate);
00380 }
00381
00382 // //////////////////////////////////////
00383 void BomDisplay::csvLegDateDisplay (std::ostream& ostream,
00384                                     const FlightDate& iFlightDate) {
00385     // Save the formatting flags for the given STL output stream
00386     FlagSaver flagSaver (ostream);
00387
00388     ostream << "*****" << std::endl;
00389     ostream << "Leg-Dates:" << std::endl
00390             << "-----" << std::endl;
00391     ostream << "Flight, Leg, BoardDate, BoardTime, "
00392             << "OffDate, OffTime, Date Offset, Time Offset, Elapsed, "
00393             << "Distance, Capacity, " << std::endl;
00394
00395     // Retrieve the key of the flight-date
00396     const AirlineCode_T& lAirlineCode = iFlightDate.getAirlineCode();
00397     const FlightNumber_T& lFlightNumber = iFlightDate.getFlightNumber();
00398     const Date_T& lFlightDateDate = iFlightDate.getDepartureDate();
00399
00400     // Check whether there are LegDate objects
00401     if (BomManager::hasList<LegDate> (iFlightDate) == false) {
00402         return;
00403     }
00404
00405     // Browse the leg-dates
00406     const LegDateList_T& lLegDateList =
00407         BomManager::getList<LegDate> (iFlightDate);
00408     for (LegDateList_T::const_iterator itLD = lLegDateList.begin();
00409          itLD != lLegDateList.end(); ++itLD) {
00410         const LegDate* lLD_ptr = *itLD;
00411         assert (lLD_ptr != NULL);
00412
00413         ostream << lAirlineCode << lFlightNumber << " "

```



```

00426         << lFlightDateDate << ", ";
00427
00428     oStream << lLD_ptr->getBoardingPoint() << "- "
00429         << lLD_ptr->getOffPoint() << ", "
00430         << lLD_ptr->getBoardingDate() << ", "
00431         << lLD_ptr->getBoardingTime() << ", "
00432         << lLD_ptr->getOffDate() << ", "
00433         << lLD_ptr->getOffTime() << ", "
00434         << lLD_ptr->getElapsedTime() << ", "
00435         << lLD_ptr->getDateOffset().days() << ", "
00436         << lLD_ptr->getTimeOffset() << ", "
00437         << lLD_ptr->getDistance() << ", "
00438         << lLD_ptr->getCapacity() << ", " << std::endl;
00439     }
00440     oStream << "*****" << std::endl;
00441 }
00442
00443 // //////////////////////////////////////
00444 void BomDisplay::csvSegmentDateDisplay (std::ostream& oStream,
00445                                         const FlightDate& iFlightDate) {
00446     // Save the formatting flags for the given STL output stream
00447     FlagSaver flagSaver (oStream);
00448
00449     oStream << "*****" << std::endl;
00450     oStream << "SegmentDates:" << std::endl
00451         << "-----" << std::endl;
00452     oStream << "Flight, Segment, Date"
00453         << std::endl;
00454
00455     // Retrieve the key of the flight-date
00456     const AirlineCode_T& lAirlineCode = iFlightDate.getAirlineCode();
00457     const FlightNumber_T& lFlightNumber = iFlightDate.getFlightNumber();
00458     const Date_T& lFlightDateDate = iFlightDate.getDepartureDate();
00459
00460     // Check whether there are SegmentDate objects
00461     if (BomManager::hasList<SegmentDate> (iFlightDate) == false) {
00462         return;
00463     }
00464
00465     // Browse the segment-dates
00466     const SegmentDateList_T& lSegmentDateList =
00467         BomManager::getList<SegmentDate> (iFlightDate);
00468     for (SegmentDateList_T::const_iterator itSD = lSegmentDateList.begin();
00469          itSD != lSegmentDateList.end(); ++itSD) {
00470         const SegmentDate* lSD_ptr = *itSD;
00471         assert (lSD_ptr != NULL);
00472
00473         // Retrieve the key of the segment-date, as well as its dates
00474         const Date_T& lSegmentDateDate = lSD_ptr->getBoardingDate();
00475         const AirportCode_T& lBoardPoint = lSD_ptr->getBoardingPoint();
00476         const AirportCode_T& lOffPoint = lSD_ptr->getOffPoint();
00477         oStream << lAirlineCode << lFlightNumber << " " << lFlightDateDate << ", "
00478             << lBoardPoint << "- " << lOffPoint << ", " << lSegmentDateDate << std::endl;
00479
00480         // Check if the current segment has corresponding marketing segments.
00481         const bool hasMarketingSDList = BomManager::hasList<SegmentDate> (*lSD_ptr);
00482         if (hasMarketingSDList == true) {
00483             //
00484             const SegmentDateList_T& lMarketingSDList = BomManager::getList<SegmentDate>
00485                 (*lSD_ptr);
00486
00487             oStream << " *** Marketed by ";
00488             for (SegmentDateList_T::const_iterator itMarketingSD = lMarketingSDList.begin();
00489                  itMarketingSD != lMarketingSDList.end(); ++itMarketingSD) {
00490                 SegmentDate* lMarketingSD_ptr = *itMarketingSD;
00491                 FlightDate* lMarketingFD_ptr = BomManager::getParentPtr<FlightDate> (*lMarketingSD_ptr);
00492                 Inventory* lMarketingInv_ptr = BomManager::getParentPtr<Inventory> (*lMarketingFD_ptr);
00493                 oStream << lMarketingInv_ptr->toString() << lMarketingFD_ptr->toString() << " * ";
00494             }
00495
00496             // Check if the current segment is operated by another segment date.
00497             const SegmentDate* lOperatingSD_ptr = lSD_ptr->getOperatingSegmentDate ();
00498             if (lOperatingSD_ptr != NULL) {
00499                 const FlightDate* lOperatingFD_ptr = BomManager::getParentPtr<FlightDate> (*lOperatingSD_ptr);
00500                 const Inventory* lOperatingInv_ptr = BomManager::getParentPtr<Inventory> (*lOperatingFD_ptr);
00501                 oStream << " *** Operated by " << lOperatingInv_ptr->toString()
00502                     << lOperatingFD_ptr->toString() << std::endl;
00503             }
00504             oStream << std::endl;
00505         }
00506     }
00507
00508 // //////////////////////////////////////
00509 void BomDisplay::csvLegCabinDisplay (std::ostream& oStream,

```

```

00515                                     const FlightDate& iFlightDate) {
00516 // Save the formatting flags for the given STL output stream
00517 FlagSaver flagSaver (oStream);
00518
00522 oStream << "*****" << std::endl;
00523 oStream << "LegCabins:" << std::endl
00524 << "-----" << std::endl;
00525 oStream << "Flight, Leg, Cabin, "
00526 << "OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, "
00527 << "CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice, "
00528 << std::endl;
00529
00530 // Retrieve the key of the flight-date
00531 const AirlineCode_T& lAirlineCode = iFlightDate.getAirlineCode();
00532 const FlightNumber_T& lFlightNumber = iFlightDate.getFlightNumber();
00533 const Date_T& lFlightDateDate = iFlightDate.getDepartureDate();
00534
00535 // Check whether there are LegDate objects
00536 if (BomManager::hasList<LegDate> (iFlightDate) == false) {
00537     return;
00538 }
00539
00540 // Browse the leg-dates
00541 const LegDateList_T& lLegDateList =
00542     BomManager::getList<LegDate> (iFlightDate);
00543 for (LegDateList_T::const_iterator itLD = lLegDateList.begin();
00544      itLD != lLegDateList.end(); ++itLD) {
00545     const LegDate* lLD_ptr = *itLD;
00546     assert (lLD_ptr != NULL);
00547
00548 // Retrieve the key of the leg-date, as well as its off point
00549 const Date_T& lLegDateDate = lLD_ptr->getBoardingDate();
00550 const AirportCode_T& lBoardPoint = lLD_ptr->getBoardingPoint();
00551 const AirportCode_T& lOffPoint = lLD_ptr->getOffPoint();
00552
00553 // Browse the leg-cabins
00554 const LegCabinList_T& lLegCabinList =
00555     BomManager::getList<LegCabin> (*lLD_ptr);
00556 for (LegCabinList_T::const_iterator itLC = lLegCabinList.begin();
00557      itLC != lLegCabinList.end(); ++itLC) {
00558     const LegCabin* lLC_ptr = *itLC;
00559     assert (lLC_ptr != NULL);
00560
00561     oStream << lAirlineCode << lFlightNumber << " "
00562 << lFlightDateDate << ", ";
00563
00564     oStream << lBoardPoint << "-" << lOffPoint
00565 << " " << lLegDateDate << ", ";
00566
00567     oStream << lLC_ptr->getCabinCode() << ", ";
00568
00569     oStream << lLC_ptr->getOfferedCapacity() << ", "
00570 << lLC_ptr->getPhysicalCapacity() << ", "
00571 << lLC_ptr->getRegradeAdjustment() << ", "
00572 << lLC_ptr->getAuthorizationLevel() << ", "
00573 << lLC_ptr->getUPR() << ", "
00574 << lLC_ptr->getSoldSeat() << ", "
00575 << lLC_ptr->getStaffNbOfSeats() << ", "
00576 << lLC_ptr->getWLNbOfSeats() << ", "
00577 << lLC_ptr->getGroupNbOfSeats() << ", "
00578 << lLC_ptr->getCommittedSpace() << ", "
00579 << lLC_ptr->getAvailabilityPool() << ", "
00580 << lLC_ptr->getAvailability() << ", "
00581 << lLC_ptr->getNetAvailability() << ", "
00582 << lLC_ptr->getGrossAvailability() << ", "
00583 << lLC_ptr->getAvgCancellationPercentage() << ", "
00584 << lLC_ptr->getETB() << ", "
00585 << lLC_ptr->getCurrentBidPrice() << ", "
00586 << std::endl;
00587 }
00588 }
00589 oStream << "*****" << std::endl;
00590 }
00591
00592 // //////////////////////////////////////
00593 void BomDisplay::csvSegmentCabinDisplay (std::ostream& oStream,
00594                                     const FlightDate& iFlightDate) {
00595 // Save the formatting flags for the given STL output stream
00596 FlagSaver flagSaver (oStream);
00597
00601 }
00602
00603 // //////////////////////////////////////
00604 void BomDisplay::csvFareFamilyDisplay (std::ostream& oStream,
00605                                     const FlightDate& iFlightDate) {
00606 // Save the formatting flags for the given STL output stream
00607 FlagSaver flagSaver (oStream);

```

```

00608
00613 oStream << "*****" << std::endl;
00614 oStream << "SegmentCabins:" << std::endl;
00615 << "-----" << std::endl;
00616 oStream << "Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, "
00617 << "CommSpace, AvPool, BP, " << std::endl;
00618
00619 // Retrieve the key of the flight-date
00620 const AirlineCode_T& lAirlineCode = iFlightDate.getAirlineCode();
00621 const FlightNumber_T& lFlightNumber = iFlightDate.getFlightNumber();
00622 const Date_T& lFlightDateDate = iFlightDate.getDepartureDate();
00623
00624 // Check whether there are SegmentDate objects
00625 if (BomManager::hasList<SegmentDate> (iFlightDate) == false) {
00626     return;
00627 }
00628
00629 // Browse the segment-dates
00630 const SegmentDateList_T& lSegmentDateList =
00631 BomManager::getList<SegmentDate> (iFlightDate);
00632 for (SegmentDateList_T::const_iterator itSD = lSegmentDateList.begin();
00633      itSD != lSegmentDateList.end(); ++itSD) {
00634     const SegmentDate* lSD_ptr = *itSD;
00635     assert (lSD_ptr != NULL);
00636
00637     // Retrieve the key of the segment-date, as well as its dates
00638     const Date_T& lSegmentDateDate = lSD_ptr->getBoardingDate();
00639     const AirportCode_T& lBoardPoint = lSD_ptr->getBoardingPoint();
00640     const AirportCode_T& lOffPoint = lSD_ptr->getOffPoint();
00641
00642     // Browse the segment-cabins
00643     const SegmentCabinList_T& lSegmentCabinList =
00644 BomManager::getList<SegmentCabin> (*lSD_ptr);
00645 for (SegmentCabinList_T::const_iterator itSC = lSegmentCabinList.begin();
00646      itSC != lSegmentCabinList.end(); ++itSC) {
00647     const SegmentCabin* lSC_ptr = *itSC;
00648     assert (lSC_ptr != NULL);
00649
00650     // Retrieve the key of the segment-cabin
00651     const CabinCode_T& lCabinCode = lSC_ptr->getCabinCode();
00652
00653     // Check whether there are fare family objects
00654     if (BomManager::hasList<FareFamily> (*lSC_ptr) == false) {
00655         continue;
00656     }
00657
00658     // Browse the fare families
00659     const FareFamilyList_T& lFareFamilyList =
00660 BomManager::getList<FareFamily> (*lSC_ptr);
00661 for (FareFamilyList_T::const_iterator itFF = lFareFamilyList.begin();
00662      itFF != lFareFamilyList.end(); ++itFF) {
00663     const FareFamily* lFF_ptr = *itFF;
00664     assert (lFF_ptr != NULL);
00665
00666     oStream << lAirlineCode << lFlightNumber << " "
00667 << lFlightDateDate << ", ";
00668
00669     oStream << lBoardPoint << "-" << lOffPoint << " "
00670 << lSegmentDateDate << ", ";
00671
00672     oStream << lCabinCode << ", " << lFF_ptr->getFamilyCode() << ", ";
00673
00674     oStream << lSC_ptr->getBookingCounter() << ", "
00675 << lSC_ptr->getMIN() << ", "
00676 << lSC_ptr->getUPR() << ", "
00677 << lSC_ptr->getCommittedSpace() << ", "
00678 << lSC_ptr->getAvailabilityPool() << ", "
00679 << lSC_ptr->getCurrentBidPrice() << ", "
00680 << std::endl;
00681     }
00682 }
00683 }
00684 oStream << "*****" << std::endl;
00685 }
00686
00687 ///////////////////////////////////////////////////////////////////
00688 void BomDisplay::csvBucketDisplay (std::ostream& oStream,
00689                                   const FlightDate& iFlightDate) {
00690     // Save the formatting flags for the given STL output stream
00691     FlagSaver flagSaver (oStream);
00692
00693     oStream << "*****" << std::endl;
00694     oStream << "Buckets:" << std::endl;
00695     << "-----" << std::endl;
00696     oStream << "Flight, Leg, Cabin, Yield, AU/SI, SS, AV, "
00697 << std::endl;
00700
00701

```

```

00702 // Retrieve the key of the flight-date
00703 const AirlineCode_T& lAirlineCode = iFlightDate.getAirlineCode();
00704 const FlightNumber_T& lFlightNumber = iFlightDate.getFlightNumber();
00705 const Date_T& lFlightDateDate = iFlightDate.getDepartureDate();
00706
00707 // Check whether there are LegDate objects
00708 if (BomManager::hasList<LegDate> (iFlightDate) == false) {
00709     return;
00710 }
00711
00712 // Browse the leg-dates
00713 const LegDateList_T& lLegDateList =
00714     BomManager::getList<LegDate> (iFlightDate);
00715 for (LegDateList_T::const_iterator itLD = lLegDateList.begin();
00716      itLD != lLegDateList.end(); ++itLD) {
00717     const LegDate* lLD_ptr = *itLD;
00718     assert (lLD_ptr != NULL);
00719
00720 // Retrieve the key of the leg-date, as well as its off point
00721 const Date_T& lLegDateDate = lLD_ptr->getBoardingDate();
00722 const AirportCode_T& lBoardPoint = lLD_ptr->getBoardingPoint();
00723 const AirportCode_T& lOffPoint = lLD_ptr->getOffPoint();
00724
00725 // Browse the leg-cabins
00726 const LegCabinList_T& lLegCabinList =
00727     BomManager::getList<LegCabin> (*lLD_ptr);
00728 for (LegCabinList_T::const_iterator itLC = lLegCabinList.begin();
00729      itLC != lLegCabinList.end(); ++itLC) {
00730     const LegCabin* lLC_ptr = *itLC;
00731     assert (lLC_ptr != NULL);
00732
00733 // Check whether there are bucket objects
00734 if (BomManager::hasList<Bucket> (*lLC_ptr) == false) {
00735     continue;
00736 }
00737
00738 // Retrieve the key of the leg-cabin
00739 const CabinCode_T& lCabinCode = lLC_ptr->getCabinCode();
00740
00741 // Browse the buckets
00742 const BucketList_T& lBucketList = BomManager::getList<Bucket> (*lLC_ptr);
00743 for (BucketList_T::const_iterator itBuck = lBucketList.begin();
00744      itBuck != lBucketList.end(); ++itBuck) {
00745     const Bucket* lBucket_ptr = *itBuck;
00746     assert (lBucket_ptr != NULL);
00747
00748     ostream << lAirlineCode << lFlightNumber << " "
00749         << lFlightDateDate << ", ";
00750
00751     ostream << lBoardPoint << "-" << lOffPoint << " "
00752         << lLegDateDate << ", " << lCabinCode << ", ";
00753
00754     ostream << lBucket_ptr->getYieldRangeUpperValue() << ", "
00755         << lBucket_ptr->getSeatIndex() << ", "
00756         << lBucket_ptr->getSoldSeats() << ", "
00757         << lBucket_ptr->getAvailability() << ", ";
00758     ostream << std::endl;
00759 }
00760 }
00761 }
00762 ostream << "*****" << std::endl;
00763 }
00764
00765 ///////////////////////////////////////////////////////////////////
00766 void BomDisplay::csvBookingClassDisplay (std::ostream& ostream,
00767     const BookingClass& iBookingClass,
00768     const std::string& iLeadingString) {
00769 // Save the formatting flags for the given STL output stream
00770 FlagSaver flagSaver (ostream);
00771
00772 ostream << iLeadingString << iBookingClass.getClassCode();
00773
00774 if (iBookingClass.getSubclassCode() == 0) {
00775     ostream << ", ";
00776 } else {
00777     ostream << iBookingClass.getSubclassCode() << ", ";
00778 }
00779
00780 ostream << iBookingClass.getAuthorizationLevel() << " ("
00781     << iBookingClass.getProtection() << ", "
00782     << iBookingClass.getNegotiatedSpace() << ", "
00783     << iBookingClass.getNoShowPercentage() << ", "
00784     << iBookingClass.getCancellationPercentage() << ", "
00785     << iBookingClass.getNbOfBookings() << ", "
00786     << iBookingClass.getNbOfGroupBookings() << " ("
00787     << iBookingClass.getNbOfPendingGroupBookings() << ")", "
00788     << iBookingClass.getNbOfStaffBookings() << ", "
00789     << iBookingClass.getNbOfWLBBookings() << ", "

```

```

00795         << iBookingClass.getETB() << " , "
00796         << iBookingClass.getNetClassAvailability() << " , "
00797         << iBookingClass.getNetRevenueAvailability() << " , "
00798         << iBookingClass.getSegmentAvailability() << " , "
00799         << std::endl;
00800     }
00801
00802     ///////////////////////////////////////////////////////////////////
00803 void BomDisplay::csvBookingClassDisplay (std::ostream& oStream,
00804                                         const FlightDate& iFlightDate) {
00805     // Save the formatting flags for the given STL output stream
00806     FlagSaver flagSaver (oStream);
00807
00808     // Headers
00809     oStream << "*****" << std::endl;
00810     oStream << "Subclasses:" << std::endl
00811     << "-----" << std::endl;
00812     oStream << "Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), "
00813     << "Nego, NS%, OB%, "
00814     << "Bkgs, GrpBks (pdg), StfBkgs, WLBkgs, ETB, "
00815     << "ClassAvl, RevAvl, SegAvl, "
00816     << std::endl;
00817
00818     // Retrieve the key of the flight-date
00819     const AirlineCode_T& lAirlineCode = iFlightDate.getAirlineCode();
00820     const FlightNumber_T& lFlightNumber = iFlightDate.getFlightNumber();
00821     const Date_T& lFlightDateDate = iFlightDate.getDepartureDate();
00822
00823     // Check whether there are SegmentDate objects
00824     if (BomManager::hasList<SegmentDate> (iFlightDate) == false) {
00825         return;
00826     }
00827
00828     // Browse the segment-dates
00829     const SegmentDateList_T& lSegmentDateList =
00830     BomManager::getList<SegmentDate> (iFlightDate);
00831     for (SegmentDateList_T::const_iterator itSD = lSegmentDateList.begin();
00832          itSD != lSegmentDateList.end(); ++itSD) {
00833         const SegmentDate* lSD_ptr = *itSD;
00834         assert (lSD_ptr != NULL);
00835
00836         // Retrieve the key of the segment-date, as well as its dates
00837         const Date_T& lSegmentDateDate = lSD_ptr->getBoardingDate();
00838         const AirportCode_T& lBoardPoint = lSD_ptr->getBoardingPoint();
00839         const AirportCode_T& lOffPoint = lSD_ptr->getOffPoint();
00840
00841         // Browse the segment-cabins
00842         const SegmentCabinList_T& lSegmentCabinList =
00843         BomManager::getList<SegmentCabin> (*lSD_ptr);
00844         for (SegmentCabinList_T::const_iterator itSC = lSegmentCabinList.begin();
00845              itSC != lSegmentCabinList.end(); ++itSC) {
00846             const SegmentCabin* lSC_ptr = *itSC;
00847             assert (lSC_ptr != NULL);
00848
00849             // Retrieve the key of the segment-cabin
00850             const CabinCode_T& lCabinCode = lSC_ptr->getCabinCode();
00851
00852             // Build the leading string to be displayed
00853             std::ostringstream oSCLeadingStr;
00854             oSCLeadingStr << lAirlineCode << lFlightNumber << " "
00855             << lFlightDateDate << " , "
00856             << lBoardPoint << "-" << lOffPoint << " "
00857             << lSegmentDateDate << " , "
00858             << lCabinCode << " , ";
00859
00860             // Default Fare Family code, when there are no FF
00861             FamilyCode_T lFamilyCode ("NoFF");
00862
00863             // Check whether there are FareFamily objects
00864             if (BomManager::hasList<FareFamily> (*lSC_ptr) == true) {
00865
00866                 // Browse the fare families
00867                 const FareFamilyList_T& lFareFamilyList =
00868                 BomManager::getList<FareFamily> (*lSC_ptr);
00869                 for (FareFamilyList_T::const_iterator itFF = lFareFamilyList.begin();
00870                      itFF != lFareFamilyList.end(); ++itFF) {
00871                     const FareFamily* lFF_ptr = *itFF;
00872                     assert (lFF_ptr != NULL);
00873
00874                     // Retrieve the key of the segment-cabin
00875                     lFamilyCode = lFF_ptr->getFamilyCode();
00876
00877                     // Complete the leading string to be displayed
00878                     std::ostringstream oFFLeadingStr;
00879                     oFFLeadingStr << oSCLeadingStr.str() << lFamilyCode << " , ";
00880
00881                     // Browse the booking-classes

```

```

00882         const BookingClassList_T& lBookingClassList =
00883             BomManager::getList<BookingClass> (*lFF_ptr);
00884         for (BookingClassList_T::const_iterator itBC =
00885             lBookingClassList.begin();
00886             itBC != lBookingClassList.end(); ++itBC) {
00887             const BookingClass* lBC_ptr = *itBC;
00888             assert (lBC_ptr != NULL);
00889
00890             //
00891             csvBookingClassDisplay (oStream, *lBC_ptr, oFFLeadingStr.str());
00892         }
00893     }
00894
00895     // Go on to the next segment-cabin
00896     continue;
00897 }
00898 assert (BomManager::hasList<FareFamily> (*lSC_ptr) == false);
00899
00900 // The fare family code is a fake one ('NoFF'), and therefore
00901 // does not vary
00902 std::ostringstream oFFLeadingStr;
00903 oFFLeadingStr << oSCLeadingStr.str() << lFamilyCode << ", ";
00904
00905 // Browse the booking-classes, directly from the segment-cabin object
00906 const BookingClassList_T& lBookingClassList =
00907     BomManager::getList<BookingClass> (*lSC_ptr);
00908 for (BookingClassList_T::const_iterator itBC =
00909     lBookingClassList.begin();
00910     itBC != lBookingClassList.end(); ++itBC) {
00911     const BookingClass* lBC_ptr = *itBC;
00912     assert (lBC_ptr != NULL);
00913
00914     //
00915     csvBookingClassDisplay (oStream, *lBC_ptr, oFFLeadingStr.str());
00916 }
00917 }
00918 }
00919 oStream << "*****" << std::endl;
00920 }
00921
00922 // //////////////////////////////////////
00923 void BomDisplay::
00924 csvDisplay (std::ostream& oStream,
00925     const TravelSolutionList_T& iTravelSolutionList) {
00926
00927     // Save the formatting flags for the given STL output stream
00928     FlagSaver flagSaver (oStream);
00929
00930     oStream << "Travel solutions:";
00931     unsigned short idx = 0;
00932     for (TravelSolutionList_T::const_iterator itTS =
00933         iTravelSolutionList.begin();
00934         itTS != iTravelSolutionList.end(); ++itTS, ++idx) {
00935         const TravelSolutionStruct& lTS = *itTS;
00936
00937         oStream << std::endl;
00938         oStream << "    [" << idx << "] " << lTS.display();
00939     }
00940 }
00941
00942 // //////////////////////////////////////
00943 void BomDisplay::
00944 csvDisplay (std::ostream& oStream,
00945     const DatePeriodList_T& iDatePeriodList) {
00946
00947     // Save the formatting flags for the given STL output stream
00948     FlagSaver flagSaver (oStream);
00949
00950     // Browse the date-period objects
00951     for (DatePeriodList_T::const_iterator itDP = iDatePeriodList.begin();
00952         itDP != iDatePeriodList.end(); ++itDP) {
00953         const DatePeriod* lDP_ptr = *itDP;
00954         assert (lDP_ptr != NULL);
00955
00956         // Display the date-period object
00957         csvDateDisplay (oStream, *lDP_ptr);
00958     }
00959 }
00960
00961 // //////////////////////////////////////
00962 void BomDisplay::csvSimFQTAirRACDisplay (std::ostream& oStream,
00963     const BomRoot& iBomRoot) {
00964
00965     // Save the formatting flags for the given STL output stream
00966     FlagSaver flagSaver (oStream);
00967
00968     oStream << std::endl;
00969     oStream << "=====

```

```

00972         << std::endl;
00973     oStream << "BomRoot: " << iBomRoot.describeKey() << std::endl;
00974     oStream << "=====
00975         << std::endl;
00976
00977     // Check whether there are airport-pair objects
00978     if (BomManager::hasList<AirportPair> (iBomRoot) == false) {
00979         return;
00980     }
00981
00982     // Browse the airport-pair objects
00983     const AirportPairList_T& lAirportPairList =
00984         BomManager::getList<AirportPair> (iBomRoot);
00985     for (AirportPairList_T::const_iterator itAir = lAirportPairList.begin();
00986         itAir != lAirportPairList.end(); ++itAir ) {
00987         const AirportPair* lAir_ptr = *itAir;
00988         assert (lAir_ptr != NULL);
00989
00990         // Display the airport pair object
00991         csvAirportPairDisplay (oStream, *lAir_ptr);
00992     }
00993 }
00994
00995 // //////////////////////////////////////
00996 void BomDisplay::csvAirportPairDisplay (std::ostream& oStream,
00997                                         const AirportPair& iAirportPair) {
00998     // Save the formatting flags for the given STL output stream
00999     FlagSaver flagSaver (oStream);
01000
01001     oStream << "+++++" << std::endl;
01002     oStream << "AirportPair: " << iAirportPair.describeKey() << std::endl;
01003     oStream << "+++++" << std::endl;
01004
01005     // Check whether there are date-period objects
01006     if (BomManager::hasList<DatePeriod> (iAirportPair) == false) {
01007         return;
01008     }
01009
01010     // Browse the date-period objects
01011     const DatePeriodList_T& lDatePeriodList =
01012         BomManager::getList<DatePeriod> (iAirportPair);
01013     for (DatePeriodList_T::const_iterator itDP = lDatePeriodList.begin();
01014         itDP != lDatePeriodList.end(); ++itDP) {
01015         const DatePeriod* lDP_ptr = *itDP;
01016         assert (lDP_ptr != NULL);
01017
01018         // Display the date-period object
01019         csvDateDisplay (oStream, *lDP_ptr);
01020     }
01021 }
01022
01023 // //////////////////////////////////////
01024 void BomDisplay::csvDateDisplay (std::ostream& oStream,
01025                                 const DatePeriod& iDatePeriod) {
01026     // Save the formatting flags for the given STL output stream
01027     FlagSaver flagSaver (oStream);
01028
01029     oStream << "-----" << std::endl;
01030     oStream << "DatePeriod: " << iDatePeriod.describeKey() << std::endl;
01031     oStream << "-----" << std::endl;
01032
01033     // Check whether there are pos-channel objects
01034     if (BomManager::hasList<PosChannel> (iDatePeriod) == false) {
01035         return;
01036     }
01037
01038     // Browse the pos-channel objects
01039     const PosChannelList_T& lPosChannelList =
01040         BomManager::getList<PosChannel> (iDatePeriod);
01041     for (PosChannelList_T::const_iterator itPC = lPosChannelList.begin();
01042         itPC != lPosChannelList.end(); ++itPC) {
01043         const PosChannel* lPC_ptr = *itPC;
01044         assert (lPC_ptr != NULL);
01045
01046         // Display the pos-channel object
01047         csvPosChannelDisplay (oStream, *lPC_ptr);
01048     }
01049 }
01050
01051 // //////////////////////////////////////
01052 void BomDisplay::csvPosChannelDisplay (std::ostream& oStream,
01053                                       const PosChannel& iPosChannel) {
01054     // Save the formatting flags for the given STL output stream
01055     FlagSaver flagSaver (oStream);
01056
01057     oStream << "*****" << std::endl;

```

```

01068     oStream << "PosChannel: " << iPosChannel.describeKey() << std::endl;
01069     oStream << "*****" << std::endl;
01070
01071     // Check whether there are time-period objects
01072     if (BomManager::hasList<TimePeriod> (iPosChannel) == false) {
01073         return;
01074     }
01075
01076     // Browse the time-period objects
01077     const TimePeriodList_T& lTimePeriodList =
01078         BomManager::getList<TimePeriod> (iPosChannel);
01079     for (TimePeriodList_T::const_iterator itTP = lTimePeriodList.begin();
01080          itTP != lTimePeriodList.end(); ++itTP) {
01081         const TimePeriod* lTP_ptr = *itTP;
01082         assert (lTP_ptr != NULL);
01083
01084         // Display the time-period object
01085         csvTimeDisplay (oStream, *lTP_ptr);
01086     }
01087 }
01088
01089 // //////////////////////////////////////
01090 void BomDisplay::csvTimeDisplay (std::ostream& oStream,
01091                                 const TimePeriod& iTimePeriod) {
01092
01093     // Save the formatting flags for the given STL output stream
01094     FlagSaver flagSaver (oStream);
01095
01096     oStream << "-----" << std::endl;
01097     oStream << "TimePeriod: " << iTimePeriod.describeKey() << std::endl;
01098     oStream << "-----" << std::endl;
01099
01100     // Only one of the fare/yield feature list exists. Each of the following
01101     // two methods will check for the existence of the list. So, only the
01102     // existing list will be actually displayed.
01103     csvFeatureListDisplay<FareFeatures> (oStream, iTimePeriod);
01104     csvFeatureListDisplay<YieldFeatures> (oStream, iTimePeriod);
01105 }
01106
01107 // //////////////////////////////////////
01108 template <typename FEATURE_TYPE>
01109 void BomDisplay::csvFeatureListDisplay (std::ostream& oStream,
01110                                         const TimePeriod& iTimePeriod) {
01111
01112     // Check whether there are fare/yield-feature objects
01113     if (BomManager::hasList<FEATURE_TYPE> (iTimePeriod) == false) {
01114         return;
01115     }
01116
01117     // Browse the fare/yield-feature objects
01118     typedef typename BomHolder<FEATURE_TYPE>::BomList_T FeaturesList_T;
01119     const FeaturesList_T& lFeaturesList =
01120         BomManager::getList<FEATURE_TYPE> (iTimePeriod);
01121     for (typename FeaturesList_T::const_iterator itFF = lFeaturesList.begin();
01122          itFF != lFeaturesList.end(); ++itFF) {
01123         const FEATURE_TYPE* lFF_ptr = *itFF;
01124         assert (lFF_ptr != NULL);
01125
01126         // Display the fare-features object
01127         csvFeaturesDisplay (oStream, *lFF_ptr);
01128     }
01129 }
01130
01131 // //////////////////////////////////////
01132 template <typename FEATURE_TYPE>
01133 void BomDisplay::csvFeaturesDisplay (std::ostream& oStream,
01134                                     const FEATURE_TYPE& iFeatures) {
01135
01136     // Save the formatting flags for the given STL output stream
01137     FlagSaver flagSaver (oStream);
01138
01139     oStream << "-----" << std::endl;
01140     oStream << "Fare/yield-Features: " << iFeatures.describeKey() << std::endl;
01141     oStream << "-----" << std::endl;
01142
01143     // Check whether there are airlineClassList objects
01144     if (BomManager::hasList<AirlineClassList> (iFeatures) == false) {
01145         return;
01146     }
01147
01148     // Browse the airlineClassList objects
01149     const AirlineClassListList_T& lAirlineClassListList =
01150         BomManager::getList<AirlineClassList> (iFeatures);
01151     for (AirlineClassListList_T::const_iterator itACL =
01152          lAirlineClassListList.begin();
01153          itACL != lAirlineClassListList.end(); ++itACL) {
01154         const AirlineClassList* lACL_ptr = *itACL;
01155         assert (lACL_ptr != NULL);
01156     }

```



```

01161
01162     // Display the airlineClassList object
01163     csvAirlineClassDisplay(oStream, *lACL_ptr);
01164 }
01165 }
01166
01167 // //////////////////////////////////////
01168 void BomDisplay::
01169 csvAirlineClassDisplay (std::ostream& oStream,
01170                        const AirlineClassList& iAirlineClassList) {
01171     // Save the formatting flags for the given STL output stream
01172     FlagSaver flagSaver (oStream);
01173
01174     oStream << "-----" << std::endl;
01175     oStream << "AirlineClassList: "
01176     << iAirlineClassList.describeKey() << std::endl;
01177     oStream << "-----" << std::endl;
01178 }
01179 }
01180 }
01181 }
01182 }
01183 }
01184

```

### 33.181 stdair/bom/BomDisplay.hpp File Reference

```

#include <iosfwd>
#include <stdair/bom/TravelSolutionTypes.hpp>
#include <stdair/bom/DatePeriodTypes.hpp>

```

#### Classes

- class [stdair::BomDisplay](#)  
*Utility class to display StdAir objects with a pretty format.*

#### Namespaces

- [stdair](#)  
*Handle on the StdAir library context.*

### 33.182 BomDisplay.hpp

```

00001 #ifndef __STDAIR_BOM_BOMDISPLAY_HPP
00002 #define __STDAIR_BOM_BOMDISPLAY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 // StdAir
00010 #include <stdair/bom/TravelSolutionTypes.hpp>
00011 #include <stdair/bom/DatePeriodTypes.hpp>
00012
00013 namespace stdair {
00014
00015     class BomRoot;
00016     class Inventory;
00017     class FlightDate;
00018     class LegDate;
00019     class SegmentDate;
00020     class LegCabin;
00021     class SegmentCabin;
00022     class FareFamily;
00023     class BookingClass;
00024     class AirportPair;
00025     class PosChannel;
00026     class DatePeriod;
00027     class TimePeriod;
00028     class FareFeatures;
00029     class YieldFeatures;
00030     class AirlineClassList;
00031     class OnDDate;
00032

```

```

00033
00038 class BomDisplay {
00039 public:
00040     // //////////////// Display support methods ////////////////
00041
00056     static void list (std::ostream&, const BomRoot&,
00057                     const AirlineCode_T& iAirlineCode = "all",
00058                     const FlightNumber_T& iFlightNumber = 0);
00059
00073     static void list (std::ostream&, const Inventory&,
00074                     const unsigned short iInventoryIndex = 0,
00075                     const FlightNumber_T& iFlightNumber = 0);
00076
00085     static void listAirportPairDateRange (std::ostream&,
00086                                         const BomRoot&);
00087
00096     static void csvDisplay (std::ostream&, const BomRoot&);
00097
00106     static void csvDisplay (std::ostream&, const Inventory&);
00107
00115     static void csvDisplay (std::ostream&, const OnDDate&);
00116
00125     static void csvDisplay (std::ostream&, const FlightDate&);
00126
00135     static void csvLegDateDisplay (std::ostream&, const
00136 FlightDate&);
00145     static void csvSegmentDateDisplay (std::ostream&, const
00146 FlightDate&);
00155     static void csvLegCabinDisplay (std::ostream&, const
00156 FlightDate&);
00165     static void csvSegmentCabinDisplay (std::ostream&, const
00166 FlightDate&);
00175     static void csvFareFamilyDisplay (std::ostream&, const
00176 FlightDate&);
00185     static void csvBucketDisplay (std::ostream&, const
00186 FlightDate&);
00196     static void csvBookingClassDisplay (std::ostream&, const
00197 BookingClass&,
00198                                         const std::string& iLeadingString);
00206     static void csvBookingClassDisplay (std::ostream&, const
00207 FlightDate&);
00216     static void csvDisplay (std::ostream&, const TravelSolutionList_T&);
00217
00226     static void csvDisplay (std::ostream&, const DatePeriodList_T&);
00227
00236     static void csvSimFQTAirRACDisplay (std::ostream&, const
00237 BomRoot&);
00247     static void csvAirportPairDisplay (std::ostream&, const
00248 AirportPair&);
00258     static void csvDateDisplay (std::ostream&, const DatePeriod&);
00259
00269     static void csvPosChannelDisplay (std::ostream&, const
00270 PosChannel&);
00280     static void csvTimeDisplay (std::ostream&, const TimePeriod&);
00281
00290     template <typename FEATURE_TYPE>
00291     static void csvFeatureListDisplay (std::ostream& oStream, const
00292 TimePeriod&);
00293
00301     template <typename FEATURE_TYPE>
00302     static void csvFeaturesDisplay (std::ostream& oStream, const FEATURE_TYPE&);
00303
00312     static void csvAirlineClassDisplay (std::ostream&, const
00313 AirlineClassList&);
00314 };
00315 }
00316 #endif // __STDAIR_BOM_BOMDISPLAY_HPP

```

### 33.183 stdair/bom/BomHolder.hpp File Reference

```
#include <iosfwd>
```

```
#include <string>
#include <list>
#include <map>
#include <stdair/bom/key_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/BomHolderKey.hpp>
```

## Classes

- class [stdair::BomHolder< BOM >](#)

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

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.184 BomHolder.hpp

```
00001 #ifndef __STDAIR_BOM_BOMHOLDER_HPP
00002 #define __STDAIR_BOM_BOMHOLDER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 #include <list>
00011 #include <map>
00012 // StdAir
00013 #include <stdair/bom/key_types.hpp>
00014 #include <stdair/bom/BomAbstract.hpp>
00015 #include <stdair/bom/BomHolderKey.hpp>
00016
00017 namespace stdair {
00018
00019     template <typename BOM>
00020     class BomHolder : public stdair::BomAbstract {
00021     public:
00022         template <typename> friend class FacBom;
00023         friend class FacBomManager;
00024
00025     public:
00026         // ////////////////////////////////// Type definitions //////////////////////////////////
00027         typedef stdair::BomHolderKey Key_T;
00028
00029         typedef std::list<BOM*> BomList_T;
00030
00031         typedef std::map<const MapKey_T, BOM*> BomMap_T;
00032
00033     public:
00034         // ////////////////////////////////// Display support methods //////////////////////////////////
00035         void toStream (std::ostream& ioOut) const {
00036             ioOut << toString();
00037         }
00038
00039         void fromStream (std::istream& ioIn) {
00040
00041         }
00042
00043         std::string toString() const {
00044             return "BomHolder";
00045         }
00046
00047         const std::string describeKey() const {
00048             return "BomHolder";
00049         }
00050
00051     protected:
00052         BomHolder();
00053
00054         BomHolder (const BomHolder&);
00055
00056 }
```

```

00094     BomHolder (const Key_T& iKey) : _key (iKey) { }
00095
00099     ~BomHolder() { };
00100
00101     public:
00102         // ////////////////////////////////// Attributes //////////////////////////////////
00106         Key_T _key;
00107
00111         BomList_T _bomList;
00112
00116         BomMap_T _bomMap;
00117     };
00118
00119 }
00120 #endif // __STDAIR_BOM_BOMHOLDER_HPP

```

### 33.185 stdair/bom/BomHolderKey.cpp File Reference

```

#include <ostream>
#include <sstream>
#include <stdair/bom/BomHolderKey.hpp>

```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.186 BomHolderKey.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <ostream>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/bom/BomHolderKey.hpp>
00009
00010 namespace stdair {
00011
00012     // //////////////////////////////////////
00013     BomHolderKey::BomHolderKey () {
00014     }
00015
00016     // //////////////////////////////////////
00017     BomHolderKey::~BomHolderKey () {
00018     }
00019
00020     // //////////////////////////////////////
00021     void BomHolderKey::toStream (std::ostream& ioOut) const {
00022         ioOut << "BomHolderKey: " << toString() << std::endl;
00023     }
00024
00025     // //////////////////////////////////////
00026     void BomHolderKey::fromStream (std::istream& ioIn) {
00027     }
00028
00029     // //////////////////////////////////////
00030     const std::string BomHolderKey::toString() const {
00031         std::ostringstream oStr;
00032         oStr << " -- HOLDER -- ";
00033         return oStr.str();
00034     }
00035
00036 }

```

### 33.187 stdair/bom/BomHolderKey.hpp File Reference

```

#include <stdair/bom/KeyAbstract.hpp>

```

## Classes

- struct [stdair::BomHolderKey](#)

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.188 BomHolderKey.hpp

```

00001 #ifndef __STDAIR_BOM_BOMHOLDERKEY_HPP
00002 #define __STDAIR_BOM_BOMHOLDERKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STDAIR
00008 #include <stdair/bom/KeyAbstract.hpp>
00009
00010 namespace stdair {
00012     struct BomHolderKey : public KeyAbstract {
00013
00014     public:
00015         // ////////////////////////////////// Construction //////////////////////////////////
00017         BomHolderKey ();
00019         ~BomHolderKey ();
00020
00021         // ////////////////////////////////// Display support methods //////////////////////////////////
00024         void toStream (std::ostream& ioOut) const;
00025
00028         void fromStream (std::istream& ioIn);
00029
00035         const std::string toString() const;
00036
00038         const std::string describe() const;
00039
00040     };
00041
00042 }
00043 #endif // __STDAIR_BOM_BOMHOLDERKEY_HPP

```

## 33.189 stdair/bom/BomID.hpp File Reference

```

#include <iosfwd>
#include <string>

```

## Classes

- struct [stdair::BomID< BOM >](#)

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

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.190 BomID.hpp

```

00001 #ifndef __STDAIR_BOM_BOMID_HPP
00002 #define __STDAIR_BOM_BOMID_HPP
00003
00004 // //////////////////////////////////////

```

```

00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010
00011 namespace stdair {
00012
00016     template <typename BOM>
00017     struct BomID {
00018
00019     public:
00020         // //////////// Getters ////////////
00024         BOM& getObject () const;
00025
00026     public:
00027         // //////////// Constructors and destructors ////////////
00031         BomID (BOM& iBOM);
00032
00036         BomID (const BomID&);
00037
00041         ~BomID();
00042
00043     private:
00047         BomID();
00048
00049     private:
00050         // //////////// Attributes ////////////
00054         BOM* _id;
00055     };
00056
00057 // //////////////////////////////////////
00058 template <typename BOM> BomID<BOM>::BomID (BOM& iBOM): _id (&iBOM) { }
00059
00060 // //////////////////////////////////////
00061 template <typename BOM> BomID<BOM>::BomID (const BomID& iBomID)
00062     : _id (iBomID._id) { }
00063
00064 // //////////////////////////////////////
00065 template <typename BOM> BomID<BOM>::~BomID () { }
00066
00067 // //////////////////////////////////////
00068 template <typename BOM> BOM& BomID<BOM>::getObject () const {
00069     assert (_id != NULL);
00070     return *_id;
00071 }
00072 }
00073 #endif // __STDAIR_BOM_BOMID_HPP

```

### 33.191 stdair/bom/BomIDTypes.hpp File Reference

```

#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
#include <stdair/bom/BomID.hpp>

```

#### Namespaces

- [stdair](#)  
Handle on the StdAir library context.

#### Typedefs

- typedef struct BomID< BookingClass > [stdair::BookingClassID\\_T](#)
- typedef std::list< BookingClassID\_T > [stdair::BookingClassIDList\\_T](#)

### 33.192 BomIDTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_BOMIDTYPES_HPP

```

```

00003 #define __STDAIR_BOM_BOMIDTYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // StdAir
00012 #include <stdair/bom/key_types.hpp>
00013 #include <stdair/bom/BomID.hpp>
00014
00015 namespace stdair {
00016
00017     // Forward declarations.
00018     class BookingClass;
00019
00021     typedef struct BomID<BookingClass> BookingClassID_T;
00022
00024     typedef std::list<BookingClassID_T> BookingClassIDList_T;
00025 }
00026 #endif // __STDAIR_BOM_BOMIDTYPES_HPP
00027

```

### 33.193 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

- [bpt](#)
- [stdair](#)

*Handle on the StdAir library context.*

#### Typedefs

- typedef char [bpt::ptree](#)

### 33.194 BomINIImport.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 #if BOOST_VERSION >= 104100
00008 // Boost Property Tree
00009 #include <boost/property_tree/ptree.hpp>
00010 #include <boost/property_tree/ini_parser.hpp>
00011 #endif // BOOST_VERSION >= 104100
00012 // StdAir
00013 #include <stdair/basic/BasFileMgr.hpp>
00014 #include <stdair/bom/BomINIImport.hpp>
00015 #include <stdair/bom/ConfigHolderStruct.hpp>
00016 #include <stdair/service/Logger.hpp>
00017
00018 #if BOOST_VERSION >= 104100
00019 namespace bpt = boost::property_tree;
00020 #else // BOOST_VERSION >= 104100
00021 namespace bpt {
00022     typedef char ptree;
00023 }
00024 #endif // BOOST_VERSION >= 104100

```

```

00025
00026 namespace stdair {
00027
00028 // //////////////////////////////////////
00029 void BomINIImport::importINIConfig (
    ConfigHolderStruct& iConfigHolder,
00030                                     const ConfigINIFile& iConfigINIFile) {
00031
00032     // Get the config file name.
00033     const stdair::Filename_T lFilename = iConfigINIFile.name();
00034
00035     // Check that the file path given as input corresponds to an actual file
00036     const bool doesExistAndIsReadable =
00037         stdair::BasFileMgr::doesExistAndIsReadable (lFilename);
00038     if (doesExistAndIsReadable == false) {
00039         STDAIR_LOG_DEBUG ("The config input file '" << lFilename
00040                         << "' can not be retrieved on the file-system.");
00041         return;
00042     }
00043     STDAIR_LOG_DEBUG ("Load the config input file '" << lFilename
00044                     << "' content into the configuration holder.");
00045
00046 #if BOOST_VERSION >= 104100
00047
00048     // Transform the INI file into a BOOST property tree.
00049     bpt::ptree pt;
00050     bpt::ini_parser::read_ini(lFilename, pt);
00051     // Add the property tree to the configuration structure.
00052     iConfigHolder.add(pt);
00053
00054 #endif // BOOST_VERSION >= 104100
00055 }
00056
00057 }

```

### 33.195 stdair/bom/BomINIImport.hpp File Reference

```

#include <string>
#include <stdair/stdair_file.hpp>

```

#### Classes

- class [stdair::BomINIImport](#)  
*Utility class to import StdAir objects in a INI format.*

#### Namespaces

- [stdair](#)  
*Handle on the StdAir library context.*

### 33.196 BomINIImport.hpp

```

00001 #ifndef __STDAIR_BOM_BOMINIIMPORT_HPP
00002 #define __STDAIR_BOM_BOMINIIMPORT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/stdair_file.hpp>
00011
00012 namespace stdair {
00013
00014     struct ConfigHolderStruct;
00015
00021     class BomINIImport {
00022     public:
00023         // ////////////////////////////////// Import support methods //////////////////////////////////

```



```

00029     static void importINIConfig (ConfigHolderStruct&,
00030                                   const ConfigINIFile&);
00031
00032     };
00033 }
00034 #endif // __STDAIR_BOM_BOMINIIMPORT_HPP

```

### 33.197 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

- [stdair](#)

*Handle on the StdAir library context.*

### 33.198 BomJSONExport.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <ostream>
00007 #if BOOST_VERSION >= 103400
00008 // Boost ForEach
00009 #include <boost/foreach.hpp>
00010 #endif // BOOST_VERSION >= 103400
00011 // StdAir
00012 #include <stdair/stdair_date_time_types.hpp>
00013 #include <stdair/basic/BasConst_BomDisplay.hpp>
00014 #include <stdair/bom/BomManager.hpp>
00015 #include <stdair/bom/BomRoot.hpp>
00016 #include <stdair/bom/Inventory.hpp>
00017 #include <stdair/bom/FlightDate.hpp>
00018 #include <stdair/bom/LegDate.hpp>
00019 #include <stdair/bom/SegmentDate.hpp>
00020 #include <stdair/bom/LegCabin.hpp>
00021 #include <stdair/bom/SegmentCabin.hpp>
00022 #include <stdair/bom/FareFamily.hpp>
00023 #include <stdair/bom/BookingClass.hpp>
00024 #include <stdair/bom/Bucket.hpp>
00025 #include <stdair/bom/EventStruct.hpp>
00026 #include <stdair/bom/EventTypes.hpp>
00027 #include <stdair/bom/BookingRequestStruct.hpp>
00028 #include <stdair/bom/BreakPointStruct.hpp>
00029 #include <stdair/bom/BomJSONExport.hpp>

```

```

00030
00031 namespace stdair {
00032
00033 // //////////////////////////////////////
00034 void BomJSONExport::
00035 jsonExportFlightDateList (std::ostream& oStream,
00036                          const BomRoot& iBomRoot,
00037                          const AirlineCode_T& iAirlineCode,
00038                          const FlightNumber_T& iFlightNumber) {
00039
00040     // Check whether there are Inventory objects
00041     if (BomManager::hasList<Inventory> (iBomRoot) == false) {
00042         return;
00043     }
00044
00045 #if BOOST_VERSION >= 104100
00046
00047     // Create empty property tree objects
00048     bpt::ptree pt;
00049     bpt::ptree ptInventoryList;
00050
00051     // Browse the inventories
00052     const InventoryList_T& lInventoryList =
00053         BomManager::getList<Inventory> (iBomRoot);
00054     for (InventoryList_T::const_iterator itInv = lInventoryList.begin();
00055          itInv != lInventoryList.end(); ++itInv) {
00056         const Inventory* lInv_ptr = *itInv;
00057         assert (lInv_ptr != NULL);
00058
00059         // Retrieve the inventory key (airline code)
00060         const AirlineCode_T& lAirlineCode = lInv_ptr->getAirlineCode();
00061
00062         // Display only the requested inventories
00063         if (iAirlineCode == "all" || iAirlineCode == lAirlineCode){
00064
00065             // Flight date tree
00066             bpt::ptree ptFD;
00067             // Create an empty flight-dates array
00068             bpt::ptree lFDDatePropertyTree;
00069
00070             // Check whether there are FlightDate objects
00071             if (BomManager::hasMap<FlightDate> (*lInv_ptr) == false) {
00072                 return;
00073             }
00074
00075             // Browse the flight-dates
00076             const FlightDateMap_T& lFlightDateList =
00077                 BomManager::getMap<FlightDate> (*lInv_ptr);
00078             for (FlightDateMap_T::const_iterator itFD = lFlightDateList.begin();
00079                  itFD != lFlightDateList.end(); ++itFD) {
00080                 const FlightDate* lFD_ptr = itFD->second;
00081                 assert (lFD_ptr != NULL);
00082
00083                 // Retrieve the key of the flight-date
00084                 const FlightNumber_T& lFlightNumber = lFD_ptr->
getFlightNumber();
00085                 const Date_T& lFlightDateDate = lFD_ptr->getDepartureDate();
00086
00087                 // Display only the requested flight number
00088                 if (iFlightNumber == 0 || iFlightNumber == lFlightNumber) {
00089
00090                     // Add the airline code to the inventory tree
00091                     ptFD.put ("airline_code", lAirlineCode);
00092                     // Put flight number in property tree
00093                     ptFD.put ("number", lFlightNumber);
00094                     // Put flight date date in property tree
00095                     ptFD.put ("date", lFlightDateDate);
00096
00097                     // Put the current flight date tree in the array
00098                     ptInventoryList.push_back(std::make_pair("", ptFD));
00099
00100                 }
00101             }
00102
00103         }
00104     }
00105
00106     // Store the inventory(ies) array tree into the global tree
00107     pt.add_child ("inventories", ptInventoryList);
00108
00109     // Write the property tree into the JSON stream.
00110     write_json (oStream, pt);
00111 #endif // BOOST_VERSION >= 104100
00112 }
00113
00114 // //////////////////////////////////////
00115 void BomJSONExport::jsonExportFlightDate (bpt::ptree& ioFDPropertyTree,

```

```

00116                                     const Inventory& iInventory,
00117                                     const FlightNumber_T& iFlightNumber) {
00118
00119     // Check whether there are FlightDate objects
00120     if (BomManager::hasMap<FlightDate> (iInventory) == false) {
00121         return;
00122     }
00123
00124 #if BOOST_VERSION >= 104100
00125     // Create an empty flight-dates array
00126     bpt::ptree lFDDatePropertyTree;
00127
00128     // Browse the flight-dates
00129     const FlightDateMap_T& lFlightDateList =
00130         BomManager::getMap<FlightDate> (iInventory);
00131     for (FlightDateMap_T::const_iterator itFD = lFlightDateList.begin();
00132          itFD != lFlightDateList.end(); ++itFD) {
00133         const FlightDate* lFD_ptr = itFD->second;
00134         assert (lFD_ptr != NULL);
00135
00136         // Retrieve the key of the flight-date
00137         const FlightNumber_T& lFlightNumber = lFD_ptr->getFlightNumber();
00138         const Date_T& lFlightDateDate = lFD_ptr->getDepartureDate();
00139
00140         // Display only the requested flight number
00141         if (iFlightNumber == 0 || iFlightNumber == lFlightNumber) {
00142             // Create an empty property tree object for the current flight date
00143             bpt::ptree lCurrFDTree;
00144
00145             // Put flight number in property tree
00146             lCurrFDTree.put ("number", lFlightNumber);
00147             // Put flight date date in property tree
00148             lCurrFDTree.put ("date", lFlightDateDate);
00149
00150             // Put the current flight date tree in the flight date array
00151             ioFDDPropertyTree.push_back(std::make_pair("", lCurrFDTree));
00152         }
00153     }
00154 #endif // BOOST_VERSION >= 104100
00155 }
00156
00157 // //////////////////////////////////////
00158 void BomJSONExport::
00159     jsonExportFlightDateObjects (std::ostream& oStream,
00160                                 const FlightDate& iFlightDate) {
00161
00162 #if BOOST_VERSION >= 104100
00163     // Create an empty property tree object
00164     bpt::ptree pt;
00165
00166     // Put the airline code in property tree
00167     const AirlineCode_T& lAirlineCode = iFlightDate.getAirlineCode();
00168     pt.put ("flight_date.airline_code", lAirlineCode);
00169
00170     // Put the flight number in property tree
00171     const FlightNumber_T& lFlightNumber = iFlightDate.
00172         getFlightNumber();
00173     pt.put ("flight_date.flight_number", lFlightNumber);
00174
00175     // Put the flight departure date in property tree
00176     const Date_T& lFlightDateDate = iFlightDate.getDepartureDate();
00177     const std::string& lDepartureDateStr =
00178         boost::gregorian::to_simple_string (lFlightDateDate);
00179     pt.put ("flight_date.departure_date", lDepartureDateStr);
00180
00181     // Create an empty legs array
00182     bpt::ptree ptLegs;
00183
00184     // Recursively construct the legs array
00185     jsonExportLegDate (ptLegs, iFlightDate);
00186
00187     // Add legs tree to the global property tree
00188     pt.add_child ("flight_date.legs", ptLegs);
00189
00190     // Create an empty segments array
00191     bpt::ptree ptSegments;
00192
00193     // Recursively construct the segments array
00194     jsonExportSegmentDate (ptSegments, iFlightDate);
00195
00196     // Add segments tree to the global property tree
00197     pt.add_child ("flight_date.segments", ptSegments);
00198
00199
00200
00201
00202
00203
00204
00205
00206
00207
00208
00209
00210

```

```

00211
00212 // Write the property tree into the JSON stream.
00213 write_json (oStream, pt);
00214
00215 #endif // BOOST_VERSION >= 104100
00216 }
00217
00218 ///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00219 void BomJSONExport::jsonExportLegDate (bpt::ptree& ioLegDateListTree,
00220                                       const FlightDate& iFlightDate) {
00221
00222 // Check whether there are LegDate objects
00223 if (BomManager::hasList<LegDate> (iFlightDate) == false) {
00224     return;
00225 }
00226
00227 // Browse the leg-dates
00228 const LegDateList_T& lLegDateList =
00229     BomManager::getList<LegDate> (iFlightDate);
00230 for (LegDateList_T::const_iterator itLD = lLegDateList.begin();
00231      itLD != lLegDateList.end(); ++itLD) {
00232     const LegDate* lLD_ptr = *itLD;
00233     assert (lLD_ptr != NULL);
00234
00235 #if BOOST_VERSION >= 104100
00236
00237 // Create an empty property tree object for the current leg date
00238 bpt::ptree lCurrLDTree;
00239
00240 // Put boarding point in property tree
00241 const AirportCode_T& lBoardingPoint = lLD_ptr->getBoardingPoint();
00242 lCurrLDTree.put ("board_point", lBoardingPoint);
00243 // Put off point in property tree
00244 const AirportCode_T& lOffPoint = lLD_ptr->getOffPoint();
00245 lCurrLDTree.put ("off_point", lOffPoint);
00246 // Put boarding date in property tree
00247 const Date_T& lBoardingDate = lLD_ptr->getBoardingDate();
00248 lCurrLDTree.put ("board_date", lBoardingDate);
00249 // Put off date in property tree
00250 const Date_T& lOffDate = lLD_ptr->getOffDate();
00251 lCurrLDTree.put ("off_dDate", lOffDate);
00252 // Put boarding time in property tree
00253 const Duration_T& lBoardingTime = lLD_ptr->getBoardingTime();
00254 lCurrLDTree.put ("board_time", lBoardingTime);
00255 // Put off time in property tree
00256 const Duration_T& lOffTime = lLD_ptr->getOffTime();
00257 lCurrLDTree.put ("off_time", lOffTime);
00258 // Put elapsed time in property tree
00259 const Duration_T& lElapsedTime = lLD_ptr->getElapsedTime();
00260 lCurrLDTree.put ("elapsed_time", lElapsedTime);
00261 // Put date offset in property tree
00262 const DateOffset_T& lDateOffset = lLD_ptr->getDateOffset();
00263 lCurrLDTree.put ("date_offset", lDateOffset);
00264 // Put time offset in property tree
00265 const Duration_T& lTimeOffset = lLD_ptr->getTimeOffset();
00266 lCurrLDTree.put ("time_offset", lTimeOffset);
00267 // Put distance in property tree
00268 const Distance_T& lDistance = lLD_ptr->getDistance();
00269 lCurrLDTree.put ("distance", lDistance);
00270 // Put capacity in property tree
00271 const CabinCapacity_T& lCapacity = lLD_ptr->getCapacity();
00272 lCurrLDTree.put ("capacity", lCapacity);
00273
00274 // Create an empty property tree object for the leg cabins array
00275 // corresponding to the current leg date.
00276 bpt::ptree lLegCabinArray;
00277
00278 // Recursively construct the leg cabins array
00279 jsonExportLegCabin (lLegCabinArray, *lLD_ptr);
00280
00281 // Add the leg cabins array to the leg date tree
00282 lCurrLDTree.add_child ("cabins", lLegCabinArray);
00283
00284 // Put the current leg date tree in the leg date list tree
00285 ioLegDateListTree.push_back (std::make_pair ("", lCurrLDTree));
00286
00287 #endif // BOOST_VERSION >= 104100
00288 }
00289 }
00290
00291 ///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00292 void BomJSONExport::jsonExportLegCabin (bpt::ptree& ioLegCabinListTree,
00293                                       const LegDate& iLegDate) {
00294
00295 // Check whether there are LegCabin objects
00296 if (BomManager::hasList<LegCabin> (iLegDate) == false) {
00297     return;

```

```

00298     }
00299
00300     // Browse the leg-cabins
00301     const LegCabinList_T& lLegCabinList =
00302         BomManager::getList<LegCabin> (iLegDate);
00303     for (LegCabinList_T::const_iterator itLC = lLegCabinList.begin();
00304          itLC != lLegCabinList.end(); ++itLC) {
00305         const LegCabin* lLC_ptr = *itLC;
00306         assert (lLC_ptr != NULL);
00307
00308         #if BOOST_VERSION >= 104100
00309
00310         // Create an empty property tree object for the current leg cabin
00311         bpt::ptree lCurrLCTree;
00312         bpt::ptree lCurrLCBPV;
00313
00314         // Put the cabin code in property tree
00315         const CabinCode_T& lCabinCode = lLC_ptr->getCabinCode();
00316         lCurrLCTree.put ("code", lCabinCode);
00317         // Put the offered capacity in property tree
00318         const CabinCapacity_T& lOfferedCapacity = lLC_ptr->getOfferedCapacity();
00319         lCurrLCTree.put ("offed_cap", lOfferedCapacity);
00320         // Put the physical capacity in property tree
00321         const CabinCapacity_T& lPhysicalCapacity = lLC_ptr->getPhysicalCapacity();
00322         lCurrLCTree.put ("phy_cap", lPhysicalCapacity);
00323         // Put regrade adjustment in property tree
00324         const CapacityAdjustment_T& lRegradeAdjustment = lLC_ptr->getRegradeAdjustment();
00325         lCurrLCTree.put ("rgd_adj", lRegradeAdjustment);
00326         // Put authorization level in property tree
00327         const AuthorizationLevel_T& lAuthorizationLevel = lLC_ptr->getAuthorizationLevel(
00328     );
00329         lCurrLCTree.put ("au", lAuthorizationLevel);
00330         // Put UPR in property tree
00331         const UPR_T& lUPR = lLC_ptr->getUPR();
00332         lCurrLCTree.put ("upr", lUPR);
00333         // Put sold seats in property tree
00334         const NbOfSeats_T& lNbOfSoldSeats = lLC_ptr->getSoldSeat();
00335         lCurrLCTree.put ("ss", lNbOfSoldSeats);
00336         // Put staff nb of seats in property tree
00337         const NbOfSeats_T& lStaffNbOfSeats = lLC_ptr->getStaffNbOfSeats();
00338         lCurrLCTree.put ("staff", lStaffNbOfSeats);
00339         // Put waiting list nb of seats in property tree
00340         const NbOfSeats_T& lWLNbOfSeats = lLC_ptr->getWLNbOfSeats();
00341         lCurrLCTree.put ("wl", lWLNbOfSeats);
00342         // Put group nb of seats in property tree
00343         const NbOfSeats_T& lGroupNbOfSeats = lLC_ptr->getGroupNbOfSeats();
00344         lCurrLCTree.put ("group", lGroupNbOfSeats);
00345         // Put committed space in property tree
00346         const CommittedSpace_T& lCommittedSpace = lLC_ptr->getCommittedSpace();
00347         lCurrLCTree.put ("comm_space", lCommittedSpace);
00348         // Put availability pool in property tree
00349         const Availability_T& lAvailabilityPool = lLC_ptr->getAvailabilityPool();
00350         lCurrLCTree.put ("av_pool", lAvailabilityPool);
00351         // Put availability in property tree
00352         const Availability_T& lAvailability = lLC_ptr->getAvailability();
00353         lCurrLCTree.put ("avl", lAvailability);
00354         // Put net availability in property tree
00355         const Availability_T& lNetAvailability = lLC_ptr->getNetAvailability();
00356         lCurrLCTree.put ("nav", lNetAvailability);
00357         // Put gross availability in property tree
00358         const Availability_T& lGrossAvailability = lLC_ptr->getGrossAvailability();
00359         lCurrLCTree.put ("gav", lGrossAvailability);
00360         // Put avg cancellation percentage in property tree
00361         const OverbookingRate_T& lAvgCancellationPercentage =
00362             lLC_ptr->getAvgCancellationPercentage();
00363         lCurrLCTree.put ("acp", lAvgCancellationPercentage);
00364         // Put ETB in property tree
00365         const NbOfSeats_T& lExpectedToBoard = lLC_ptr->getETB();
00366         lCurrLCTree.put ("etb", lExpectedToBoard);
00367         // Put current bid price in property tree
00368         const BidPrice_T& lCurrentBidPrice = lLC_ptr->getCurrentBidPrice();
00369         lCurrLCTree.put ("bid_price", lCurrentBidPrice);
00370         // Put current bid price vector in property tree
00371         const BidPriceVector_T& lCurrentBidPriceVector =
00372             lLC_ptr->getBidPriceVector();
00373         std::ostringstream ostr;
00374         BidPriceVector_T::const_iterator itBP = lCurrentBidPriceVector.begin();
00375         while (itBP != lCurrentBidPriceVector.end()) {
00376             ostr << *itBP;
00377             ++itBP;
00378             if (itBP != lCurrentBidPriceVector.end()) {
00379                 ostr << ",";
00380             }
00381         }
00382         lCurrLCTree.put ("BPV", ostr.str());
00383
00384         // Create an empty property tree object for the buckets array

```

```

00384     // corresponding to the current leg cabin.
00385     bpt::ptree lBucketTree;
00386
00387     // Recursively construct the buckets array
00388     jsonExportBucket (lBucketTree, *lC_ptr);
00389
00390     // Add the buckets array to the leg cabin tree
00391     lCurrLCTree.add_child ("buckets", lBucketTree);
00392
00393     // Put the current leg cabin tree in the leg cabin list tree
00394     ioLegCabinListTree.push_back(std::make_pair("", lCurrLCTree));
00395
00396 #endif // BOOST_VERSION >= 104100
00397 }
00398 }
00399
00400 // //////////////////////////////////////
00401 void BomJSONExport::jsonExportBucket (bpt::ptree& ioBucketListTree,
00402                                     const LegCabin& iLegCabin) {
00403
00404     // Check whether there are Bucket objects
00405     if (BomManager::hasList<Bucket> (iLegCabin) == false) {
00406         return;
00407     }
00408
00409     // Browse the buckets
00410     const BucketList_T& lBucketList = BomManager::getList<Bucket> (iLegCabin);
00411     for (BucketList_T::const_iterator itBuck = lBucketList.begin();
00412          itBuck != lBucketList.end(); ++itBuck) {
00413         const Bucket* lBucket_ptr = *itBuck;
00414         assert (lBucket_ptr != NULL);
00415
00416 #if BOOST_VERSION >= 104100
00417         // Create an empty property tree object for the current bucket
00418         bpt::ptree lCurrBucketTree;
00419
00420         // Put yield in property tree
00421         const Yield_T& lYieldRangeUpperValue =
00422             lBucket_ptr->getYieldRangeUpperValue();
00423         lCurrBucketTree.put ("yield", lYieldRangeUpperValue);
00424         // Put seat_index in property tree
00425         const SeatIndex_T& lSeatIndex = lBucket_ptr->getSeatIndex();
00426         lCurrBucketTree.put ("si", lSeatIndex);
00427         // Put sold_seats in property tree
00428         const NbofSeats_T& lSoldSeats = lBucket_ptr->getSoldSeats();
00429         lCurrBucketTree.put ("ss", lSoldSeats);
00430         // Put availability in property tree
00431         const CabinCapacity_T& lAvailability = lBucket_ptr->getAvailability();
00432         lCurrBucketTree.put ("av", lAvailability);
00433
00434         // Put the current bucket tree in the bucket list tree
00435         ioBucketListTree.push_back(std::make_pair("", lCurrBucketTree));
00436 #endif
00437     }
00438 }
00439
00440 // //////////////////////////////////////
00441 void BomJSONExport::jsonExportSegmentDate (bpt::ptree& ioSegmentDateTree,
00442                                           const FlightDate& iFlightDate) {
00443
00444     // Check whether there are SegmentDate objects
00445     if (BomManager::hasList<SegmentDate> (iFlightDate) == false) {
00446         return;
00447     }
00448
00449     // Browse the segment-dates
00450     const SegmentDateList_T& lSegmentDateList =
00451         BomManager::getList<SegmentDate> (iFlightDate);
00452     for (SegmentDateList_T::const_iterator itSD = lSegmentDateList.begin();
00453          itSD != lSegmentDateList.end(); ++itSD) {
00454         const SegmentDate* lSD_ptr = *itSD;
00455         assert (lSD_ptr != NULL);
00456
00457 #if BOOST_VERSION >= 104100
00458         // Create an empty property tree object for the current segment date
00459         bpt::ptree lCurrSDTree;
00460
00461         // Put segment key in property tree
00462         lCurrSDTree.put ("segment", lSD_ptr->toString());
00463
00464         // Create an empty property tree object for the segment cabin array
00465         // corresponding to the current segment date
00466         bpt::ptree lSegmentCabinTree;
00467
00468

```

```

00475 // Recursively construct the segment cabin array
00476 jsonExportSegmentCabin (lSegmentCabinTree, *lSD_ptr);
00477
00478 // Add the segment cabin array to the tree of the current segment date
00479 lCurrSDTree.add_child ("sub_classes", lSegmentCabinTree);
00480
00481 // Put segment date array in property tree
00482 ioSegmentDateTree.push_back(std::make_pair("", lCurrSDTree));
00483
00484 #endif // BOOST_VERSION >= 104100
00485 }
00486 }
00487
00488 // //////////////////////////////////////
00489 void BomJSONExport::jsonExportSegmentCabin (bpt::ptree& ioPropertyTree,
00490                                             const SegmentDate& iSegmentDate) {
00491
00492 // Check whether there are SegmentCabin objects
00493 if (BomManager::hasList<SegmentCabin> (iSegmentDate) == false) {
00494     return;
00495 }
00496
00497 // Browse the segment-cabins
00498 const SegmentCabinList_T& lSegmentCabinList =
00499     BomManager::getList<SegmentCabin> (iSegmentDate);
00500 for (SegmentCabinList_T::const_iterator itSC = lSegmentCabinList.begin();
00501      itSC != lSegmentCabinList.end(); ++itSC) {
00502     const SegmentCabin* lSC_ptr = *itSC;
00503     assert (lSC_ptr != NULL);
00504
00505 #if BOOST_VERSION >= 104100
00506 // Create an empty property tree object for the current segment cabin
00507 bpt::ptree lSCArray;
00508
00509 // Put cabin in property tree
00510
00511 lSCArray.put ("cabin_code", lSC_ptr->toString());
00512
00513 // Export the cabin tree to add fare-families and sub-classes details
00514 jsonExportFareFamily (ioPropertyTree, lSCArray, *lSC_ptr);
00515 #endif // BOOST_VERSION >= 104100
00516 }
00517 }
00518 }
00519 }
00520
00521 // //////////////////////////////////////
00522 void BomJSONExport::jsonExportFareFamily (bpt::ptree& ioPropertyTree,
00523                                           bpt::ptree& ioSCTree,
00524                                           const SegmentCabin& iSegmentCabin) {
00525
00526 // Check whether there are FareFamily objects
00527 if (BomManager::hasList<FareFamily> (iSegmentCabin) == true) {
00528
00529 // Browse the fare-families
00530 const FareFamilyList_T& lFareFamilyList =
00531     BomManager::getList<FareFamily> (iSegmentCabin);
00532 for (FareFamilyList_T::const_iterator itFF = lFareFamilyList.begin();
00533      itFF != lFareFamilyList.end(); ++itFF) {
00534     const FareFamily* lFF_ptr = *itFF;
00535     assert (lFF_ptr != NULL);
00536
00537 // Browse the booking-classes
00538 const BookingClassList_T& lBookingClassList =
00539     BomManager::getList<BookingClass> (*lFF_ptr);
00540 for (BookingClassList_T::const_iterator itBC =
00541      lBookingClassList.begin();
00542      itBC != lBookingClassList.end(); ++itBC) {
00543     const BookingClass* lBC_ptr = *itBC;
00544     assert (lBC_ptr != NULL);
00545
00546 #if BOOST_VERSION >= 104100
00547 // Put family code in property tree
00548 const FamilyCode_T& lFamilyCode = lFF_ptr->getFamilyCode();
00549 ioSCTree.put ("family_code", lFamilyCode);
00550
00551 // Export the cabin tree to add sub-classes details
00552 jsonExportBookingClass (ioPropertyTree, ioSCTree, *lBC_ptr);
00553 #endif // BOOST_VERSION >= 104100
00554 }
00555 } else {
00556
00557 // The fare family code is a fake one ('NoFF'), and therefore

```

```

00562         // does not vary
00563         const FamilyCode_T lDefaultFamilyCode ("NoFF");
00564
00565         // Browse the booking-classes, directly from the segment-cabin object
00566         const BookingClassList_T& lBookingClassList =
00567             BomManager::getList<BookingClass> (iSegmentCabin);
00568         for (BookingClassList_T::const_iterator itBC =
00569             lBookingClassList.begin();
00570             itBC != lBookingClassList.end(); ++itBC) {
00571             const BookingClass* lBC_ptr = *itBC;
00572             assert (lBC_ptr != NULL);
00573
00574 #if BOOST_VERSION >= 104100
00575
00576             // Put family code in property tree
00577             ioSCTree.put ("family_code", lDefaultFamilyCode);
00578
00579             // Export the cabin tree to add sub-classes details
00580             jsonExportBookingClass (ioPropertyTree, ioSCTree, *lBC_ptr);
00581
00582 #endif // BOOST_VERSION >= 104100
00583         }
00584     }
00585 }
00586
00587 // //////////////////////////////////////
00588 void BomJSONExport::jsonExportBookingClass (bpt::ptree& ioPropertyTree,
00589                                             bpt::ptree& ioSCTree,
00590                                             const BookingClass& iBookingClass) {
00591
00592 #if BOOST_VERSION >= 104100
00593
00594     // Put sub class in property tree
00595     ioSCTree.put ("class_code", iBookingClass.toString());
00596     // Put authorization level in property tree
00597     std::ostringstream oAUBlStr;
00598     oAUBlStr << iBookingClass.getAuthorizationLevel();
00599     //<< " (" << iBookingClass.getCumulatedBookingLimit()
00600     //<< ") ";
00601     ioSCTree.put ("au", oAUBlStr.str());
00602     // Put negotiated space in property tree
00603     const NbOfSeats_T& lNegotiatedSpace =
00604         iBookingClass.getNegotiatedSpace();
00605     ioSCTree.put ("nego", lNegotiatedSpace);
00606     // Put no show percentage in property tree
00607     const OverbookingRate_T& lNoShowPercentage =
00608         iBookingClass.getNoShowPercentage();
00609     ioSCTree.put ("ns%", lNoShowPercentage);
00610     // Put cancellation percentage in property tree
00611     const OverbookingRate_T& lCancellationPercentage =
00612         iBookingClass.getCancellationPercentage();
00613     ioSCTree.put ("ob%", lCancellationPercentage);
00614     // Put sub nb of bookings in property tree
00615     const NbOfBookings_T lNbOfBookings =
00616         iBookingClass.getNbOfBookings();
00617     ioSCTree.put ("bkgs", lNbOfBookings);
00618     // Put nb of group bookings in property tree
00619     const NbOfBookings_T& lNbOfGroupBookings =
00620         iBookingClass.getNbOfGroupBookings();
00621     ioSCTree.put ("grp_bks (pdg)", lNbOfGroupBookings);
00622     // Put nb of staff bookings in property tree
00623     const NbOfBookings_T& lNbOfStaffBookings =
00624         iBookingClass.getNbOfStaffBookings();
00625     ioSCTree.put ("stf_bkgs", lNbOfStaffBookings);
00626     // Put nb of WL bookings in property tree
00627     const NbOfBookings_T& lNbOfWLBookings =
00628         iBookingClass.getNbOfWLBookings();
00629     ioSCTree.put ("wl_bkgs", lNbOfWLBookings);
00630     // Put ETB in property tree
00631     const NbOfBookings_T& lETB = iBookingClass.getETB();
00632     ioSCTree.put ("etb", lETB);
00633     // Put net class availability in property tree
00634     const Availability_T& lNetClassAvailability =
00635         iBookingClass.getNetClassAvailability();
00636     ioSCTree.put ("class_avl", lNetClassAvailability);
00637     // Put segment availability in property tree
00638     const Availability_T& lSegmentAvailability =
00639         iBookingClass.getSegmentAvailability();
00640     ioSCTree.put ("seg_avl", lSegmentAvailability);
00641     // Put net revenue availability in property tree
00642     const Availability_T& lNetRevenueAvailability =
00643         iBookingClass.getNetRevenueAvailability();
00644     ioSCTree.put ("rev_avl", lNetRevenueAvailability);
00645
00646     // Add the sub-classe (containing cabin and fare-families information)
00647     // to the global tree
00648     ioPropertyTree.push_back(std::make_pair("", ioSCTree));
00649
00650 #endif
00651 }

```



```

00654
00655 #endif // BOOST_VERSION >= 104100
00656 }
00657
00658 ///////////////////////////////////////////////////////////////////
00659 void BomJSONExport::
00660 jsonExportBookingRequestObject (std::ostream& oStream,
00661                                 const EventStruct& iEventStruct) {
00662
00663     // Get the current event type: it should be booking request
00664     const EventType::EN_EventType& lEventType =
00665         iEventStruct.getEventType();
00666     assert (lEventType == EventType::BKG_REQ);
00667
00668     // Get the booking request (the current event type is booking request)
00669     const BookingRequestStruct& lBookingRequest =
00670         iEventStruct.getBookingRequest();
00671
00672 #if BOOST_VERSION >= 104100
00673
00674     // Create an empty property tree object for the current booking request
00675     bpt::ptree ptBookingRequest;
00676
00677     // Put request date time in property tree
00678     const DateTime_T& lRequestDateTime =
00679         lBookingRequest.getRequestDateTime();
00680     ptBookingRequest.put ("time_stamp", lRequestDateTime);
00681     // Put event type in property tree
00682     ptBookingRequest.put ("event_type", EventType::getLabel(lEventType));
00683     // Put origin in property tree
00684     const AirportCode_T& lOrigin = lBookingRequest.getOrigin();
00685     ptBookingRequest.put ("org", lOrigin);
00686     // Put destination in property tree
00687     const AirportCode_T& lDestination = lBookingRequest.
getDestination();
00688     ptBookingRequest.put ("des", lDestination);
00689     // Put preferred cabin in property tree
00690     const CabinCode_T& lCabinCode = lBookingRequest.getPreferredCabin();
00691     ptBookingRequest.put ("cab", lCabinCode);
00692     // Put party size in property tree
00693     const NbOfSeats_T& lNbOfSeats = lBookingRequest.getPartySize();
00694     ptBookingRequest.put ("pax", lNbOfSeats);
00695     // Put point-of-sale in property tree
00696     const AirportCode_T& lPOS = lBookingRequest.getPOS();
00697     ptBookingRequest.put ("pos", lPOS);
00698     // Put channel in property tree
00699     const ChannelLabel_T& lChannelLabel =
00700         lBookingRequest.getBookingChannel();
00701     ptBookingRequest.put ("cha", lChannelLabel);
00702     // Put WTP in property tree
00703     const WTP_T& lWTP = lBookingRequest.getWTP();
00704     ptBookingRequest.put ("wtp", lWTP);
00705     // Put request date in property tree
00706     const Date_T& lRequestDate =
00707         lRequestDateTime.boost::posix_time::ptime::date();
00708     ptBookingRequest.put ("bkg_date", lRequestDate);
00709     // Put departure date in property tree
00710     const Date_T& lPreferredDepartureDate =
00711         lBookingRequest.getPreferredDepartureDate();
00712     ptBookingRequest.put ("dep_date", lPreferredDepartureDate);
00713     // Put advance purchase in property tree
00714     assert (lPreferredDepartureDate >= lRequestDate);
00715     const DateOffset_T& lAdvancePurchase =
00716         lPreferredDepartureDate - lRequestDate;
00717     ptBookingRequest.put ("adv_purchase", lAdvancePurchase);
00718     // Put stay duration in property tree
00719     const DayDuration_T& lStayDuration =
00720         lBookingRequest.getStayDuration();
00721     ptBookingRequest.put ("stay_duration", lStayDuration);
00722     // Put return date in property tree
00723     const DateOffset_T lDayDuration (lStayDuration);
00724     const Date_T& lReturnDate =
00725         lPreferredDepartureDate + lDayDuration;
00726     ptBookingRequest.put ("return_date", lReturnDate);
00727     // Put cancellation date in property tree
00728     // TODO: cancellation date
00729     ptBookingRequest.put ("cancel_date", "xxxx-xx-xx");
00730     // Put preferred departure time in property tree
00731     const Duration_T& lPreferredDepartureTime =
00732         lBookingRequest.getPreferredDepartureTime();
00733     ptBookingRequest.put ("dep_time", lPreferredDepartureTime);
00734     // Put preferred return time in property tree
00735     // TODO: preferred return time
00736     ptBookingRequest.put ("return_time", "xxPM");
00737     // Put preferred carriers in property tree
00738     // TODO: preferred carriers
00739     ptBookingRequest.put ("pref_carriers", "XX");

```

```

00740
00741     // Write the property tree into the JSON stream.
00742     write_json (oStream, ptBookingRequest);
00743
00744 #endif // BOOST_VERSION >= 104100
00745 }
00746
00747 // //////////////////////////////////////
00748 void BomJSONExport::
00749 jsonExportBreakPointObject (std::ostream& oStream,
00750                             const EventStruct& iEventStruct) {
00751
00752     // Get the current event type: it should be break point
00753     const EventType::EN_EventType& lEventType =
00754         iEventStruct.getEventType();
00755     assert (lEventType == EventType::BRK_PT);
00756
00757     // Get the break point (the current event type is break point)
00758     const BreakPointStruct& lBreakPoint =
00759         iEventStruct.getBreakPoint();
00760
00761 #if BOOST_VERSION >= 104100
00762
00763     // Create an empty property tree object for the current break point
00764     bpt::ptree ptBreakPoint;
00765
00766     // Put break point date time in property tree
00767     const DateTime_T& lRequestDateTime =
00768         lBreakPoint.getBreakPointTime();
00769     ptBreakPoint.put ("time_stamp", lRequestDateTime);
00770     // Put event type in property tree
00771     ptBreakPoint.put ("event_type", EventType::getLabel(lEventType));
00772
00773     // Write the property tree into the JSON stream.
00774     write_json (oStream, ptBreakPoint);
00775
00776 #endif // BOOST_VERSION >= 104100
00777 }
00778
00779
00780 }

```

### 33.199 stdair/bom/BomJSONExport.hpp File Reference

```

#include <iosfwd>
#include <stdair/bom/TravelSolutionTypes.hpp>

```

#### Classes

- class [stdair::BomJSONExport](#)  
*Utility class to export StdAir objects in a JSON format.*

#### Namespaces

- [bpt](#)
- [stdair](#)  
*Handle on the StdAir library context.*

### 33.200 BomJSONExport.hpp

```

00001 #ifndef __STDAIR_BOM_BOMJSONEXPORT_HPP
00002 #define __STDAIR_BOM_BOMJSONEXPORT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 // Boost Property Tree
00010 #if BOOST_VERSION >= 104100

```

```

00011 #include <boost/property_tree/ptree.hpp>
00012 #include <boost/property_tree/json_parser.hpp>
00013 #endif // BOOST_VERSION >= 104100
00014 // StdAir
00015 #include <stdair/bom/TravelSolutionTypes.hpp>
00016
00017 #if BOOST_VERSION >= 104100
00018     namespace bpt = boost::property_tree;
00019 #else // BOOST_VERSION >= 104100
00020     namespace bpt {
00021         typedef char ptree;
00022     }
00023 #endif // BOOST_VERSION >= 104100
00024
00025 namespace stdair {
00026
00027     class BomRoot;
00028     class Inventory;
00029     class FlightDate;
00030     class LegDate;
00031     class LegCabin;
00032     class SegmentDate;
00033     class SegmentCabin;
00034     class BookingClass;
00035     class EventStruct;
00036
00037     class BomJSONExport {
00038     public:
00039         // ////////////////////////////////// Export support methods //////////////////////////////////
00040
00041         static void jsonExportFlightDateList (std::ostream&, const
00042         BomRoot&,
00043                                     const AirlineCode_T& iAirlineCode = "all",
00044                                     const FlightNumber_T& iFlightNumber = 0);
00045
00046         static void jsonExportFlightDateObjects (std::ostream&, const
00047         FlightDate&);
00048
00049         static void jsonExportBookingRequestObject (std::ostream&,
00050                                     const EventStruct&);
00051
00052         static void jsonExportBreakPointObject (std::ostream&,
00053                                     const EventStruct&);
00054
00055     private:
00056
00057         static void jsonExportFlightDate (bpt::ptree&,
00058                                     const Inventory&,
00059                                     const FlightNumber_T&);
00060
00061         static void jsonExportLegDate (bpt::ptree&, const FlightDate&);
00062
00063         static void jsonExportLegCabin (bpt::ptree&, const LegDate&);
00064
00065         static void jsonExportBucket (bpt::ptree&, const LegCabin&);
00066
00067         static void jsonExportSegmentDate (bpt::ptree&, const FlightDate&);
00068
00069         static void jsonExportSegmentCabin (bpt::ptree&, const SegmentDate&);
00070
00071         static void jsonExportFareFamily (bpt::ptree&, bpt::ptree&,
00072                                     const SegmentCabin&);
00073
00074         static void jsonExportBookingClass (bpt::ptree&, bpt::ptree&,
00075                                     const BookingClass&);
00076
00077     };
00078
00079 }
00080 #endif // __STDAIR_BOM_BOMJSONEXPORT_HPP

```

### 33.201 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>

```



```

00079         hasCommandBeenSuccessfullyRetrieved = false;
00080         return hasCommandBeenSuccessfullyRetrieved;
00081     }
00082
00083     assert (oTokenList.size() >= 2);
00084     // Retrieved the command string into the token list
00085     const std::string lCommandStr = oTokenList.at(1);
00086     const JSonCommand lJSoCommand (lCommandStr);
00087     ioEnumJSoCommand = lJSoCommand.getCommand();
00088
00089     } catch (stdair::CodeConversionException& ccException) {
00090         hasCommandBeenSuccessfullyRetrieved = false;
00091     }
00092
00093     return hasCommandBeenSuccessfullyRetrieved;
00094 }
00095
00096
00097 // //////////////////////////////////////
00098 bool BomJSONImport::jsonImportInventoryKey (const
JSONString& iBomJSONStr,
00099                                             AirlineCode_T& ioAirlineCode) {
00100     bool hasKeyBeenSuccessfullyRetrieved = true;
00101
00102     #if BOOST_VERSION >= 104100
00103         // Create an empty property tree object
00104         bpt::ptree pt;
00105
00106         try {
00107
00108             // Load the JSON formatted string into the property tree.
00109             // If reading fails (cannot open stream, parse error), an
00110             // exception is thrown.
00111             std::istream iStr (iBomJSONStr.getString());
00112             read_json (iStr, pt);
00113
00114             // Build the right path to obtain the airline code value.
00115             bpt::ptree::const_iterator itBegin = pt.begin();
00116             const std::string lCommandName = itBegin->first;
00117             std::ostream lPath;
00118             lPath << lCommandName << ".airline_code";
00119
00120             // Get the airline_code.
00121             // If the path key is not found, an exception is thrown.
00122             ioAirlineCode = pt.get<AirlineCode_T> (lPath.str());
00123
00124         } catch (bpt::ptree_error& bptException) {
00125             hasKeyBeenSuccessfullyRetrieved = false;
00126         }
00127
00128     #endif // BOOST_VERSION >= 104100
00129     return hasKeyBeenSuccessfullyRetrieved;
00130 }
00131
00132 // //////////////////////////////////////
00133 bool BomJSONImport::jsonImportFlightDate (const
JSONString& iBomJSONStr,
00134                                             Date_T& ioDepartureDate) {
00135     bool hasKeyBeenSuccessfullyRetrieved = true;
00136
00137     #if BOOST_VERSION >= 104100
00138         // Create an empty property tree object
00139         bpt::ptree pt;
00140
00141         try {
00142
00143             // Load the JSON formatted string into the property tree.
00144             // If reading fails (cannot open stream, parse error), an
00145             // exception is thrown.
00146             std::istream iStr (iBomJSONStr.getString());
00147             read_json (iStr, pt);
00148
00149             // Build the right path to obtain the departure date value.
00150             const std::string& lDepartureDateStr =
pt.get<std::string> ("flight_date.departure_date");
00151
00152             // Get the departure_date.
00153             // If the path key is not found, an exception is thrown.
00154             ioDepartureDate =
boost::gregorian::from_simple_string (lDepartureDateStr);
00155
00156         } catch (bpt::ptree_error& bptException) {
00157             hasKeyBeenSuccessfullyRetrieved = false;
00158         }
00159     #endif // BOOST_VERSION >= 104100
00160
00161     return hasKeyBeenSuccessfullyRetrieved;
00162 }
00163

```

```

00164     }
00165
00166     // //////////////////////////////////////
00167     bool BomJSONImport::jsonImportFlightNumber (const
JSONString& iBomJSONStr,
00168
00169
00170
00171
00172
00173
00174
00175
00176
00177
00178
00179
00180
00181
00182
00183
00184
00185
00186
00187
00188
00189
00190
00191
00192
00193
00194
00195
00196
00197
00198
00199
00200
00201
00202
00203
00204
00205
00206
00207
00208
00209
00210
00211
00212
00213
00214
00215
00216
00217
00218
00219
00220
00221
00222
00223
00224
00225
00226
00227
00228
00229
00230
00231
00232
00233
00234
00235
00236
00237
00238
00239
00240
00241
00242
00243
00244
00245
00246
00247
00248
FlightNumber_T& ioFlightNumber) {
    bool hasKeyBeenSuccessfullyRetrieved = true;
    #if BOOST_VERSION >= 104100
    // Create an empty property tree object
    bpt::ptree pt;
    try {
        // Load the JSON formatted string into the property tree.
        // If reading fails (cannot open stream, parse error), an
        // exception is thrown.
        std::istream iStr (iBomJSONStr.getString());
        read_json (iStr, pt);
        // Build the right path to obtain the flight number value.
        bpt::ptree::const_iterator itBegin = pt.begin();
        const std::string lCommandName = itBegin->first;
        std::ostringstream lPath;
        lPath << lCommandName << ".flight_number";
        // Get the flight_number.
        // If the path key is not found, an exception is thrown.
        ioFlightNumber = pt.get<FlightNumber_T> (lPath.str());
    } catch (bpt::ptree_error& bptException) {
        hasKeyBeenSuccessfullyRetrieved = false;
    }
    #endif // BOOST_VERSION >= 104100
    return hasKeyBeenSuccessfullyRetrieved;
}

// //////////////////////////////////////
00203 bool BomJSONImport::jsonImportBreakPoints (const
JSONString& iBomJSONStr,
00204
00205
00206
00207
00208
00209
00210
00211
00212
00213
00214
00215
00216
00217
00218
00219
00220
00221
00222
00223
00224
00225
00226
00227
00228
00229
00230
00231
00232
00233
00234
00235
00236
00237
00238
00239
00240
00241
00242
00243
00244
00245
00246
00247
00248
BreakPointList_T& oBreakPointList) {
    bool hasKeyBeenSuccessfullyRetrieved = true;
    #if BOOST_VERSION >= 104100
    // Create an empty property tree object
    bpt::ptree pt;
    try {
        // Load the JSON formatted string into the property tree.
        // If reading fails (cannot open stream, parse error), an
        // exception is thrown.
        std::istream iStr (iBomJSONStr.getString());
        read_json (iStr, pt);
        // Access the break point list tree
        bpt::ptree::const_iterator itBegin = pt.begin();
        bpt::ptree ptListOfBP = itBegin->second;
        // Browse the break point list
        for (bpt::ptree::const_iterator itBP = ptListOfBP.begin();
            itBP != ptListOfBP.end(); ++itBP) {
            // Access the current break point tree
            bpt::ptree ptBP = itBP->second;
            // Access the date of the break point
            bpt::ptree::const_iterator itDate = ptBP.begin();
            bpt::ptree ptDate = itDate->second;
            // Recover the string containing the date
            std::string lDateString = ptDate.data();
            if (lDateString.empty() == false) {
                // Construct the break point using the recovered string
                const Date_T lDate =
                    boost::gregorian::from_simple_string (lDateString);
                BreakPointStruct lBreakPoint (lDate);
                // Add the break point to the list
                oBreakPointList.push_back (lBreakPoint);
            }
        }
    } catch (bpt::ptree_error& bptException) {
        hasKeyBeenSuccessfullyRetrieved = false;
    } catch (boost::bad_lexical_cast& eCast) {
        hasKeyBeenSuccessfullyRetrieved = false;
    }
    #endif // BOOST_VERSION >= 104100
}

```

```

00249     return hasKeyBeenSuccessfullyRetrieved;
00250 }
00251
00252 // //////////////////////////////////////
00253 bool BomJSONImport::jsonImportEventType (const
JSONString& iBomJSONStr,
                                EventType::EN_EventType& ioEventType) {
00254
00255     bool hasKeyBeenSuccessfullyRetrieved = true;
00256
00257     #if BOOST_VERSION >= 104100
00258         // Create an empty property tree object
00259         bpt::ptree pt;
00260
00261         try {
00262
00263             // Load the JSON formatted string into the property tree.
00264             // If reading fails (cannot open stream, parse error), an
00265             // exception is thrown.
00266             std::istringstream iStr (iBomJSONStr.getString());
00267             read_json (iStr, pt);
00268
00269             // Build the right path to obtain the event type value.
00270             bpt::ptree::const_iterator itBegin = pt.begin();
00271             const std::string lEventTypeName = itBegin->first;
00272             std::ostringstream lPath;
00273             lPath << lEventTypeName << ".event_type";
00274
00275             // Get the event type string
00276             // If the path key is not found, an exception bpt::ptree_error is thrown.
00277             const std::string lEventTypeNameStr = pt.get<std::string> (lPath.str());
00278             // Build the event type using the string.
00279             // If the input string is incorrect, an exception
00280             // stdair::CodeConversionException is thrown.
00281             const EventType lEventType (lEventTypeNameStr);
00282             ioEventType = lEventType.getType();
00283
00284             } catch (bpt::ptree_error& bptException) {
00285                 hasKeyBeenSuccessfullyRetrieved = false;
00286             } catch (stdair::CodeConversionException& cceException) {
00287                 hasKeyBeenSuccessfullyRetrieved = false;
00288             }
00289         }
00290     #endif // BOOST_VERSION >= 104100
00291
00292     return hasKeyBeenSuccessfullyRetrieved;
00293 }
00294
00295 // //////////////////////////////////////
00296 bool BomJSONImport::jsonImportConfig (const
JSONString& iBomJSONStr,
                                ConfigHolderStruct& iConfigHolderStruct) {
00297
00298     bool hasConfigBeenSuccessfullyRetrieved = true;
00299
00300     #if BOOST_VERSION >= 104100
00301         // Create an empty property tree object
00302         bpt::ptree pt;
00303
00304         try {
00305
00306             // Load the JSON formatted string into the property tree.
00307             // If reading fails (cannot open stream, parse error), an
00308             // exception is thrown.
00309             std::istringstream iStr (iBomJSONStr.getString());
00310             read_json (iStr, pt);
00311
00312             // Load the pt in the configuration holder
00313             iConfigHolderStruct.add (pt);
00314             } catch (bpt::ptree_error& bptException) {
00315                 hasConfigBeenSuccessfullyRetrieved = false;
00316             }
00317         }
00318     #endif // BOOST_VERSION >= 104100
00319
00320     return hasConfigBeenSuccessfullyRetrieved;
00321 }
00322
00323 }

```

### 33.203 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>
```

## Classes

- class [stdair::BomJSONImport](#)  
*Utility class to import StdAir objects in a JSON format.*

## Namespaces

- [stdair](#)  
*Handle on the StdAir library context.*

## 33.204 BomJSONImport.hpp

```
00001 #ifndef __STDAIR_BOM_BOMJSONIMPORT_HPP
00002 #define __STDAIR_BOM_BOMJSONIMPORT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/stdair_basic_types.hpp>
00011 #include <stdair/stdair_date_time_types.hpp>
00012 #include <stdair/basic/JsonCommand.hpp>
00013 #include <stdair/basic/EventType.hpp>
00014 #include <stdair/bom/BreakPointStruct.hpp>
00015
00016
00017 namespace stdair {
00018
00020     class JSONString;
00021     class ConfigHolderStruct;
00022
00026     class BomJSONImport {
00027     public:
00028         // ////////////////////////////////// Import support methods //////////////////////////////////
00036         static bool jsonImportCommand (const JSONString&,
00037                                         JSONCommand::EN_JSonCommand&);
00045         static bool jsonImportInventoryKey (const JSONString&,
00046                                             AirlineCode_T&);
00047
00055         static bool jsonImportFlightDate (const JSONString&,
00056                                           Date_T&);
00057
00065         static bool jsonImportFlightNumber (const JSONString&,
00066                                             FlightNumber_T&);
00067
00075         static bool jsonImportBreakPoints (const JSONString&,
00076                                           BreakPointList_T&);
00077
00085         static bool jsonImportEventType (const JSONString&,
00086                                           EventType::EN_EventType&);
00087
00096         static bool jsonImportConfig (const JSONString&,
00097                                       ConfigHolderStruct&);
00098     };
00099
00100 }
00101 #endif // __STDAIR_BOM_BOMJSONIMPORT_HPP
```

## 33.205 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

- [stdair](#)

*Handle on the StdAir library context.*

## Typedefs

- `typedef boost::tokenizer< boost::char_separator< char > > stdair::Tokeniser\_T`

## 33.206 BomKeyManager.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost
00008 #include <boost/tokenizer.hpp>
00009 #include <boost/lexical_cast.hpp>
00010 #include <boost/date_time/gregorian/parsers.hpp>
00011 // StdAir
00012 #include <stdair/stdair_exceptions.hpp>
00013 #include <stdair/basic/BasConst_BomDisplay.hpp>
00014 #include <stdair/bom/InventoryKey.hpp>
00015 #include <stdair/bom/FlightDateKey.hpp>
00016 #include <stdair/bom/SegmentDateKey.hpp>
00017 #include <stdair/bom/LegDateKey.hpp>
00018 #include <stdair/bom/ParsedKey.hpp>
00019 #include <stdair/bom/BomKeyManager.hpp>
00020 #include <stdair/service/Logger.hpp>
00021
00022 namespace stdair {
00023
00024 // ////////////////////////////////// Tokenising support //////////////////////////////////
00028 typedef boost::tokenizer<boost::char_separator<char> > Tokeniser_T;
00029
00030 // //////////////////////////////////
00031 ParsedKey BomKeyManager::extractKeys (const std::string& iFullKeyStr)
00032 {
00033     ParsedKey oParsedKey;
00034     oParsedKey._fullKey = iFullKeyStr;
00035
00036     // Token-ise the full key string
00037     Tokeniser_T lTokens (iFullKeyStr, DEFAULT_KEY_TOKEN_DELIMITER);
00038     Tokeniser_T::iterator itToken = lTokens.begin();
00039
00040     // Airline code
00041     if (itToken != lTokens.end()) {
00042         oParsedKey._airlineCode = *itToken;
00043
00044         // Flight number
00045         ++itToken;
00046         if (itToken != lTokens.end()) {
00047             oParsedKey._flightNumber = *itToken;
00048         }
00049     }
00050 }

```

```

00048         // Departure date
00049         ++itToken;
00050         if (itToken != lTokens.end()) {
00051             oParsedKey._departureDate = *itToken;
00052
00053             // Origin
00054             ++itToken;
00055             if (itToken != lTokens.end()) {
00056                 oParsedKey._boardingPoint = *itToken;
00057
00058                 // Destination
00059                 ++itToken;
00060                 if (itToken != lTokens.end()) {
00061                     oParsedKey._offPoint = *itToken;
00062
00063                     // Boarding time
00064                     ++itToken;
00065                     if (itToken != lTokens.end()) {
00066                         oParsedKey._boardingTime = *itToken;
00067                     }
00068                 }
00069             }
00070         }
00071     }
00072 }
00073
00074     return oParsedKey;
00075 }
00076
00077 // //////////////////////////////////////
00078 InventoryKey BomKeyManager::
00079 extractInventoryKey (const std::string& iFullKeyStr) {
00080     ParsedKey lParsedKey = extractKeys (iFullKeyStr);
00081
00082     return lParsedKey.getInventoryKey();
00083 }
00084
00085 // //////////////////////////////////////
00086 FlightDateKey BomKeyManager::
00087 extractFlightDateKey (const std::string& iFullKeyStr) {
00088     ParsedKey lParsedKey = extractKeys (iFullKeyStr);
00089
00090     return lParsedKey.getFlightDateKey();
00091 }
00092
00093 // //////////////////////////////////////
00094 SegmentDateKey BomKeyManager::
00095 extractSegmentDateKey (const std::string& iFullKeyStr) {
00096     ParsedKey lParsedKey = extractKeys (iFullKeyStr);
00097
00098     return lParsedKey.getSegmentKey();
00099 }
00100
00101 // //////////////////////////////////////
00102 LegDateKey BomKeyManager::
00103 extractLegDateKey (const std::string& iFullKeyStr) {
00104     ParsedKey lParsedKey = extractKeys (iFullKeyStr);
00105
00106     return lParsedKey.getLegKey();
00107 }
00108 }

```

## 33.207 stdair/bom/BomKeyManager.hpp File Reference

```

#include <iosfwd>
#include <stdair/stdair_basic_types.hpp>

```

### Classes

- class [stdair::BomKeyManager](#)  
*Utility class to extract key structures from strings.*

### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.208 BomKeyManager.hpp

```

00001 #ifndef __STDAIR_BOM_BOMKEYMANAGER_HPP
00002 #define __STDAIR_BOM_BOMKEYMANAGER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 // StdAir
00010 #include <stdair/stdair_basic_types.hpp>
00011
00012 namespace stdair {
00013
00014     struct BomRootKey;
00015     struct InventoryKey;
00016     struct FlightDateKey;
00017     struct LegDateKey;
00018     struct SegmentDateKey;
00019     struct LegCabinKey;
00020     struct SegmentCabinKey;
00021     struct FareFamilyKey;
00022     struct BookingClassKey;
00023     struct ParsedKey;
00024
00025     class BomKeyManager {
00026     public:
00027         // ////////////////////////////////// Key management support methods //////////////////////////////////
00028         static ParsedKey extractKeys (const std::string& iFullKeyStr);
00029
00030         static InventoryKey extractInventoryKey (const std::string& iFullKeyStr)
00031         ;
00032
00033         static FlightDateKey extractFlightDateKey (const std::string&
00034 iFullKeyStr);
00035
00036         static SegmentDateKey extractSegmentDateKey (const std::string&
00037 iFullKeyStr);
00038
00039         static LegDateKey extractLegDateKey (const std::string& iFullKeyStr);
00040
00041     };
00042 }
00043 #endif // __STDAIR_BOM_BOMKEYMANAGER_HPP

```

### 33.209 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>

```

#### Classes

- class [stdair::BomManager](#)

*Utility class for StdAir-based objects.*

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.210 BomManager.hpp

```

00001 #ifndef __STDAIR_BOM_BOMMANAGER_HPP
00002 #define __STDAIR_BOM_BOMMANAGER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 #include <list>
00011 #include <map>
00012 // Boost
00013 #include <boost/static_assert.hpp>
00014 #include <boost/type_traits/is_same.hpp>
00015 // StdAir
00016 #include <stdair/stdair_exceptions.hpp>
00017 #include <stdair/bom/BomAbstract.hpp>
00018 #include <stdair/bom/BomHolder.hpp>
00019 #include <stdair/service/Logger.hpp>
00020 // Stdair BOM Objects
00021 #include <stdair/bom/SegmentDate.hpp>
00022 #include <stdair/bom/Inventory.hpp>
00023 #include <stdair/bom/AirlineFeature.hpp>
00024
00025 namespace stdair {
00026
00034     class BomManager {
00035         friend class FacBomManager;
00036
00037     public:
00041         template <typename OBJECT2, typename OBJECT1>
00042         static const typename BomHolder<OBJECT2>::BomList_T&
00043         getList (const OBJECT1&);
00047         template <typename OBJECT2, typename OBJECT1>
00048         static const typename BomHolder<OBJECT2>::BomMap_T&
00049         getMap (const OBJECT1&);
00053         template <typename OBJECT2, typename OBJECT1>
00054         static bool hasList (const OBJECT1&);
00055
00059         template <typename OBJECT2, typename OBJECT1>
00060         static bool hasMap (const OBJECT1&);
00061
00067         template <typename PARENT, typename CHILD>
00068         static PARENT* getParentPtr (const CHILD&);
00069
00073         template <typename PARENT, typename CHILD>
00074         static PARENT& getParent (const CHILD&);
00075
00081         template <typename OBJECT2, typename OBJECT1>
00082         static OBJECT2* getObjectPtr (const OBJECT1&, const MapKey_T&);
00083
00087         template <typename OBJECT2, typename OBJECT1>
00088         static OBJECT2& getObject (const OBJECT1&, const MapKey_T&);
00089
00090     private:
00096         template <typename OBJECT2, typename OBJECT1>
00097         static const BomHolder<OBJECT2>& getBomHolder (const OBJECT1&);
00098     };
00099
00100 // //////////////////////////////////////
00101 // Private method.
00102 template <typename OBJECT2, typename OBJECT1>
00103 const BomHolder<OBJECT2>& BomManager::getBomHolder (const OBJECT1& iObject1) {
00104     //
00105     // Compile time assertion: this function must never be called with the
00106     // following list of couple types:
00107     // <SegmentDate, SegmentDate>
00108     // <AirlineFeature, Inventory>
00109     //
00111     BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, SegmentDate>::value == false
00112         || boost::is_same<OBJECT2, SegmentDate>::value == false));
00113     BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, Inventory>::value == false

```

```

00114         || boost::is_same<OBJECT2, AirlineFeature>::value == false));
00115
00116     const HolderMap_T& lHolderMap = iObject1.getHolderMap();
00117
00118     HolderMap_T::const_iterator itHolder = lHolderMap.find (&typeid (OBJECT2));
00119
00120     if (itHolder == lHolderMap.end()) {
00121         const std::string lName (typeid (OBJECT2).name());
00122         throw NonInitialisedContainerException ("Cannot find the holder of
type "
00123         + lName + " within: "
00124         + iObject1.describeKey());
00125     }
00126
00127     const BomHolder<OBJECT2>* lBomHolder_ptr =
00128         static_cast<const BomHolder<OBJECT2>*> (itHolder->second);
00129     assert (lBomHolder_ptr != NULL);
00130
00131     return *lBomHolder_ptr;
00132 }
00133
00134 // //////////////////////////////////////
00135 // Public business method.
00136 // This method is specialized for the following couple types:
00137 // <SegmentDate, SegmentDate>
00138 template <typename OBJECT2, typename OBJECT1>
00139 const typename BomHolder<OBJECT2>::BomList_T& BomManager::
00140 getList (const OBJECT1& iObject1) {
00141     //
00142     // Compile time assertion: this function must never be called with the
00143     // following list of couple types:
00144     // <AirlineFeature, Inventory>
00145     //
00146     BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, Inventory>::value == false
|| boost::is_same<OBJECT2, AirlineFeature>::value == false));
00147
00148     const BomHolder<OBJECT2>& lBomHolder = getBomHolder<OBJECT2> (iObject1);
00149     return lBomHolder._bomList;
00150 }
00151
00152 // //////////////////////////////////////
00153 // Public business method.
00154 // Compile time assertion to check OBJECT1 and OBJECT2 types.
00155 template <typename OBJECT2, typename OBJECT1>
00156 const typename BomHolder<OBJECT2>::BomMap_T&
BomManager::
00159 getMap (const OBJECT1& iObject1) {
00160     //
00161     // Compile time assertion: this function must never be called with the
00162     // following list of couple types:
00163     // <SegmentDate, SegmentDate>
00164     // <AirlineFeature, Inventory>
00165     //
00166     BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, SegmentDate>::value == false
|| boost::is_same<OBJECT2, SegmentDate>::value == false));
00167     BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, Inventory>::value == false
|| boost::is_same<OBJECT2, AirlineFeature>::value == false));
00168
00169     const BomHolder<OBJECT2>& lBomHolder = getBomHolder<OBJECT2> (iObject1);
00170     return lBomHolder._bomMap;
00171 }
00172
00173 // //////////////////////////////////////
00174 // Public business method.
00175 // This method is specialized for the following couple types:
00176 // <SegmentDate, SegmentDate>
00177 template <typename OBJECT2, typename OBJECT1>
00178 bool BomManager::hasList (const OBJECT1& iObject1) {
00179     //
00180     const HolderMap_T& lHolderMap = iObject1.getHolderMap();
00181     HolderMap_T::const_iterator itHolder = lHolderMap.find (&typeid (OBJECT2));
00182
00183     if (itHolder == lHolderMap.end()) {
00184         return false;
00185     }
00186     const BomHolder<OBJECT2>* lBomHolder_ptr =
00187         static_cast<const BomHolder<OBJECT2>*> (itHolder->second);
00188     assert (lBomHolder_ptr != NULL);
00189     return !lBomHolder_ptr->_bomList.empty();
00190 }
00191
00192 // //////////////////////////////////////
00193 // Public business method.
00194 // This method is specialized for the following couple types:

```

```

00199 // <SegmentDate, SegmentDate>
00200 template <typename OBJECT2, typename OBJECT1>
00201 bool BomManager::hasMap (const OBJECT1& iObject1) {
00202
00203     const HolderMap_T& lHolderMap = iObject1.getHolderMap();
00204     HolderMap_T::const_iterator itHolder = lHolderMap.find (&typeid (OBJECT2));
00205
00206     if (itHolder == lHolderMap.end()) {
00207         return false;
00208     }
00209     const BomHolder<OBJECT2>* lBomHolder_ptr =
00210         static_cast<const BomHolder<OBJECT2>*> (itHolder->second);
00211     assert (lBomHolder_ptr != NULL);
00212
00213     return !lBomHolder_ptr->_bomMap.empty();
00214 }
00215
00216 // //////////////////////////////////////
00217 // Public business method valid for all PARENT and CHILD types.
00218 // (No compile time assertion to check PARENT and CHILD types.)
00219 template <typename PARENT, typename CHILD>
00220 PARENT* BomManager::getParentPtr (const CHILD& iChild) {
00221
00222     PARENT* const lParent_ptr = static_cast<PARENT* const> (iChild.getParent());
00223     return lParent_ptr;
00224 }
00225
00226 // //////////////////////////////////////
00227 // Public business method valid for all PARENT and CHILD types.
00228 // (No compile time assertion to check PARENT and CHILD types.)
00229 template <typename PARENT, typename CHILD>
00230 PARENT& BomManager::getParent (const CHILD& iChild) {
00231
00232     PARENT* const lParent_ptr = getParentPtr<PARENT> (iChild);
00233     assert (lParent_ptr != NULL);
00234     return *lParent_ptr;
00235 }
00236
00237 // //////////////////////////////////////
00238 // Public business method.
00239 // Compile time assertion to check OBJECT1 and OBJECT2 types.
00240 template <typename OBJECT2, typename OBJECT1>
00241 OBJECT2* BomManager::getObjectPtr (const OBJECT1& iObject1,
00242                                     const MapKey_T& iKey) {
00243
00244     //
00245     // Compile time assertion: this function must never be called with the
00246     // following list of couple types:
00247     // <SegmentDate, SegmentDate>
00248     //
00249     BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, SegmentDate>::value == false
00250                             || boost::is_same<OBJECT2, SegmentDate>::value == false));
00251
00252     OBJECT2* oBom_ptr = NULL;
00253
00254     const HolderMap_T& lHolderMap = iObject1.getHolderMap();
00255
00256     typename HolderMap_T::const_iterator itHolder =
00257         lHolderMap.find (&typeid (OBJECT2));
00258
00259     if (itHolder != lHolderMap.end()) {
00260
00261         BomHolder<OBJECT2>* const lBomHolder_ptr =
00262             static_cast<BomHolder<OBJECT2>*> const> (itHolder->second);
00263         assert (lBomHolder_ptr != NULL);
00264
00265         //
00266         typedef typename BomHolder<OBJECT2>::BomMap_T BomMap_T;
00267         BomMap_T& lBomMap = lBomHolder_ptr->_bomMap;
00268         typename BomMap_T::iterator itBom = lBomMap.find (iKey);
00269
00270         if (itBom != lBomMap.end()) {
00271             oBom_ptr = itBom->second;
00272             assert (oBom_ptr != NULL);
00273         }
00274     }
00275
00276     return oBom_ptr;
00277 }
00278
00279 // //////////////////////////////////////
00280 // Public business method.
00281 // Compile time assertion to check OBJECT1 and OBJECT2 types.
00282 template <typename OBJECT2, typename OBJECT1>
00283 OBJECT2& BomManager::getObject (const OBJECT1& iObject1,
00284                                 const MapKey_T& iKey) {
00285

```

```

00286 //
00287 // Compile time assertion: this function must never be called with the
00288 // following list of couple types:
00289 // <SegmentDate, SegmentDate>
00290 //
00291 BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, SegmentDate>::value == false
00292                      || boost::is_same<OBJECT2, SegmentDate>::value == false));
00293
00294 OBJECT2* oBom_ptr = NULL;
00295
00296 typedef std::map<const MapKey_T, OBJECT2*> BomMap_T;
00297 const BomMap_T& lBomMap = getMap<OBJECT2> (iObject1);
00298
00299 typename BomMap_T::const_iterator itBom = lBomMap.find (iKey);
00300
00301 if (itBom == lBomMap.end()) {
00302     const std::string lName (typeid (OBJECT2).name());
00303
00304     STDAIR_LOG_ERROR ("Cannot find the objet of type " << lName
00305                     << " with key " << iKey << " within: "
00306                     << iObject1.describeKey());
00307     assert (false);
00308 }
00309
00310 oBom_ptr = itBom->second;
00311 assert (oBom_ptr != NULL);
00312
00313 return *oBom_ptr;
00314 }
00315
00316 // //////////////////////////////////////
00317 //
00318 // Specialization of the template methods above for a segment
00319 // date and its corresponding marketing segment dates.
00320 //
00321 // //////////////////////////////////////
00322
00323 // Specialization of the template method hasList above for the types
00324 // <SegmentDate, SegmentDate>.
00325 // Return a boolean saying if the marketing segment date list is empty
00326 // or not.
00327 template<>
00328 inline bool BomManager::hasList<SegmentDate,SegmentDate>
00329 (const SegmentDate& ioSegmentDate) {
00330
00331     const SegmentDateList_T& lMarketingSegmentDateList =
00332         ioSegmentDate.getMarketingSegmentDateList ();
00333     const bool isMarketingSegmentDateListEmpty =
00334         lMarketingSegmentDateList.empty();
00335     const bool hasMarketingSegmentDateList =
00336         !isMarketingSegmentDateListEmpty;
00337     return hasMarketingSegmentDateList;
00338 }
00339
00340 // Specialization of the template method hasList above for the types
00341 // <SegmentDate, SegmentDate>.
00342 // Return the marketing segment date list.
00343 template<>
00344 inline const BomHolder<SegmentDate>::BomList_T&
00345 BomManager::getList<SegmentDate,SegmentDate> (const SegmentDate& ioSegmentDate) {
00346
00347     const SegmentDateList_T& lMarketingSegmentDateList =
00348         ioSegmentDate.getMarketingSegmentDateList ();
00349     return lMarketingSegmentDateList;
00350 }
00351
00352 // Specialization of the template method hasMap above for the types
00353 // <SegmentDate, SegmentDate>.
00354 // A segment date does not have a Segment Date Map but it can have a
00355 // Segment Date list (containing its marketing segment dates).
00356 template<>
00357 inline bool BomManager::hasMap<SegmentDate,SegmentDate>
00358 (const SegmentDate& ioSegmentDate) {
00359
00360     const bool hasMap = false;
00361     return hasMap;
00362 }
00363
00364 // //////////////////////////////////////
00365 //
00366 // Specialization of the template methods above for an inventory
00367 // and its airline features.
00368 //
00369 // //////////////////////////////////////
00370
00371 // Specialization of the template method hasList above for the types
00372 // <AirlineFeature,Inventory>.

```

```

00373     template<>
00374     inline bool BomManager::hasList<AirlineFeature,Inventory>
00375     (const Inventory& ioInventory) {
00376
00377         const bool hasList = false;
00378         return hasList;
00379     }
00380
00381     // Specialization of the template method hasMap above for the types
00382     // <AirlineFeature,Inventory>.
00383     template<>
00384     inline bool BomManager::hasMap<AirlineFeature,Inventory>
00385     (const Inventory& ioInventory) {
00386
00387         const bool hasMap = false;
00388         return hasMap;
00389     }
00390
00391     // Specialization of the template method getObjectPtr above for the types
00392     // <AirlineFeature,Inventory>.
00393     template<>
00394     inline AirlineFeature* BomManager::getObjectPtr<AirlineFeature,Inventory>
00395     (const Inventory& iInventory, const MapKey_T& iKey) {
00396
00397         AirlineFeature* lAirlineFeature_ptr = iInventory.
getAirlineFeature ();
00398
00399         return lAirlineFeature_ptr;
00400     }
00401
00402     // Specialization of the template method getObject above for the types
00403     // <AirlineFeature,Inventory>.
00404     template<>
00405     inline AirlineFeature& BomManager::getObject<AirlineFeature,Inventory>
00406     (const Inventory& iInventory, const MapKey_T& iKey) {
00407
00408         AirlineFeature* lAirlineFeature_ptr =
00409             getObjectPtr<AirlineFeature,Inventory> (iInventory, iKey);
00410         assert (lAirlineFeature_ptr != NULL);
00411
00412         return *lAirlineFeature_ptr;
00413     }
00414
00415 }
00416 }
00417 #endif // __STDAIR_BOM_BOMMANAGER_HPP

```

### 33.211 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

- `stdair`

*Handle on the StdAir library context.*

## 33.212 BomRetriever.cpp

```

00001 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00002 // Import section
00003 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_Inventory.hpp>
00009 #include <stdair/basic/BasConst_BomDisplay.hpp>
00010 #include <stdair/bom/BomKeyManager.hpp>
00011 #include <stdair/bom/BomManager.hpp>
00012 #include <stdair/bom/BomRoot.hpp>
00013 #include <stdair/bom/Inventory.hpp>
00014 #include <stdair/bom/AirlineFeature.hpp>
00015 #include <stdair/bom/FlightDate.hpp>
00016 #include <stdair/bom/LegDate.hpp>
00017 #include <stdair/bom/SegmentDate.hpp>
00018 #include <stdair/bom/LegCabin.hpp>
00019 #include <stdair/bom/SegmentCabin.hpp>
00020 #include <stdair/bom/FareFamily.hpp>
00021 #include <stdair/bom/BookingClass.hpp>
00022 #include <stdair/bom/BomRetriever.hpp>
00023 #include <stdair/bom/ParsedKey.hpp>
00024 #include <stdair/bom/AirportPair.hpp>
00025 #include <stdair/service/Logger.hpp>
00026
00027 namespace stdair {
00028
00029 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00030 Inventory* BomRetriever::
00031 retrieveInventoryFromLongKey (const BomRoot& iBomRoot,
00032                             const std::string& iFullKeyStr) {
00033     Inventory* oInventory_ptr = NULL;
00034
00035     // Extract the inventory key (i.e., airline code)
00036     const InventoryKey& lInventoryKey =
00037         BomKeyManager::extractInventoryKey (iFullKeyStr);
00038
00039     oInventory_ptr = iBomRoot.getInventory (lInventoryKey);
00040
00041     return oInventory_ptr;
00042 }
00043
00044 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00045 Inventory* BomRetriever::
00046 retrieveInventoryFromLongKey (const Inventory& iInventory,
00047                             const std::string& iFullKeyStr) {
00048     Inventory* oInventory_ptr = NULL;
00049
00050     // Extract the inventory key (i.e., airline code)
00051     const InventoryKey& lInventoryKey =
00052         BomKeyManager::extractInventoryKey (iFullKeyStr);
00053     const stdair::AirlineCode_T lAirlineCode =
00054         lInventoryKey.getAirlineCode();
00055
00056     oInventory_ptr =
00057         BomManager::getObjectPtr<Inventory> (iInventory,
00058                                             lAirlineCode);
00059     return oInventory_ptr;
00060 }
00061
00062 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00063 Inventory* BomRetriever::retrieveInventoryFromKey (const
00064 BomRoot& iBomRoot,
00065                                             const InventoryKey& iKey) {
00066     Inventory* oInventory_ptr = NULL;
00067
00068     //
00069     oInventory_ptr = iBomRoot.getInventory (iKey);
00070
00071     return oInventory_ptr;
00072 }
00073
00074 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00075 Inventory* BomRetriever::
00076 retrieveInventoryFromKey (const BomRoot& iBomRoot,

```

```

00076             const AirlineCode_T& iAirlineCode) {
00077     Inventory* oInventory_ptr = NULL;
00078
00079     //
00080     const InventoryKey lKey (iAirlineCode);
00081     oInventory_ptr = iBomRoot.getInventory (lKey);
00082
00083     return oInventory_ptr;
00084 }
00085
00086 // //////////////////////////////////////
00087 AirlineFeature* BomRetriever::
00088 retrieveAirlineFeatureFromKey (const BomRoot& iBomRoot,
00089                               const AirlineCode_T& iAirlineCode) {
00090     Inventory* oInventory_ptr = NULL;
00091     AirlineFeature* oAirlineFeature_ptr = NULL;
00092
00093     //
00094     oInventory_ptr = retrieveInventoryFromKey (iBomRoot, iAirlineCode);
00095     if (oInventory_ptr == NULL) {
00096         return oAirlineFeature_ptr;
00097     }
00098     assert (oInventory_ptr != NULL);
00099
00100     oAirlineFeature_ptr =
00101         BomManager::getObjectPtr<AirlineFeature,Inventory> (*oInventory_ptr,
00102                                                             iAirlineCode);
00103
00104     return oAirlineFeature_ptr;
00105 }
00106
00107 // //////////////////////////////////////
00108 FlightDate* BomRetriever::
00109 retrieveFlightDateFromLongKey (const BomRoot& iBomRoot,
00110                               const std::string& iFullKeyStr) {
00111     FlightDate* oFlightDate_ptr = NULL;
00112
00113     // Retrieve the inventory
00114     Inventory* oInventory_ptr =
00115         BomRetriever::retrieveInventoryFromLongKey (iBomRoot,
00116 iFullKeyStr);
00117     if (oInventory_ptr == NULL) {
00118         return oFlightDate_ptr;
00119     }
00120     assert (oInventory_ptr != NULL);
00121
00122     // Extract the flight-date key (i.e., flight number and date)
00123     const FlightDateKey& lFlightDateKey =
00124         BomKeyManager::extractFlightDateKey (iFullKeyStr);
00125
00126     oFlightDate_ptr = oInventory_ptr->getFlightDate (lFlightDateKey);
00127
00128     return oFlightDate_ptr;
00129 }
00130
00131 // //////////////////////////////////////
00132 FlightDate* BomRetriever::
00133 retrieveFlightDateFromKeySet (const BomRoot& iBomRoot,
00134                               const AirlineCode_T& iAirlineCode,
00135                               const FlightNumber_T& iFlightNumber,
00136                               const Date_T& iFlightDateDate) {
00137     FlightDate* oFlightDate_ptr = NULL;
00138
00139     // Retrieve the inventory
00140     Inventory* oInventory_ptr =
00141         BomRetriever::retrieveInventoryFromKey (iBomRoot, iAirlineCode)
00142 ;
00143     if (oInventory_ptr == NULL) {
00144         return oFlightDate_ptr;
00145     }
00146     assert (oInventory_ptr != NULL);
00147
00148     //
00149     oFlightDate_ptr = retrieveFlightDateFromKey (*oInventory_ptr,
00150 iFlightNumber, iFlightDateDate);
00151
00152     return oFlightDate_ptr;
00153 }
00154
00155 // //////////////////////////////////////
00156 FlightDate* BomRetriever::
00157 retrieveFlightDateFromLongKey (const Inventory& iInventory,
00158                               const std::string& iFullKeyStr) {
00159     FlightDate* oFlightDate_ptr = NULL;
00160
00161     // Extract the flight-date key (i.e., flight number and date)
00162     const FlightDateKey& lFlightDateKey =

```

```

00161         BomKeyManager::extractFlightDateKey (iFullKeyStr);
00162
00163         oFlightDate_ptr = iInventory.getFlightDate (lFlightDateKey);
00164
00165         return oFlightDate_ptr;
00166     }
00167
00168     // //////////////////////////////////////
00169     FlightDate* BomRetriever::
00170     retrieveFlightDateFromKey (const Inventory& iInventory,
00171                             const FlightDateKey& iKey) {
00172         FlightDate* oFlightDate_ptr = NULL;
00173
00174         //
00175         oFlightDate_ptr = iInventory.getFlightDate (iKey);
00176
00177         return oFlightDate_ptr;
00178     }
00179
00180     // //////////////////////////////////////
00181     FlightDate* BomRetriever::
00182     retrieveFlightDateFromKey (const Inventory& iInventory,
00183                             const FlightNumber_T& iFlightNumber,
00184                             const Date_T& iFlightDateDate) {
00185         FlightDate* oFlightDate_ptr = NULL;
00186
00187         //
00188         const FlightDateKey lKey (iFlightNumber, iFlightDateDate);
00189         oFlightDate_ptr = iInventory.getFlightDate (lKey);
00190
00191         return oFlightDate_ptr;
00192     }
00193
00194     // //////////////////////////////////////
00195     SegmentDate* BomRetriever::
00196     retrieveSegmentDateFromLongKey (const BomRoot& iBomRoot,
00197                                   const std::string& iFullKeyStr) {
00198         SegmentDate* oSegmentDate_ptr = NULL;
00199
00200         // Retrieve the flight-date
00201         FlightDate* oFlightDate_ptr =
00202             BomRetriever::retrieveFlightDateFromLongKey (iBomRoot,
00203 iFullKeyStr);
00204         if (oFlightDate_ptr == NULL) {
00205             return oSegmentDate_ptr;
00206         }
00207         assert (oFlightDate_ptr != NULL);
00208
00209         // Extract the segment-date key (i.e., origin and destination)
00210         const SegmentDateKey& lSegmentDateKey =
00211             BomKeyManager::extractSegmentDateKey (iFullKeyStr);
00212
00213         oSegmentDate_ptr = oFlightDate_ptr->getSegmentDate (lSegmentDateKey);
00214
00215         return oSegmentDate_ptr;
00216     }
00217
00218     // //////////////////////////////////////
00219     SegmentDate* BomRetriever::
00220     retrieveSegmentDateFromLongKey (const
00221     Inventory& iInventory,
00222                                   const std::string& iFullKeyStr) {
00223         SegmentDate* oSegmentDate_ptr = NULL;
00224
00225         //
00226         const std::string lFullKeyStr = iFullKeyStr;
00227         const std::string lAirlineCode = BomKeyManager::extractAirlineCode (lFullKeyStr);
00228         const std::string lFlightDateKey = BomKeyManager::extractFlightDateKey (lFullKeyStr);
00229         const std::string lSegmentDateKey = BomKeyManager::extractSegmentDateKey (lFullKeyStr);
00230
00231         if (iInventory.getAirlineCode() != lAirlineCode) {
00232             STDAIR_LOG_DEBUG ("Airline code: " << lAirlineCode);
00233             return oSegmentDate_ptr;
00234         }
00235
00236         FlightDate* lFlightDate_ptr =
00237             retrieveFlightDateFromKey (iInventory, lFlightDateKey);
00238         if (lFlightDate_ptr == NULL) {
00239             STDAIR_LOG_DEBUG ("Flight-date key: "
00240                             << lFlightDateKey);
00241             return oSegmentDate_ptr;
00242         }
00243
00244         oSegmentDate_ptr =
00245             retrieveSegmentDateFromKey (*lFlightDate_ptr, lSegmentDateKey);
00246         if (oSegmentDate_ptr == NULL) {
00247             STDAIR_LOG_DEBUG ("Segment-date key: "

```

```

00242         << lParsedKey.getSegmentKey().toString();
00243     return oSegmentDate_ptr;
00244 }
00245
00246     return oSegmentDate_ptr;
00247 }
00248
00249 ///////////////////////////////////////////////////////////////////
00250 SegmentDate* BomRetriever::
00251 retrieveSegmentDateFromLongKey (const
FlightDate& iFlightDate,
00252                                 const std::string& iFullKeyStr) {
00253     SegmentDate* oSegmentDate_ptr = NULL;
00254
00255     // Extract the segment-date key (i.e., origin and destination)
00256     const SegmentDateKey& lSegmentDateKey =
00257         BomKeyManager::extractSegmentDateKey (iFullKeyStr);
00258
00259     oSegmentDate_ptr = iFlightDate.getSegmentDate (lSegmentDateKey);
00260
00261     return oSegmentDate_ptr;
00262 }
00263
00264 ///////////////////////////////////////////////////////////////////
00265 LegDate* BomRetriever::
00266 retrieveOperatingLegDateFromLongKey (const
FlightDate& iFlightDate,
00267                                     const std::string& iFullKeyStr) {
00268     LegDate* oLegDate_ptr = NULL;
00269
00270     // Extract the segment-date key (i.e., origin and destination)
00271     const LegDateKey& lLegDateKey =
00272         BomKeyManager::extractLegDateKey (iFullKeyStr);
00273
00274     oLegDate_ptr = iFlightDate.getLegDate (lLegDateKey);
00275
00276     return oLegDate_ptr;
00277 }
00278
00279 ///////////////////////////////////////////////////////////////////
00280 SegmentDate* BomRetriever::
00281 retrievePartnerSegmentDateFromLongKey (const
Inventory& iInventory,
00282                                       const std::string& iFullKeyStr) {
00283     SegmentDate* oSegmentDate_ptr = NULL;
00284     Inventory* oInventory_ptr = NULL;
00285
00286     // Extract the inventory key (i.e., airline code)
00287     const InventoryKey& lInventoryKey =
00288         BomKeyManager::extractInventoryKey (iFullKeyStr);
00289     const stdair::AirlineCode_T lAirlineCode =
00290         lInventoryKey.getAirlineCode();
00291
00292     // Retrieve the inventory
00293     oInventory_ptr =
00294         retrieveInventoryFromLongKey (iInventory, lAirlineCode);
00295
00296     if (oInventory_ptr != NULL) {
00297         oSegmentDate_ptr =
00298             retrieveSegmentDateFromLongKey (*oInventory_ptr, iFullKeyStr);
00299     }
00300
00301     return oSegmentDate_ptr;
00302 }
00303
00304 ///////////////////////////////////////////////////////////////////
00305 SegmentDate* BomRetriever::
00306 retrieveSegmentDateFromKey (const FlightDate& iFlightDate,
00307                             const SegmentDateKey& iKey) {
00308     SegmentDate* oSegmentDate_ptr = NULL;
00309
00310     //
00311     oSegmentDate_ptr = iFlightDate.getSegmentDate (iKey);
00312
00313     return oSegmentDate_ptr;
00314 }
00315
00316 ///////////////////////////////////////////////////////////////////
00317 SegmentDate* BomRetriever::
00318 retrieveSegmentDateFromKey (const FlightDate& iFlightDate,
00319                             const AirportCode_T& iOrigin,
00320                             const AirportCode_T& iDestination) {
00321     SegmentDate* oSegmentDate_ptr = NULL;
00322
00323     //
00324     const SegmentDateKey lKey (iOrigin, iDestination);
00325

```

```

00326     oSegmentDate_ptr = iFlightDate.getSegmentDate (lKey);
00327
00328     return oSegmentDate_ptr;
00329 }
00330
00331 // //////////////////////////////////////
00332 BookingClass* BomRetriever::
00333 retrieveBookingClassFromLongKey (const
Inventory& iInventory,
                                const std::string& iFullKeyStr,
                                const ClassCode_T& iClassCode) {
00334     BookingClass* oBookingClass_ptr = NULL;
00335
00336     SegmentDate* lSegmentDate_ptr = retrieveSegmentDateFromLongKey
(iInventory,
                                iFullKeyStr);
00337
00338     if (lSegmentDate_ptr == NULL) {
00339         return oBookingClass_ptr;
00340     }
00341     assert (lSegmentDate_ptr != NULL);
00342
00343     //
00344     oBookingClass_ptr =
00345         BomManager::getObjectPtr<BookingClass> (*lSegmentDate_ptr, iClassCode);
00346
00347     return oBookingClass_ptr;
00348 }
00349
00350 // //////////////////////////////////////
00351 AirportPair* BomRetriever::
00352 retrieveAirportPairFromKeySet (const BomRoot& iBomRoot,
                                const stdair::AirportCode_T& iOrigin,
                                const stdair::AirportCode_T& iDestination) {
00353
00354     // Get the Airport pair stream of the segment path.
00355     const AirportPairKey lAirportPairKey (iOrigin, iDestination);
00356
00357     // Search for the fare rules having the same origin and
00358     // destination airport as the travel solution
00359     AirportPair* oAirportPair_ptr = BomManager::
00360         getObjectPtr<AirportPair> (iBomRoot, lAirportPairKey.toString());
00361
00362     return oAirportPair_ptr;
00363 }
00364
00365 // //////////////////////////////////////
00366 void BomRetriever::
00367 retrieveDatePeriodListFromKey (const
AirportPair& iAirportPair,
                                const stdair::Date_T& iDepartureDate,
                                stdair::DatePeriodList_T& ioDatePeriodList) {
00368
00369     // Get the list of date-period
00370     const DatePeriodList_T& lFareDatePeriodList =
00371         BomManager::getList<DatePeriod> (iAirportPair);
00372
00373     // Browse the date-period list
00374     for (DatePeriodList_T::const_iterator itDateRange =
00375         lFareDatePeriodList.begin();
00376         itDateRange != lFareDatePeriodList.end(); ++itDateRange) {
00377
00378         DatePeriod* lCurrentFareDatePeriod_ptr = *itDateRange ;
00379         assert (lCurrentFareDatePeriod_ptr != NULL);
00380
00381         // Select the date-period objects having a corresponding date range
00382         const bool isDepartureDateValid =
00383             lCurrentFareDatePeriod_ptr->isDepartureDateValid (iDepartureDate);
00384
00385         // Add the date-period objects having a corresponding date range
00386         // to the list to display
00387         if (isDepartureDateValid == true) {
00388             ioDatePeriodList.push_back(lCurrentFareDatePeriod_ptr);
00389         }
00390     }
00391 }
00392
00393 // //////////////////////////////////////
00394 void BomRetriever::
00395 retrieveDatePeriodListFromKeySet (const
BomRoot& iBomRoot,
                                const stdair::AirportCode_T& iOrigin,
                                const stdair::AirportCode_T& iDestination,
                                const stdair::Date_T& iDepartureDate,
                                stdair::DatePeriodList_T& ioDatePeriodList) {

```

```

00409
00410 // Retrieve the airport-pair
00411 AirportPair* oAirportPair_ptr =
00412 BomRetriever::retrieveAirportPairFromKeySet (iBomRoot,
iOrigin,
00413 iDestination);
00414 if (oAirportPair_ptr == NULL) {
00415     return;
00416 }
00417 assert (oAirportPair_ptr != NULL);
00418
00419 // Retrieve the flight date
00420 BomRetriever::retrieveDatePeriodListFromKey (*
oAirportPair_ptr, iDepartureDate,
00421 ioDatePeriodList);
00422
00423 }
00424
00425 // //////////////////////////////////////
00426 LegCabin& BomRetriever::
00427 retrieveDummyLegCabin (stdair::BomRoot& iBomRoot,
00428 const bool isForFareFamilies) {
00429
00430 LegCabin* oLegCabin_ptr = NULL;
00431
00432 // Retrieve the Inventory
00433 const Inventory* lInventory_ptr = BomRetriever::
00434 retrieveInventoryFromKey (iBomRoot,
DEFAULT_AIRLINE_CODE);
00435
00436 if (lInventory_ptr == NULL) {
00437     std::ostringstream oStr;
00438     oStr << "The inventory corresponding to the '"
00439 << DEFAULT_AIRLINE_CODE << "' airline can not be found";
00440     throw ObjectNotFoundException (oStr.str());
00441 }
00442
00443 // Retrieve the FlightDate
00444 FlightDate* lFlightDate_ptr = NULL;
00445 if (isForFareFamilies == true) {
00446     lFlightDate_ptr = BomRetriever::
00447 retrieveFlightDateFromKey (*lInventory_ptr,
DEFAULT_FLIGHT_NUMBER_FF,
00448 DEFAULT_DEPARTURE_DATE);
00449
00450 if (lFlightDate_ptr == NULL) {
00451     std::ostringstream oStr;
00452     oStr << "The flight-date corresponding to ("
00453 << DEFAULT_FLIGHT_NUMBER_FF << ", "
00454 << DEFAULT_DEPARTURE_DATE << ") can not be found";
00455     throw ObjectNotFoundException (oStr.str());
00456 }
00457 } else {
00458     lFlightDate_ptr = BomRetriever::
00459 retrieveFlightDateFromKey (*lInventory_ptr,
DEFAULT_FLIGHT_NUMBER,
00460 DEFAULT_DEPARTURE_DATE);
00461
00462 if (lFlightDate_ptr == NULL) {
00463     std::ostringstream oStr;
00464     oStr << "The flight-date corresponding to ("
00465 << DEFAULT_FLIGHT_NUMBER << ", "
00466 << DEFAULT_DEPARTURE_DATE << ") can not be found";
00467     throw ObjectNotFoundException (oStr.str());
00468 }
00469 }
00470 assert(lFlightDate_ptr != NULL);
00471
00472 // Retrieve the LegDate
00473 const LegDateKey lLegDateKey (DEFAULT_ORIGIN);
00474 const LegDate* lLegDate_ptr =
00475 lFlightDate_ptr->getLegDate (lLegDateKey);
00476
00477 if (lLegDate_ptr == NULL) {
00478     std::ostringstream oStr;
00479     oStr << "The leg-date corresponding to the '"
00480 << DEFAULT_ORIGIN << "' origin can not be found";
00481     throw ObjectNotFoundException (oStr.str());
00482 }
00483
00484 // Retrieve the LegCabin
00485 const LegCabinKey lLegCabinKey (DEFAULT_CABIN_CODE);
00486 oLegCabin_ptr = lLegDate_ptr->getLegCabin (lLegCabinKey);
00487
00488 if (oLegCabin_ptr == NULL) {
00489     std::ostringstream oStr;
00490     oStr << "The leg-cabin corresponding to the '"

```

```

00491         << DEFAULT_CABIN_CODE << "' cabin code can not be found";
00492         throw ObjectNotFoundException (oStr.str());
00493     }
00494
00495     assert (oLegCabin_ptr != NULL);
00496     return *oLegCabin_ptr;
00497
00498 }
00499
00500 // //////////////////////////////////////
00501 SegmentCabin& BomRetriever::
00502 retrieveDummySegmentCabin (stdair::BomRoot& iBomRoot,
00503                             const bool isForFareFamilies) {
00504
00505     SegmentCabin* oSegmentCabin_ptr = NULL;
00506
00507     // Retrieve the Inventory
00508     const Inventory* lInventory_ptr = BomRetriever::
00509         retrieveInventoryFromKey (iBomRoot,
00510     DEFAULT_AIRLINE_CODE);
00511
00512     if (lInventory_ptr == NULL) {
00513         std::ostringstream oStr;
00514         oStr << "The inventory corresponding to the '"
00515             << DEFAULT_AIRLINE_CODE << "' airline can not be found";
00516         throw ObjectNotFoundException (oStr.str());
00517     }
00518
00519     // Retrieve the FlightDate
00520     FlightDate* lFlightDate_ptr = NULL;
00521     if (isForFareFamilies == true) {
00522         lFlightDate_ptr = BomRetriever::
00523             retrieveFlightDateFromKey (*lInventory_ptr,
00524     DEFAULT_FLIGHT_NUMBER_FF,
00525                                     DEFAULT_DEPARTURE_DATE);
00526
00527         if (lFlightDate_ptr == NULL) {
00528             std::ostringstream oStr;
00529             oStr << "The flight-date corresponding to ("
00530                 << DEFAULT_FLIGHT_NUMBER_FF << ", "
00531                 << DEFAULT_DEPARTURE_DATE << ") can not be found";
00532             throw ObjectNotFoundException (oStr.str());
00533         }
00534     } else {
00535         lFlightDate_ptr = BomRetriever::
00536             retrieveFlightDateFromKey (*lInventory_ptr,
00537     DEFAULT_FLIGHT_NUMBER,
00538                                     DEFAULT_DEPARTURE_DATE);
00539
00540         if (lFlightDate_ptr == NULL) {
00541             std::ostringstream oStr;
00542             oStr << "The flight-date corresponding to ("
00543                 << DEFAULT_FLIGHT_NUMBER << ", "
00544                 << DEFAULT_DEPARTURE_DATE << ") can not be found";
00545             throw ObjectNotFoundException (oStr.str());
00546         }
00547     }
00548     assert(lFlightDate_ptr != NULL);
00549
00550     // Retrieve the SegmentDate
00551     const SegmentDateKey lSegmentDateKey (DEFAULT_ORIGIN,
00552     DEFAULT_DESTINATION);
00553     const SegmentDate* lSegmentDate_ptr =
00554         lFlightDate_ptr->getSegmentDate (lSegmentDateKey);
00555
00556     if (lSegmentDate_ptr == NULL) {
00557         std::ostringstream oStr;
00558         oStr << "The segment-date corresponding to the '"
00559             << DEFAULT_ORIGIN << "' origin and '"
00560             << DEFAULT_DESTINATION << "' destination can not be found";
00561         throw ObjectNotFoundException (oStr.str());
00562     }
00563
00564     // Retrieve the SegmentCabin
00565     const SegmentCabinKey lSegmentCabinKey (DEFAULT_CABIN_CODE);
00566     oSegmentCabin_ptr =
00567         BomManager::getObjectPtr<SegmentCabin> (*lSegmentDate_ptr, lSegmentCabinKey.
00568     toString());
00569
00570     if (oSegmentCabin_ptr == NULL) {
00571         std::ostringstream oStr;
00572         oStr << "The segment-cabin corresponding to the '"
00573             << DEFAULT_CABIN_CODE << "' cabin code can not be found";
00574         throw ObjectNotFoundException (oStr.str());
00575     }
00576
00577     assert (oSegmentCabin_ptr != NULL);

```

```

00573     return *oSegmentCabin_ptr;
00574 }
00575
00576 // //////////////////////////////////////
00577 std::string BomRetriever::
00578 retrieveFullKeyFromSegmentDate (const
SegmentDate& iSegmentdate) {
00579
00580     std::ostringstream lFullKeyStr;
00581
00582     // Get the parent flight date
00583     FlightDate* lFlightDate_ptr =
00584         BomManager::getParentPtr<FlightDate>(iSegmentdate);
00585     if (lFlightDate_ptr == NULL) {
00586         return lFullKeyStr.str();
00587     }
00588     assert (lFlightDate_ptr != NULL);
00589
00590     // Get the parent inventory
00591     Inventory* lInventory_ptr =
00592         BomManager::getParentPtr<Inventory> (*lFlightDate_ptr);
00593     if (lInventory_ptr == NULL) {
00594         return lFullKeyStr.str();
00595     }
00596     assert (lInventory_ptr != NULL);
00597
00598     lFullKeyStr << lInventory_ptr->describeKey()
00599         << DEFAULT_KEY_SUB_FLD_DELIMITER;
00600     lFullKeyStr << lFlightDate_ptr->describeKey()
00601         << DEFAULT_KEY_SUB_FLD_DELIMITER;
00602     lFullKeyStr << iSegmentdate.describeKey();
00603
00604     return lFullKeyStr.str();
00605 }
00606 }
00607
00608 }

```

### 33.213 stdair/bom/BomRetriever.hpp File Reference

```

#include <iosfwd>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/bom/DatePeriod.hpp>

```

#### Classes

- class [stdair::BomRetriever](#)  
*Utility class to retrieve StdAir objects.*

#### Namespaces

- [stdair](#)  
*Handle on the StdAir library context.*

### 33.214 BomRetriever.hpp

```

00001 #ifndef __STDAIR_BOM_BOMRETRIEVER_HPP
00002 #define __STDAIR_BOM_BOMRETRIEVER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 // StdAir
00010 #include <stdair/stdair_basic_types.hpp>
00011 #include <stdair/stdair_date_time_types.hpp>
00012 #include <stdair/bom/DatePeriod.hpp>
00013

```



```

00014 namespace stdair {
00015
00017     class BomRoot;
00018     struct InventoryKey;
00019     class Inventory;
00020     class AirlineFeature;
00021     struct FlightDateKey;
00022     class FlightDate;
00023     class LegDate;
00024     struct SegmentDateKey;
00025     class SegmentDate;
00026     class LegCabin;
00027     class SegmentCabin;
00028     class FareFamily;
00029     class BookingClass;
00030     class DatePeriod;
00031     class AirportPair;
00032
00036     class BomRetriever {
00037     public:
00038         // ////////////////////////////////// Key management support methods //////////////////////////////////
00051         static Inventory*
00052         retrieveInventoryFromLongKey (const BomRoot&,
00053                                     const std::string& iFullKeyStr);
00054
00067         static Inventory*
00068         retrieveInventoryFromLongKey (const Inventory&,
00069                                     const std::string& iFullKeyStr);
00070
00078         static Inventory* retrieveInventoryFromKey (const
BomRoot&,
00079                                                    const InventoryKey&);
00080
00088         static Inventory* retrieveInventoryFromKey (const
BomRoot&,
00089                                                    const AirlineCode_T&);
00090
00098         static AirlineFeature* retrieveAirlineFeatureFromKey (const
BomRoot&,
00099                                                    const AirlineCode_T&);
00100
00113         static FlightDate*
00114         retrieveFlightDateFromLongKey (const BomRoot&,
00115                                     const std::string& iFullKeyStr);
00116
00126         static FlightDate*
00127         retrieveFlightDateFromKeySet (const BomRoot&,
00128                                     const AirlineCode_T&, const
FlightNumber_T&,
00129                                     const Date_T& iFlightDateDate);
00130
00143         static FlightDate*
00144         retrieveFlightDateFromLongKey (const Inventory&,
00145                                     const std::string& iFullKeyStr);
00146
00154         static FlightDate* retrieveFlightDateFromKey (const
Inventory&,
00155                                                    const FlightDateKey&);
00156
00165         static FlightDate* retrieveFlightDateFromKey (const
Inventory&,
00166                                                    const FlightNumber_T&,
00167                                                    const Date_T& iFlightDateDate);
00168
00169
00182         static LegDate*
00183         retrieveOperatingLegDateFromLongKey (const
FlightDate&,
00184                                                    const std::string& iFullKeyStr);
00185
00198         static SegmentDate*
00199         retrievePartnerSegmentDateFromLongKey (const
Inventory&,
00200                                                    const std::string& iFullKeyStr);
00201
00214         static SegmentDate*
00215         retrieveSegmentDateFromLongKey (const BomRoot&,
00216                                     const std::string& iFullKeyStr);
00217
00230         static SegmentDate*
00231         retrieveSegmentDateFromLongKey (const
Inventory&,
00232                                                    const std::string& iFullKeyStr);
00233
00246         static SegmentDate*
00247         retrieveSegmentDateFromLongKey (const
FlightDate&,

```

```

00248                                     const std::string& iFullKeyStr);
00249
00257     static SegmentDate* retrieveSegmentDateFromKey (const
FlightDate&,
00258                                     const SegmentDateKey&);
00259
00268     static SegmentDate*
00269     retrieveSegmentDateFromKey (const FlightDate&,
00270                                 const AirportCode_T& iOrigin,
00271                                 const AirportCode_T& iDestination);
00272
00296     static BookingClass*
00297     retrieveBookingClassFromLongKey (const
Inventory&,
00298                                     const std::string& iFullKeyStr,
00299                                     const ClassCode_T&);
00300
00301
00310     static AirportPair*
00311     retrieveAirportPairFromKeySet (const BomRoot& ,
00312                                    const stdair::AirportCode_T&,
00313                                    const stdair::AirportCode_T&);
00314
00324     static void
00325     retrieveDatePeriodListFromKey (const
AirportPair&,
00326                                     const stdair::Date_T&,
00327                                     stdair::DatePeriodList_T&);
00328
00341     static void
00342     retrieveDatePeriodListFromKeySet (const
BomRoot&,
00343                                     const stdair::AirportCode_T&,
00344                                     const stdair::AirportCode_T&,
00345                                     const stdair::Date_T&,
00346                                     stdair::DatePeriodList_T&);
00347
00357     static stdair::LegCabin&
00358     retrieveDummyLegCabin (stdair::BomRoot&,
00359                            const bool isForFareFamilies = false);
00360
00370     static stdair::SegmentCabin&
00371     retrieveDummySegmentCabin (stdair::BomRoot&,
00372                                const bool isForFareFamilies = false);
00373
00383     static std::string retrieveFullKeyFromSegmentDate (const
SegmentDate&);
00384
00385 };
00386
00387 }
00388 #endif // __STDAIR_BOM_BOMRETRIEVER_HPP

```

### 33.215 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

- [stdair](#)

*Handle on the StdAir library context.*

### 33.216 BomRoot.cpp

```

00001 // //////////////////////////////////////
00002 // Import section

```

```

00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_General.hpp>
00009 #include <stdair/bom/BomManager.hpp>
00010 #include <stdair/bom/BomRoot.hpp>
00011 #include <stdair/bom/InventoryKey.hpp>
00012 #include <stdair/bom/Inventory.hpp>
00013
00014 namespace stdair {
00015
00016 // //////////////////////////////////////
00017 BomRoot::BomRoot() {
00018     assert (false);
00019 }
00020
00021 // //////////////////////////////////////
00022 BomRoot::BomRoot (const BomRoot& iBomRoot) :
00023     _key (iBomRoot._key), _frat5CurveHolder (iBomRoot._frat5CurveHolder),
00024     _ffDisutilityCurveHolder (iBomRoot._ffDisutilityCurveHolder) {
00025 }
00026
00027 // //////////////////////////////////////
00028 BomRoot::BomRoot (const Key_T& iKey) : _key (iKey) {
00029 }
00030
00031 // //////////////////////////////////////
00032 BomRoot::~BomRoot() {
00033 }
00034
00035 // //////////////////////////////////////
00036 std::string BomRoot::toString() const {
00037     std::ostringstream oStr;
00038     oStr << _key.toString();
00039     return oStr.str();
00040 }
00041
00042 // //////////////////////////////////////
00043 Inventory* BomRoot::getInventory (const std::string& iInventoryKeyStr)
00044 const {
00045     Inventory* oInventory_ptr =
00046         BomManager::getObjectPtr<Inventory> (*this, iInventoryKeyStr);
00047     return oInventory_ptr;
00048 }
00049
00050 // //////////////////////////////////////
00050 Inventory* BomRoot::getInventory (const
00051     InventoryKey& iInventoryKey) const {
00052     return getInventory (iInventoryKey.toString());
00053 }
00054 }

```

### 33.217 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>

```

#### Classes

- class [stdair::BomRoot](#)  
Class representing the actual attributes for the Bom root.

#### Namespaces

- [boost](#)

*Forward declarations.*

- [boost::serialization](#)
- [stdair](#)

*Handle on the StdAir library context.*

### 33.218 BomRoot.hpp

```

00001 #ifndef __STDAIR_BOM_BOMROOT_HPP
00002 #define __STDAIR_BOM_BOMROOT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/bom/BomAbstract.hpp>
00012 #include <stdair/bom/BomRootKey.hpp>
00013 #include <stdair/bom/FRAT5CurveHolderStruct.hpp>
00014 #include <stdair/bom/FFDisutilityCurveHolderStruct.hpp>
00015
00016 namespace boost {
00017     namespace serialization {
00018         class access;
00019     }
00020 }
00021
00022 namespace stdair {
00023
00024     struct InventoryKey;
00025     class Inventory;
00026
00027     class BomRoot : public BomAbstract {
00028     public:
00029         template <typename BOM> friend class FacBom;
00030         template <typename BOM> friend class FacCloneBom;
00031         friend class FacBomManager;
00032         friend class boost::serialization::access;
00033
00034     public:
00035         typedef BomRootKey Key_T;
00036
00037     public:
00038         // ////////// Getters //////////
00039         const Key_T& getKey() const {
00040             return _key;
00041         }
00042
00043         const HolderMap_T& getHolderMap() const {
00044             return _holderMap;
00045         }
00046
00047         const FRAT5Curve_T& getFRAT5Curve (const std::string& iKey) const {
00048             return _frat5CurveHolder.getFRAT5Curve (iKey);
00049         }
00050
00051         const FFDisutilityCurve_T& getFFDisutilityCurve (const
00052             std::string& iKey) const{
00053             return _ffDisutilityCurveHolder.
00054                 getFFDisutilityCurve (iKey);
00055         }
00056
00057         Inventory* getInventory (const std::string& iInventoryKeyStr) const;
00058
00059         Inventory* getInventory (const InventoryKey&) const;
00060
00061         // ////////// Business Methods //////////
00062         void addFRAT5Curve (const std::string& iKey, const FRAT5Curve_T& iCurve) {
00063             _frat5CurveHolder.addCurve (iKey, iCurve);
00064         }
00065
00066         void addFFDisutilityCurve (const std::string& iKey,
00067             const FFDisutilityCurve_T& iCurve) {
00068             _ffDisutilityCurveHolder.addCurve (iKey, iCurve);
00069         }
00070
00071     public:
00072         // ////////// Display support methods //////////
00073         void toStream (std::ostream& ioOut) const {
00074             ioOut << toString();
00075         }
00076     };
00077
00078 }
00079
00080 #endif

```

```

00113     }
00114
00120 void fromStream (std::istream& ioIn) {
00121     }
00122
00126     std::string toString() const;
00127
00131     const std::string describeKey() const {
00132         return _key.toString();
00133     }
00134
00135
00136 public:
00137     // ////////// (Boost) Serialisation support methods //////////
00148     template<class Archive>
00149     void serialize (Archive& ar, const unsigned int iFileVersion);
00150
00151 private:
00159     void serialisationImplementationExport() const;
00160     void serialisationImplementationImport();
00161
00162
00163 protected:
00164     // ////////// Constructors and destructors //////////
00168     BomRoot();
00169
00173     BomRoot (const BomRoot&);
00174
00178     BomRoot (const Key_T& iKey);
00179
00183     ~BomRoot();
00184
00185
00186 protected:
00187     // ////////// Attributes //////////
00191     Key_T _key;
00192
00196     HolderMap_T _holderMap;
00197
00201     FRAT5CurveHolderStruct _frat5CurveHolder;
00202
00206     FFDisutilityCurveHolderStruct
00207     _ffDisutilityCurveHolder;
00208 };
00209 }
00210 #endif // __STDAIR_BOM_BOMROOT_HPP

```

### 33.219 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

- [stdair](#)

*Handle on the StdAir library context.*

#### Functions

- template void [stdair::BomRootKey::serialize< ba::text\\_oarchive >](#) (ba::text\_oarchive &, unsigned int)
- template void [stdair::BomRootKey::serialize< ba::text\\_iarchive >](#) (ba::text\_iarchive &, unsigned int)

## 33.220 BomRootKey.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/access.hpp>
00011 // StdAir
00012 #include <stdair/basic/BasConst_General.hpp>
00013 #include <stdair/bom/BomRootKey.hpp>
00014
00015 namespace stdair {
00016
00017 // //////////////////////////////////////
00018 BomRootKey::BomRootKey()
00019     : _id (DEFAULT_BOM_ROOT_KEY) {
00020 }
00021
00022 // //////////////////////////////////////
00023 BomRootKey::BomRootKey (const BomRootKey& iBomRootKey)
00024     : _id (iBomRootKey._id) {
00025 }
00026
00027 // //////////////////////////////////////
00028 BomRootKey::BomRootKey (const std::string& iIdentification)
00029     : _id (iIdentification) {
00030 }
00031
00032 // //////////////////////////////////////
00033 BomRootKey::~BomRootKey () {
00034 }
00035
00036 // //////////////////////////////////////
00037 void BomRootKey::toStream (std::ostream& ioOut) const {
00038     ioOut << "BomRootKey: " << toString() << std::endl;
00039 }
00040
00041 // //////////////////////////////////////
00042 void BomRootKey::fromStream (std::istream& ioIn) {
00043 }
00044
00045 // //////////////////////////////////////
00046 const std::string BomRootKey::toString() const {
00047     std::ostringstream oStr;
00048     oStr << _id;
00049     return oStr.str();
00050 }
00051
00052 // //////////////////////////////////////
00053 void BomRootKey::serialisationImplementationExport() const {
00054     std::ostringstream oStr;
00055     boost::archive::text_oarchive oa (oStr);
00056     oa << *this;
00057 }
00058
00059 // //////////////////////////////////////
00060 void BomRootKey::serialisationImplementationImport () {
00061     std::istringstream iStr;
00062     boost::archive::text_iarchive ia (iStr);
00063     ia >> *this;
00064 }
00065
00066 // //////////////////////////////////////
00067 template<class Archive>
00068 void BomRootKey::serialize (Archive& ioArchive,
00069                             const unsigned int iFileVersion) {
00070     ioArchive & _id;
00071 }
00072
00073 // //////////////////////////////////////
00074 // Explicit template instantiation
00075 namespace ba = boost::archive;
00076 template void BomRootKey::serialize<ba::text_oarchive> (ba::text_oarchive&,
00077                                                         unsigned int);
00078 template void BomRootKey::serialize<ba::text_iarchive> (ba::text_iarchive&,
00079                                                         unsigned int);
00080 // //////////////////////////////////////
00081
00082 }

```

### 33.221 stdair/bom/BomRootKey.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/bom/KeyAbstract.hpp>
```

#### Classes

- struct [stdair::BomRootKey](#)  
*Key of the BOM structure root.*

#### Namespaces

- [boost](#)  
*Forward declarations.*
- [boost::serialization](#)
- [stdair](#)  
*Handle on the StdAir library context.*

### 33.222 BomRootKey.hpp

```
00001 #ifndef __STDAIR_BOM_BOMROOTKEY_HPP
00002 #define __STDAIR_BOM_BOMROOTKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/bom/KeyAbstract.hpp>
00012
00014 namespace boost {
00015     namespace serialization {
00016         class access;
00017     }
00018 }
00019
00020 namespace stdair {
00021
00025     struct BomRootKey : public KeyAbstract {
00026         friend class boost::serialization::access;
00027
00028         // ////////// Constructors and destructors //////////
00029     public:
00033         BomRootKey ();
00034
00038         BomRootKey (const std::string& iIdentification);
00039
00043         BomRootKey (const BomRootKey&);
00044
00048         ~BomRootKey ();
00049
00050
00051     public:
00052         // ////////// Getters //////////
00056         const std::string& getID() const {
00057             return _id;
00058         }
00059
00060
00061     public:
00062         // ////////// Display support methods //////////
00068         void toStream (std::ostream& ioOut) const;
00069
00075         void fromStream (std::istream& ioIn);
00076
00086         const std::string toString() const;
00087
00088
```

```

00089 public:
00090 // ////////// (Boost) Serialisation support methods //////////
00094 template<class Archive>
00095 void serialize (Archive& ar, const unsigned int iFileVersion);
00096
00097 private:
00102 void serialisationImplementationExport() const;
00103 void serialisationImplementationImport();
00104
00105
00106 private:
00107 // ////////// Attributes //////////
00111 std::string _id;
00112 };
00113
00114 }
00115 #endif // __STDAIR_BOM_BOMROOTKEY_HPP

```

### 33.223 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

- **stdair**  
Handle on the StdAir library context.

### 33.224 BookingClass.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_General.hpp>
00009 #include <stdair/basic/BasConst_Inventory.hpp>
00010 #include <stdair/basic/RandomGeneration.hpp>
00011 #include <stdair/bom/BookingClass.hpp>
00012
00013 namespace stdair {
00014
00015 // //////////////////////////////////////
00016 BookingClass::BookingClass() : _key (DEFAULT_CLASS_CODE), _parent (NULL) {
00017     assert (false);
00018 }
00019
00020 // //////////////////////////////////////
00021 BookingClass::BookingClass (const BookingClass& iBookingClass)
00022 : _key (iBookingClass._key),
00023   _parent (NULL),
00024   _subclassCode (iBookingClass._subclassCode),
00025   _cumulatedProtection (iBookingClass._cumulatedProtection),
00026   _protection (iBookingClass._protection),
00027   _cumulatedBookingLimit (iBookingClass._cumulatedBookingLimit),
00028   _au (iBookingClass._au),
00029   _nego (iBookingClass._nego),
00030   _noShowPercentage (iBookingClass._noShowPercentage),
00031   _cancellationPercentage (iBookingClass._cancellationPercentage),
00032   _nbOfBookings (iBookingClass._nbOfBookings),
00033   _groupNbOfBookings (iBookingClass._groupNbOfBookings),
00034   _groupPendingNbOfBookings (iBookingClass._groupPendingNbOfBookings),
00035   _staffNbOfBookings (iBookingClass._staffNbOfBookings),
00036   _w1NbOfBookings (iBookingClass._w1NbOfBookings),
00037   _nbOfCancellations (iBookingClass._nbOfCancellations),
00038   _etb (iBookingClass._etb),
00039   _netClassAvailability (iBookingClass._netClassAvailability),

```



```

00040     _segmentAvailability (iBookingClass._segmentAvailability),
00041     _netRevenueAvailability (iBookingClass._netRevenueAvailability),
00042     _yield (iBookingClass._yield),
00043     _adjustedYield (iBookingClass._adjustedYield),
00044     _mean (iBookingClass._mean),
00045     _stdDev (iBookingClass._stdDev) {
00046 }
00047
00048 // //////////////////////////////////////
00049 BookingClass::BookingClass (const Key_T& iKey)
00050 : _key (iKey), _parent (NULL), _subclassCode (0), _cumulatedProtection (0.0),
00051   _protection (0.0), _cumulatedBookingLimit (0.0), _au (0.0), _nego (0.0),
00052   _noShowPercentage (0.0), _cancellationPercentage (0.0),
00053   _nbOfBookings (0.0), _groupNbOfBookings (0.0),
00054   _groupPendingNbOfBookings (0.0), _staffNbOfBookings (0.0),
00055   _wlNbOfBookings (0.0), _nbOfCancellations (0.), _etb (0.0),
00056   _netClassAvailability (0.0), _segmentAvailability (0.0),
00057   _netRevenueAvailability (0.0), _yield (0.0), _mean (0.0), _stdDev (0.0) {
00058 }
00059
00060 // //////////////////////////////////////
00061 BookingClass::~BookingClass() {
00062 }
00063
00064 // //////////////////////////////////////
00065 std::string BookingClass::toString() const {
00066     std::ostringstream ostr;
00067     ostr << describeKey();
00068     return ostr.str();
00069 }
00070
00071 // //////////////////////////////////////
00072 void BookingClass::sell (const NbOfBookings_T& iNbOfBookings) {
00073     _nbOfBookings += iNbOfBookings;
00074 }
00075
00076 // //////////////////////////////////////
00077 void BookingClass::cancel (const NbOfBookings_T& iNbOfCancellations) {
00078     _nbOfBookings -= iNbOfCancellations;
00079     _nbOfCancellations += iNbOfCancellations;
00080 }
00081
00082 // //////////////////////////////////////
00083 void BookingClass::generateDemandSamples (const
NbOfSamples_T& K) {
00084     _generatedDemandVector.clear();
00085     if (_stdDev > 0) {
00086         RandomGeneration lGenerator (DEFAULT_RANDOM_SEED);
00087         for (unsigned int i = 0; i < K; ++i) {
00088             RealNumber_T lDemandSample = lGenerator.generateNormal (
00089 _mean, _stdDev);
00090             _generatedDemandVector.push_back (lDemandSample);
00091         }
00092     }
00093 }
00094 // //////////////////////////////////////
00095 void BookingClass::generateDemandSamples (const
NbOfSamples_T& K,
00096                                     const RandomSeed_T& iSeed) {
00097     _generatedDemandVector.clear();
00098     if (_stdDev > 0) {
00099         RandomGeneration lGenerator (iSeed);
00100         for (unsigned int i = 0; i < K; ++i) {
00101             RealNumber_T lDemandSample = lGenerator.generateNormal (
00102 _mean, _stdDev);
00103             _generatedDemandVector.push_back (lDemandSample);
00104         }
00105     }
00106 }
00107 }
00108

```

### 33.225 stdair/bom/BookingClass.hpp File Reference

```
#include <iosfwd>
```

```

#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/stdair_rm_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/BookingClassKey.hpp>
#include <stdair/bom/BookingClassTypes.hpp>

```

## Classes

- class [stdair::BookingClass](#)

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.226 BookingClass.hpp

```

00001 #ifndef __STDAIR_BOM_BOOKINGCLASS_HPP
00002 #define __STDAIR_BOM_BOOKINGCLASS_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/stdair_maths_types.hpp>
00013 #include <stdair/stdair_rm_types.hpp>
00014 #include <stdair/bom/BomAbstract.hpp>
00015 #include <stdair/bom/BookingClassKey.hpp>
00016 #include <stdair/bom/BookingClassTypes.hpp>
00017
00018 namespace stdair {
00019
00024     class BookingClass : public BomAbstract {
00025     public:
00026         template <typename BOM> friend class FacBom;
00027         template <typename BOM> friend class FacCloneBom;
00028         friend class FacBomManager;
00029
00030         // ////////////////////////////////// Type definitions //////////////////////////////////
00032         typedef BookingClassKey Key_T;
00033
00034     public:
00035         // ////////////////////////////////// Getters //////////////////////////////////
00037         const Key_T& getKey() const {
00038             return _key;
00039         }
00040
00042         const ClassCode_T& getClassCode() const {
00043             return _key.getClassCode();
00044         }
00045
00047         BomAbstract* const getParent() const {
00048             return _parent;
00049         }
00050
00052         const HolderMap_T& getHolderMap() const {
00053             return _holderMap;
00054         }
00055
00057         const SubclassCode_T& getSubclassCode() const {
00058             return _subclassCode;
00059         }
00060
00062         const AuthorizationLevel_T& getAuthorizationLevel() const {
00063             return _au;
00064         }
00065

```

```

00067     const ProtectionLevel_T& getProtection() const {
00068         return _protection;
00069     }
00070
00072     const ProtectionLevel_T& getCumulatedProtection() const {
00073         return _cumulatedProtection;
00074     }
00075
00077     const BookingLimit_T& getCumulatedBookingLimit() const {
00078         return _cumulatedBookingLimit;
00079     }
00080
00082     const NbOfSeats_T& getNegotiatedSpace() const {
00083         return _nego;
00084     }
00085
00087     const OverbookingRate_T& getNoShowPercentage() const {
00088         return _noShowPercentage;
00089     }
00090
00092     const OverbookingRate_T& getCancellationPercentage() const {
00093         return _cancellationPercentage;
00094     }
00095
00097     const NbOfBookings_T& getNbOfBookings() const {
00098         return _nbOfBookings;
00099     }
00100
00102     const NbOfBookings_T& getNbOfGroupBookings() const {
00103         return _groupNbOfBookings;
00104     }
00105
00107     const NbOfBookings_T& getNbOfPendingGroupBookings() const {
00108         return _groupPendingNbOfBookings;
00109     }
00110
00112     const NbOfBookings_T& getNbOfStaffBookings() const {
00113         return _staffNbOfBookings;
00114     }
00115
00117     const NbOfBookings_T& getNbOfWLBookings() const {
00118         return _wlNbOfBookings;
00119     }
00120
00122     const NbOfCancellations_T& getNbOfCancellations() const {
00123         return _nbOfCancellations;
00124     }
00125
00127     const NbOfBookings_T& getETB() const {
00128         return _etb;
00129     }
00130
00132     const Availability_T& getNetClassAvailability() const {
00133         return _netClassAvailability;
00134     }
00135
00137     const Availability_T& getSegmentAvailability() const {
00138         return _segmentAvailability;
00139     }
00140
00142     const Availability_T& getNetRevenueAvailability() const {
00143         return _netRevenueAvailability;
00144     }
00145
00147     const Yield_T& getYield () const { return _yield; }
00148     const Yield_T& getAdjustedYield () const { return
    _adjustedYield; }
00149
00151     const MeanValue_T& getMean () const { return _mean; }
00152     const StdDevValue_T& getStdDev () const {return
    _stdDev; }
00153     const MeanValue_T& getPriceDemMean () const { return
    _priceDemMean; }
00154     const StdDevValue_T& getPriceDemStdDev () const { return
    _priceDemStdDev; }
00155     const MeanValue_T& getCumuPriceDemMean () const {
00156         return _cumuPriceDemMean;
00157     }
00158     const StdDevValue_T& getCumuPriceDemStdDev () const {
00159         return _cumuPriceDemStdDev;
00160     }
00161     const MeanValue_T& getProductDemMean () const { return
    _productDemMean; }
00162     const StdDevValue_T& getProductDemStdDev () const {return
    _productDemStdDev; }
00163
00165     const GeneratedDemandVector_T&

```

```

00166     getGeneratedDemandVector () const {
00167         return _generatedDemandVector;
00168     }
00169 public:
00170     // //////////// Setters ////////////
00172     void setCumulatedProtection (const ProtectionLevel_T& iPL) {
00173         _cumulatedProtection = iPL;
00174     }
00175
00177     void setProtection (const ProtectionLevel_T& iPL) {
00178         _protection = iPL;
00179     }
00180
00182     void setCumulatedBookingLimit (const BookingLimit_T& iBL) {
00183         _cumulatedBookingLimit = iBL;
00184     }
00185
00187     void setAuthorizationLevel (const AuthorizationLevel_T& iAU) {
00188         _au = iAU;
00189     }
00190
00192     void setSegmentAvailability (const Availability_T& iAvl) {
00193         _segmentAvailability = iAvl;
00194     }
00195
00197     void setYield (const Yield_T& iYield) {
00198         _yield = iYield;
00199         _adjustedYield = iYield;
00200     }
00201     void setAdjustedYield (const Yield_T& iYield) {
00202         _adjustedYield = iYield; }
00203
00204     void setMean (const MeanValue_T& iMean) { _mean = iMean; }
00205     void setStdDev (const StdDevValue_T& iStdDev) {
00206         _stdDev = iStdDev; }
00207     void setPriceDemMean (const MeanValue_T& iMean) {
00208         _priceDemMean = iMean; }
00209     void setPriceDemStdDev (const StdDevValue_T& iStdDev) {
00210         _priceDemStdDev = iStdDev; }
00211     void setCumuPriceDemMean (const MeanValue_T& iMean) {
00212         _cumuPriceDemMean = iMean; }
00213     void setCumuPriceDemStdDev (const StdDevValue_T& iStdDev) {
00214         _cumuPriceDemStdDev = iStdDev; }
00215     void setProductDemMean (const MeanValue_T& iMean) {
00216         _productDemMean = iMean; }
00217     void setProductDemStdDev (const StdDevValue_T& iStdDev) {
00218         _productDemStdDev = iStdDev; }
00219     void setProductDemStdDev (const StdDevValue_T& iStdDev) {
00220         _productDemStdDev = iStdDev; }
00221
00222 public:
00223     // //////////// Display support methods ////////////
00226     void toStream (std::ostream& ioOut) const {
00227         ioOut << toString();
00228     }
00229
00232     void fromStream (std::istream& ioIn) {
00233     }
00234
00236     std::string toString() const;
00237
00239     const std::string describeKey() const {
00240         return _key.toString();
00241     }
00242
00243 public:
00244     // //////////// Business Methods ////////////
00246     void sell (const NbOfBookings_T&);
00247
00249     void cancel (const NbOfBookings_T&);
00250
00253     void generateDemandSamples (const NbOfSamples_T&);
00254
00257     void generateDemandSamples (const NbOfSamples_T&, const
RandomSeed_T&);
00258
00259 protected:
00260     // //////////// Constructors and destructors ////////////
00262     BookingClass (const Key_T&);
00264     virtual ~BookingClass();
00265
00266 private:
00268     BookingClass();
00270     BookingClass (const BookingClass&);

```

```

00271
00272
00273     protected:
00274         // ////////// Attributes //////////
00276         Key_T _key;
00277
00279         BomAbstract* _parent;
00280
00282         HolderMap_T _holderMap;
00283
00285         SubclassCode_T _subclassCode;
00286
00288         ProtectionLevel_T _cumulatedProtection;
00289
00291         ProtectionLevel_T _protection;
00292
00294         BookingLimit_T _cumulatedBookingLimit;
00295
00297         AuthorizationLevel_T _au;
00298
00300         NbOfSeats_T _nego;
00301
00303         OverbookingRate_T _noShowPercentage;
00304
00306         OverbookingRate_T _cancellationPercentage;
00307
00309         NbOfBookings_T _nbOfBookings;
00310
00312         NbOfBookings_T _groupNbOfBookings;
00313
00315         NbOfBookings_T _groupPendingNbOfBookings;
00316
00318         NbOfBookings_T _staffNbOfBookings;
00319
00321         NbOfBookings_T _wlNbOfBookings;
00322
00324         NbOfCancellations_T _nbOfCancellations;
00325
00327         NbOfBookings_T _etb;
00328
00330         Availability_T _netClassAvailability;
00331
00333         Availability_T _segmentAvailability;
00334
00336         Availability_T _netRevenueAvailability;
00337
00339         Yield_T _yield;
00340         Yield_T _adjustedYield;
00341
00343         MeanValue_T _mean;
00344         StdDevValue_T _stdDev;
00345
00347         MeanValue_T _priceDemMean;
00348         StdDevValue_T _priceDemStdDev;
00349
00351         MeanValue_T _cumuPriceDemMean;
00352         StdDevValue_T _cumuPriceDemStdDev;
00353
00355         MeanValue_T _productDemMean;
00356         StdDevValue_T _productDemStdDev;
00357
00359         GeneratedDemandVector_T _generatedDemandVector;
00360     };
00361
00362 }
00363 #endif // __STDAIR_BOM_BOOKINGCLASS_HPP

```

### 33.227 stdair/bom/BookingClassKey.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BookingClassKey.hpp>

```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.228 BookingClassKey.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_Inventory.hpp>
00009 #include <stdair/bom/BookingClassKey.hpp>
00010
00011 namespace stdair {
00012
00013 // //////////////////////////////////////
00014 BookingClassKey::BookingClassKey() : _classCode (DEFAULT_CLASS_CODE) {
00015     assert (false);
00016 }
00017
00018 // //////////////////////////////////////
00019 BookingClassKey::BookingClassKey (const BookingClassKey& iKey)
00020     : _classCode (iKey._classCode) {
00021 }
00022
00023 // //////////////////////////////////////
00024 BookingClassKey::BookingClassKey (const ClassCode_T& iClassCode)
00025     : _classCode (iClassCode) {
00026 }
00027
00028 // //////////////////////////////////////
00029 BookingClassKey::~BookingClassKey () {
00030 }
00031
00032 // //////////////////////////////////////
00033 void BookingClassKey::toStream (std::ostream& ioOut) const {
00034     ioOut << "BookingClassKey: " << toString();
00035 }
00036
00037 // //////////////////////////////////////
00038 void BookingClassKey::fromStream (std::istream& ioIn) {
00039 }
00040
00041 // //////////////////////////////////////
00042 const std::string BookingClassKey::toString() const {
00043     std::ostringstream oStr;
00044     oStr << _classCode;
00045     return oStr.str();
00046 }
00047
00048 }

```

### 33.229 stdair/bom/BookingClassKey.hpp File Reference

```

#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>

```

#### Classes

- struct [stdair::BookingClassKey](#)

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.230 BookingClassKey.hpp

```

00001 #ifndef __STDAIR_BOM_BOOKINGCLASKEY_HPP

```

```

00002 #define __STDAIR_BOM_BOOKINGCLASKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009 #include <stdair/bom/KeyAbstract.hpp>
00010
00011 namespace stdair {
00012
00016     struct BookingClassKey : public KeyAbstract {
00017
00018         // ////////////////////////////////// Constructors and destructors //////////////////////////////////
00019     private:
00021         BookingClassKey();
00022
00023     public:
00025         BookingClassKey (const ClassCode_T& iClassCode);
00027         BookingClassKey (const BookingClassKey&);
00029         ~BookingClassKey();
00030
00031
00032         // ////////////////////////////////// Getters //////////////////////////////////
00034         const ClassCode_T& getClassCode () const {
00035             return _classCode;
00036         }
00037
00038
00039         // ////////////////////////////////// Display support methods //////////////////////////////////
00042         void toStream (std::ostream& ioOut) const;
00043
00046         void fromStream (std::istream& ioIn);
00047
00053         const std::string toString() const;
00054
00055
00056     private:
00057         // ////////////////////////////////// Attributes //////////////////////////////////
00059         ClassCode_T _classCode;
00060     };
00061
00062 }
00063 #endif // __STDAIR_BOM_BOOKINGCLASKEY_HPP

```

### 33.231 stdair/bom/BookingClassTypes.hpp File Reference

```

#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>

```

#### Namespaces

- [stdair](#)  
*Handle on the StdAir library context.*

#### Typedefs

- typedef std::list< BookingClass \* > [stdair::BookingClassList\\_T](#)
- typedef std::map< const MapKey\_T, BookingClass \* > [stdair::BookingClassMap\\_T](#)

### 33.232 BookingClassTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_BOOKINGCLASSTYPES_HPP
00003 #define __STDAIR_BOM_BOOKINGCLASSTYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL

```

```

00009 #include <map>
00010 #include <list>
00011 // StdAir
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015     // Forward declarations.
00016     class BookingClass;
00017     typedef std::list<BookingClass*> BookingClassList_T;
00018
00019     typedef std::map<const MapKey_T, BookingClass*> BookingClassMap_T;
00020 }
00021 #endif // __STDAIR_BOM_BOOKINGCLASSTYPES_HPP
00022

```

### 33.233 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

- [stdair](#)  
Handle on the StdAir library context.

#### Functions

- void [stdair::intDisplay](#) (std::ostream &oStream, const int &iInt)

### 33.234 BookingRequestStruct.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost
00008 #include <boost/date_time/gregorian/formatters.hpp>
00009 #include <boost/date_time/posix_time/posix_time.hpp>
00010 // StdAir
00011 #include <stdair/basic/BasConst_Inventory.hpp>
00012 #include <stdair/basic/BasConst_Request.hpp>
00013 #include <stdair/bom/BookingRequestStruct.hpp>
00014
00015 namespace stdair {
00016
00017     // //////////////////////////////////////
00018     BookingRequestStruct::BookingRequestStruct ()
00019         : _origin (DEFAULT_ORIGIN), _destination (DEFAULT_DESTINATION),
00020           _pos (DEFAULT_POS),
00021           _preferredDepartureDate (DEFAULT_PREFERRED_DEPARTURE_DATE),
00022           _preferredDepartureTime (DEFAULT_PREFERRED_DEPARTURE_TIME),
00023           _requestDateTime (DEFAULT_REQUEST_DATE_TIME),
00024           _preferredCabin (DEFAULT_PREFERRED_CABIN),
00025           _partySize (DEFAULT_PARTY_SIZE),
00026           _channel (DEFAULT_CHANNEL),
00027           _tripType (TRIP_TYPE_ONE_WAY),
00028           _stayDuration (DEFAULT_STAY_DURATION),
00029           _frequentFlyerType (DEFAULT_FF_TIER),
00030           _wtp (DEFAULT_WTP),
00031           _valueOfTime (DEFAULT_VALUE_OF_TIME),

```



```

00032     _changeFees (false), _changeFeeDisutility (0.0),
00033     _nonRefundable (false), _nonRefundableDisutility (0.0) {
00034     assert (false);
00035 }
00036
00037 // //////////////////////////////////////
00038 BookingRequestStruct::
00039 BookingRequestStruct (const BookingRequestStruct& iBookingRequest)
00040 : _generatorKey (iBookingRequest._generatorKey),
00041   _origin (iBookingRequest._origin),
00042   _destination (iBookingRequest._destination),
00043   _pos (iBookingRequest._pos),
00044   _preferredDepartureDate (iBookingRequest._preferredDepartureDate),
00045   _preferredDepartureTime (iBookingRequest._preferredDepartureTime),
00046   _requestDateTime (iBookingRequest._requestDateTime),
00047   _preferredCabin (iBookingRequest._preferredCabin),
00048   _partySize (iBookingRequest._partySize),
00049   _channel (iBookingRequest._channel),
00050   _tripType (iBookingRequest._tripType),
00051   _stayDuration (iBookingRequest._stayDuration),
00052   _frequentFlyerType (iBookingRequest._frequentFlyerType),
00053   _wtp (iBookingRequest._wtp),
00054   _valueOfTime (iBookingRequest._valueOfTime),
00055   _changeFees (iBookingRequest._changeFees),
00056   _changeFeeDisutility (iBookingRequest._changeFeeDisutility),
00057   _nonRefundable (iBookingRequest._nonRefundable),
00058   _nonRefundableDisutility (iBookingRequest._nonRefundableDisutility) {
00059 }
00060
00061 // //////////////////////////////////////
00062 BookingRequestStruct::
00063 BookingRequestStruct (const DemandGeneratorKey_T& iGeneratorKey,
00064                     const AirportCode_T& iOrigin,
00065                     const AirportCode_T& iDestination,
00066                     const CityCode_T& iPOS,
00067                     const Date_T& iDepartureDate,
00068                     const DateTime_T& iRequestDateTime,
00069                     const CabinCode_T& iPreferredCabin,
00070                     const NbOfSeats_T& iPartySize,
00071                     const ChannelLabel_T& iChannel,
00072                     const TripType_T& iTripType,
00073                     const DayDuration_T& iStayDuration,
00074                     const FrequentFlyer_T& iFrequentFlyerType,
00075                     const Duration_T& iPreferredDepartureTime,
00076                     const WTP_T& iWTP,
00077                     const PriceValue_T& iValueOfTime,
00078                     const ChangeFees_T& iChangeFees,
00079                     const Disutility_T& iChangeFeeDisutility,
00080                     const NonRefundable_T& iNonRefundable,
00081                     const Disutility_T& iNonRefundableDisutility)
00082 : _generatorKey (iGeneratorKey), _origin (iOrigin),
00083   _destination (iDestination), _pos (iPOS),
00084   _preferredDepartureDate (iDepartureDate),
00085   _preferredDepartureTime (iPreferredDepartureTime),
00086   _requestDateTime (iRequestDateTime),
00087   _preferredCabin (iPreferredCabin), _partySize (iPartySize),
00088   _channel (iChannel), _tripType (iTripType),
00089   _stayDuration (iStayDuration), _frequentFlyerType (iFrequentFlyerType),
00090   _wtp (iWTP), _valueOfTime (iValueOfTime),
00091   _changeFees (iChangeFees), _changeFeeDisutility (iChangeFeeDisutility),
00092   _nonRefundable (iNonRefundable),
00093   _nonRefundableDisutility (iNonRefundableDisutility) {
00094 }
00095
00096 // //////////////////////////////////////
00097 BookingRequestStruct::
00098 BookingRequestStruct (const AirportCode_T& iOrigin,
00099                     const AirportCode_T& iDestination,
00100                     const CityCode_T& iPOS,
00101                     const Date_T& iDepartureDate,
00102                     const DateTime_T& iRequestDateTime,
00103                     const CabinCode_T& iPreferredCabin,
00104                     const NbOfSeats_T& iPartySize,
00105                     const ChannelLabel_T& iChannel,
00106                     const TripType_T& iTripType,
00107                     const DayDuration_T& iStayDuration,
00108                     const FrequentFlyer_T& iFrequentFlyerType,
00109                     const Duration_T& iPreferredDepartureTime,
00110                     const WTP_T& iWTP,
00111                     const PriceValue_T& iValueOfTime,
00112                     const ChangeFees_T& iChangeFees,
00113                     const Disutility_T& iChangeFeeDisutility,
00114                     const NonRefundable_T& iNonRefundable,
00115                     const Disutility_T& iNonRefundableDisutility)
00116 : _generatorKey (""), _origin (iOrigin),
00117   _destination (iDestination), _pos (iPOS),
00118   _preferredDepartureDate (iDepartureDate),

```

```

00119     _preferredDepartureTime (iPreferredDepartureTime),
00120     _requestDateTime (iRequestDateTime),
00121     _preferredCabin (iPreferredCabin), _partySize (iPartySize),
00122     _channel (iChannel), _tripType (iTripType),
00123     _stayDuration (iStayDuration), _frequentFlyerType (iFrequentFlyerType),
00124     _wtp (iWTP), _valueOfTime (iValueOfTime),
00125     _changeFees (iChangeFees), _changeFeeDisutility (iChangeFeeDisutility),
00126     _nonRefundable (iNonRefundable),
00127     _nonRefundableDisutility (iNonRefundableDisutility) {
00128 }
00129
00130 // //////////////////////////////////////
00131 BookingRequestStruct::~BookingRequestStruct() {
00132 }
00133
00134 // //////////////////////////////////////
00135 void BookingRequestStruct::toStream (std::ostream& ioOut) const {
00136     ioOut << describe();
00137 }
00138
00139 // //////////////////////////////////////
00140 void BookingRequestStruct::fromStream (std::istream& ioIn) {
00141 }
00142
00143 // //////////////////////////////////////
00144 const std::string BookingRequestStruct::describe() const {
00145     std::ostringstream oStr;
00146     oStr << "At " << _requestDateTime
00147         << ", for (" << _pos << ", " << _channel << ")"
00148         << " " << _origin << "-" << _destination << " (" << _tripType << ")"
00149         << " " << _preferredDepartureDate << " (" << _stayDuration << " days)"
00150         << " " << _preferredDepartureTime
00151         << " " << _preferredCabin << " " << _partySize
00152         << " " << _frequentFlyerType << " " << _wtp << " " << _valueOfTime
00153         << " " << _changeFees << " " << _changeFeeDisutility << " "
00154         << _nonRefundable << " " << _nonRefundableDisutility;
00155     return oStr.str();
00156 }
00157
00158 // //////////////////////////////////////
00159 void intDisplay (std::ostream& oStream, const int& iInt) {
00160     const int dInt = iInt - static_cast<int> (iInt / 100) * 100;
00161     if (dInt < 10) {
00162         oStream << "0" << dInt;
00163     } else {
00164         oStream << dInt;
00165     }
00166 }
00167
00168 // //////////////////////////////////////
00169 const std::string BookingRequestStruct::display() const {
00170     std::ostringstream oStr;
00171
00172     // Request date and time
00173     const Date_T& lRequestDate = _requestDateTime.date();
00174     oStr << boost::gregorian::to_iso_extended_string (lRequestDate);
00175
00176     const Duration_T& lRequestTime = _requestDateTime.time_of_day();
00177     oStr << ", " << boost::posix_time::to_simple_string (lRequestTime);
00178
00179     // POS
00180     oStr << ", " << _pos;
00181
00182     // Channel
00183     oStr << ", " << _channel;
00184
00185     // Origin
00186     oStr << ", " << _origin;
00187
00188     // Destination
00189     oStr << ", " << _destination;
00190
00191     // Preferred departure date
00192     oStr << ", "
00193         << boost::gregorian::to_iso_extended_string (_preferredDepartureDate);
00194
00195     // Preferred departure time
00196     oStr << ", "
00197         << boost::posix_time::to_simple_string (_preferredDepartureTime);
00198
00199     // MIN & MAX preferred departure time (hardcode)
00200     oStr << ", " << "00:00-23:59";
00201
00202     // Preferred arrival date (hardcode to the preferred departure date)
00203     oStr << ", "
00204         << boost::gregorian::to_iso_extended_string (_preferredDepartureDate);
00205

```

```

00206 // Preferred arrival time (hard-coded to 23:55)
00207 oStr << ", " << "23:55";
00208
00209 // Preferred cabin
00210 oStr << ", " << _preferredCabin;
00211
00212 // Trip type
00213 oStr << ", " << _tripType;
00214
00215 // Duration of stay
00216 oStr << ", ";
00217 if (_tripType == TRIP_TYPE_ONE_WAY) {
00218     oStr << "0";
00219 } else {
00220     oStr << _stayDuration;
00221 }
00222
00223 // Frequent flyer tier
00224 oStr << ", " << _frequentFlyerType;
00225
00226 // Willingness-to-pay
00227 oStr << ", " << _wtp;
00228
00229 // Disutility per stop (hardcode to 100, expressed as a monetary
00230 // unit per hour)
00231 oStr << ", " << "100";
00232
00233 // Value of time
00234 oStr << ", " << _valueOfTime;
00235
00236 // Change fees
00237 oStr << ", " << _changeFees;
00238
00239 // Change fee disutility
00240 oStr << ", " << _changeFeeDisutility;
00241
00242 // Non refundable
00243 oStr << ", " << _nonRefundable;
00244
00245 // Non refundable disutility
00246 oStr << ", " << _nonRefundableDisutility;
00247
00248 return oStr.str();
00249 }
00250
00251 }

```

### 33.235 stdair/bom/BookingRequestStruct.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/BookingRequestTypes.hpp>

```

#### Classes

- struct [stdair::BookingRequestStruct](#)  
*Structure holding the elements of a booking request.*

#### Namespaces

- [stdair](#)  
*Handle on the StdAir library context.*

### 33.236 BookingRequestStruct.hpp

```
00001 #ifndef __STDAIR_BOM_BOOKINGREQUESTSTRUCT_HPP
```

```

00002 #define __STDAIR_BOM_BOOKINGREQUESTSTRUCT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/stdair_demand_types.hpp>
00013 #include <stdair/basic/StructAbstract.hpp>
00014 #include <stdair/bom/BookingRequestTypes.hpp>
00015
00016 namespace stdair {
00017
00021 struct BookingRequestStruct : public StructAbstract {
00022 public:
00023 // ////////////////////////////////// Getters //////////////////////////////////
00025 const DemandGeneratorKey_T& getDemandGeneratorKey () const {
00026     return _generatorKey;
00027 }
00028
00030 const AirportCode_T& getOrigin() const {
00031     return _origin;
00032 }
00033
00035 const AirportCode_T& getDestination() const {
00036     return _destination;
00037 }
00038
00040 const CityCode_T& getPOS() const {
00041     return _pos;
00042 }
00043
00045 const Date_T& getPreferredDepartureDate() const {
00046     return _preferredDepartureDate;
00047 }
00048
00050 const Duration_T& getPreferredDepartureTime() const {
00051     return _preferredDepartureTime;
00052 }
00053
00055 const DateTime_T& getRequestDateTime() const {
00056     return _requestDateTime;
00057 }
00058
00060 const CabinCode_T& getPreferredCabin() const {
00061     return _preferredCabin;
00062 }
00063
00065 const NbOfSeats_T& getPartySize() const {
00066     return _partySize;
00067 }
00068
00070 const ChannelLabel_T& getBookingChannel() const {
00071     return _channel;
00072 }
00073
00075 const TripType_T& getTripType() const {
00076     return _tripType;
00077 }
00078
00080 const DayDuration_T& getStayDuration() const {
00081     return _stayDuration;
00082 }
00083
00085 const FrequentFlyer_T& getFrequentFlyerType() const {
00086     return _frequentFlyerType;
00087 }
00088
00090 const WTP_T& getWTP() const {
00091     return _wtp;
00092 }
00093
00095 const PriceValue_T& getValueOfTime () const {
00096     return _valueOfTime;
00097 }
00098
00100 const ChangeFees_T& getChangeFees () const {
00101     return _changeFees;
00102 }
00103
00105 const Disutility_T& getChangeFeeDisutility () const {
00106     return _changeFeeDisutility;
00107 }
00108

```

```

00110     const NonRefundable_T& getNonRefundable () const {
00111         return _nonRefundable;
00112     }
00113
00115     const Disutility_T& getNonRefundableDisutility () const {
00116         return _nonRefundableDisutility;
00117     }
00118
00119
00120 public:
00121     // /////////// Display support method ///////////
00122     void toStream (std::ostream& ioOut) const;
00123
00124     void fromStream (std::istream& ioIn);
00125
00126     const std::string describe() const;
00127
00128     const std::string display() const;
00129
00130 public:
00131     // /////////// Constructors and Destructors ///////////
00132     BookingRequestStruct (const DemandGeneratorKey_T& iGeneratorKey
00133
00134
00135         const AirportCode_T& iOrigin,
00136         const AirportCode_T& iDestination,
00137         const CityCode_T& iPOS,
00138         const Date_T& iDepartureDate,
00139         const DateTime_T& iRequestDateTime,
00140         const CabinCode_T& iPreferredCabin,
00141         const NbOfSeats_T& iPartySize,
00142         const ChannelLabel_T& iChannel,
00143         const TripType_T& iTripType,
00144         const DayDuration_T& iStayDuration,
00145         const FrequentFlyer_T& iFrequentFlyerType,
00146         const Duration_T& iPreferredDepartureTime,
00147         const WTP_T& iWTP,
00148         const PriceValue_T& iValueOfTime,
00149         const ChangeFees_T& iChangeFees,
00150         const Disutility_T& iChangeFeeDisutility,
00151         const NonRefundable_T& iNonRefundable,
00152         const Disutility_T& iNonRefundableDisutility);
00153
00154     BookingRequestStruct (const AirportCode_T& iOrigin,
00155         const AirportCode_T& iDestination,
00156         const CityCode_T& iPOS,
00157         const Date_T& iDepartureDate,
00158         const DateTime_T& iRequestDateTime,
00159         const CabinCode_T& iPreferredCabin,
00160         const NbOfSeats_T& iPartySize,
00161         const ChannelLabel_T& iChannel,
00162         const TripType_T& iTripType,
00163         const DayDuration_T& iStayDuration,
00164         const FrequentFlyer_T& iFrequentFlyerType,
00165         const Duration_T& iPreferredDepartureTime,
00166         const WTP_T& iWTP,
00167         const PriceValue_T& iValueOfTime,
00168         const ChangeFees_T& iChangeFees,
00169         const Disutility_T& iChangeFeeDisutility,
00170         const NonRefundable_T& iNonRefundable,
00171         const Disutility_T& iNonRefundableDisutility);
00172
00173     BookingRequestStruct (const BookingRequestStruct&);
00174
00175     ~BookingRequestStruct ();
00176
00177 private:
00178     BookingRequestStruct ();
00179
00180 private:
00181     // /////////// Attributes ///////////
00182     const DemandGeneratorKey_T _generatorKey;
00183
00184     const AirportCode_T _origin;
00185
00186     const AirportCode_T _destination;
00187
00188     const CityCode_T _pos;
00189
00190     const Date_T _preferredDepartureDate;
00191
00192     const Duration_T _preferredDepartureTime;
00193
00194     const DateTime_T _requestDateTime;
00195
00196     const CabinCode_T _preferredCabin;

```

```

00283
00285     const NbOfSeats_T _partySize;
00286
00288     const ChannelLabel_T _channel;
00289
00292     const TripType_T _tripType;
00293
00295     const DayDuration_T _stayDuration;
00296
00298     const FrequentFlyer_T _frequentFlyerType;
00299
00301     const WTP_T _wtp;
00302
00304     const PriceValue_T _valueOfTime;
00305
00307     const ChangeFees_T _changeFees;
00308
00310     const Disutility_T _changeFeeDisutility;
00311
00313     const NonRefundable_T _nonRefundable;
00314
00316     const Disutility_T _nonRefundableDisutility;
00317 };
00318
00319 }
00320 #endif // __STDAIR_BOM_BOOKINGREQUESTSTRUCT_HPP

```

### 33.237 stdair/bom/BookingRequestTypes.hpp File Reference

```
#include <boost/shared_ptr.hpp>
```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

#### Typedefs

- typedef boost::shared\_ptr< BookingRequestStruct > [stdair::BookingRequestPtr\\_T](#)
- typedef std::string [stdair::DemandGeneratorKey\\_T](#)

### 33.238 BookingRequestTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_BOOKINGREQUESTTYPES_HPP
00003 #define __STDAIR_BOM_BOOKINGREQUESTTYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // Boost
00009 #include <boost/shared_ptr.hpp>
00010
00011 namespace stdair {
00012
00013     // Forward declarations
00014     struct BookingRequestStruct;
00015
00016     // ////////////////////////////////// Type definitions //////////////////////////////////
00018     typedef boost::shared_ptr<BookingRequestStruct> BookingRequestPtr_T;
00019
00021     typedef std::string DemandGeneratorKey_T;
00022
00023
00024 }
00025 #endif // __STDAIR_BOM_BOOKINGREQUESTTYPES_HPP
00026

```

### 33.239 stdair/bom/BreakPointStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/BreakPointStruct.hpp>
```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.240 BreakPointStruct.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_General.hpp>
00009 #include <stdair/bom/BreakPointStruct.hpp>
00010
00011 namespace stdair {
00012
00013 // //////////////////////////////////////
00014 BreakPointStruct::BreakPointStruct() {
00015     assert (false);
00016 }
00017
00018 // //////////////////////////////////////
00019 BreakPointStruct::
00020 BreakPointStruct (const BreakPointStruct& iBreakPoint)
00021     : _breakPointTime (iBreakPoint._breakPointTime) {
00022 }
00023
00024 // //////////////////////////////////////
00025 BreakPointStruct::
00026 BreakPointStruct (const DateTime_T& iBreakPointTime)
00027     : _breakPointTime (iBreakPointTime) {
00028 }
00029
00030 // //////////////////////////////////////
00031 BreakPointStruct::
00032 BreakPointStruct (const Date_T& iBreakPointDate)
00033     : _breakPointTime (iBreakPointDate, DEFAULT_NULL_DURATION) {
00034 }
00035
00036 // //////////////////////////////////////
00037 BreakPointStruct::~BreakPointStruct () {
00038 }
00039
00040 // //////////////////////////////////////
00041 void BreakPointStruct::toStream (std::ostream& ioOut) const {
00042     ioOut << describe();
00043 }
00044
00045 // //////////////////////////////////////
00046 void BreakPointStruct::fromStream (std::istream& ioIn) {
00047 }
00048
00049 // //////////////////////////////////////
00050 const std::string BreakPointStruct::describe() const {
00051     std::ostringstream oStr;
00052     oStr << _breakPointTime;
00053     return oStr.str();
00054 }
00055
00056 }
```

## 33.241 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>
```

## Classes

- struct [stdair::BreakPointStruct](#)

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.242 BreakPointStruct.hpp

```
00001 #ifndef __STDAIR_BOM_BREAKPOINTSTRUCT_HPP
00002 #define __STDAIR_BOM_BREAKPOINTSTRUCT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_date_time_types.hpp>
00012 #include <stdair/basic/StructAbstract.hpp>
00013 #include <stdair/bom/BreakPointTypes.hpp>
00014
00015 namespace stdair {
00016
00017     struct BreakPointStruct : public StructAbstract {
00018     public:
00019         // ////////////////////////////////////// Getters //////////////////////////////////////
00020         const DateTime_T& getBreakPointTime() const {
00021             return _breakPointTime;
00022         }
00023
00024         // ////////////////////////////////////// Display support method //////////////////////////////////////
00025         void toStream (std::ostream& ioOut) const;
00026
00027         void fromStream (std::istream& ioIn);
00028
00029         const std::string describe() const;
00030
00031         // ////////////////////////////////////// Constructors and Destructors //////////////////////////////////////
00032     public:
00033         BreakPointStruct (const DateTime_T&);
00034
00035         BreakPointStruct (const Date_T&);
00036
00037         BreakPointStruct (const BreakPointStruct&);
00038
00039     private:
00040         BreakPointStruct ();
00041
00042     public:
00043         ~BreakPointStruct ();
00044
00045     private:
00046         // ////////////////////////////////////// Attributes //////////////////////////////////////
00047         const DateTime_T _breakPointTime;
00048     };
00049
00050 }
00051 #endif // __STDAIR_BOM_BREAKPOINTSTRUCT_HPP
```



### 33.243 stdair/bom/BreakPointTypes.hpp File Reference

```
#include <list>
#include <boost/shared_ptr.hpp>
```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

#### Typedefs

- typedef boost::shared\_ptr< BreakPointStruct > [stdair::BreakPointPtr\\_T](#)
- typedef std::list< BreakPointStruct > [stdair::BreakPointList\\_T](#)

### 33.244 BreakPointTypes.hpp

```
00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_BREAKPOINTTYPES_HPP
00003 #define __STDAIR_BOM_BREAKPOINTTYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <list>
00010 // Boost
00011 #include <boost/shared_ptr.hpp>
00012
00013 namespace stdair {
00014
00015     // Forward declarations
00016     struct BreakPointStruct;
00017
00018     // ////////////////////////////////// Type definitions //////////////////////////////////
00020     typedef boost::shared_ptr<BreakPointStruct> BreakPointPtr_T;
00021
00023     typedef std::list<BreakPointStruct> BreakPointList_T;
00024
00025 }
00026 #endif // __STDAIR_BOM_BREAKPOINTTYPES_HPP
00027
```

### 33.245 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

- [stdair](#)

*Handle on the StdAir library context.*

## 33.246 Bucket.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/access.hpp>
00011 // StdAir
00012 #include <stdair/basic/BasConst_Inventory.hpp>
00013 #include <stdair/bom/Bucket.hpp>
00014
00015 namespace stdair {
00016
00017 // //////////////////////////////////////
00018 Bucket::Bucket()
00019 : _key (DEFAULT_SEAT_INDEX), _parent (NULL) {
00020     assert (false);
00021 }
00022
00023 // //////////////////////////////////////
00024 Bucket::Bucket (const Bucket& iBucket) :
00025     _key (iBucket._key),
00026     _parent (NULL),
00027     _yieldRangeUpperValue (iBucket._yieldRangeUpperValue),
00028     _availability (iBucket._availability),
00029     _soldSeats (iBucket._soldSeats) {
00030 }
00031
00032
00033 // //////////////////////////////////////
00034 Bucket::Bucket (const Key_T& iKey) : _key (iKey), _parent (NULL) {
00035 }
00036
00037 // //////////////////////////////////////
00038 Bucket::~Bucket () {
00039 }
00040
00041 // //////////////////////////////////////
00042 std::string Bucket::toString() const {
00043     std::ostringstream oStr;
00044     oStr << describeKey();
00045     return oStr.str();
00046 }
00047
00048 // //////////////////////////////////////
00049 void Bucket::serialisationImplementationExport() const {
00050     std::ostringstream oStr;
00051     boost::archive::text_oarchive oa (oStr);
00052     oa << *this;
00053 }
00054
00055 // //////////////////////////////////////
00056 void Bucket::serialisationImplementationImport() {
00057     std::istringstream iStr;
00058     boost::archive::text_iarchive ia (iStr);
00059     ia >> *this;
00060 }
00061
00062 // //////////////////////////////////////
00063 template<class Archive>
00064 void Bucket::serialize (Archive& ioArchive, const unsigned int iFileVersion) {
00065     ioArchive & _key;
00066 }
00067
00068 }
00069

```

## 33.247 stdair/bom/Bucket.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/BucketKey.hpp>
#include <stdair/bom/BucketTypes.hpp>

```

## Classes

- class `stdair::Bucket`  
*Class representing the actual attributes for an airline booking class.*

## Namespaces

- `boost`  
*Forward declarations.*
- `boost::serialization`
- `stdair`  
*Handle on the StdAir library context.*

### 33.248 Bucket.hpp

```

00001 #ifndef __STDAIR_BOM_BUCKET_HPP
00002 #define __STDAIR_BOM_BUCKET_HPP
00003
00004 // ////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/bom/BomAbstract.hpp>
00013 #include <stdair/bom/BucketKey.hpp>
00014 #include <stdair/bom/BucketTypes.hpp>
00015
00016 namespace boost {
00017     namespace serialization {
00018         class access;
00019     }
00020 }
00021
00022 namespace stdair {
00023     class Bucket : public BomAbstract {
00024     public:
00025         template <typename BOM> friend class FacBom;
00026         template <typename BOM> friend class FacCloneBom;
00027         friend class FacBomManager;
00028         friend class boost::serialization::access;
00029
00030         // ////////////////////////////////// Type definitions //////////////////////////////////
00031         typedef BucketKey Key_T;
00032
00033     public:
00034         // ////////////////////////////////// Getters //////////////////////////////////
00035         const Key_T& getKey() const {
00036             return _key;
00037         }
00038
00039         BomAbstract* const getParent() const {
00040             return _parent;
00041         }
00042
00043         const HolderMap_T& getHolderMap() const {
00044             return _holderMap;
00045         }
00046
00047         const SeatIndex_T& getSeatIndex() const {
00048             return _key.getSeatIndex();
00049         }
00050
00051         const Yield_T& getYieldRangeUpperValue() const {
00052             return _yieldRangeUpperValue;
00053         }
00054
00055         const CabinCapacity_T& getAvailability() const {
00056             return _availability;
00057         }
00058     };
00059
00060 }
00061
00062 
```

```

00077
00079     const NbOfSeats_T& getSoldSeats() const {
00080         return _soldSeats;
00081     }
00082
00083
00084     // /////////// Setters ///////////
00086     void setYieldRangeUpperValue (const Yield_T& iYield) {
00087         _yieldRangeUpperValue = iYield;
00088     }
00089
00091     void setAvailability (const CabinCapacity_T& iAvl) {
00092         _availability = iAvl;
00093     }
00094
00096     void setSoldSeats (const NbOfSeats_T& iSoldSeats) {
00097         _soldSeats = iSoldSeats;
00098     }
00099
00100
00101 public:
00102     // /////////// Display support methods ///////////
00108     void toStream (std::ostream& ioOut) const {
00109         ioOut << toString();
00110     }
00111
00117     void fromStream (std::istream& ioIn) {
00118     }
00119
00123     std::string toString() const;
00124
00128     const std::string describeKey() const {
00129         return _key.toString();
00130     }
00131
00132
00133 public:
00134     // /////////// (Boost) Serialisation support methods ///////////
00138     template<class Archive>
00139     void serialize (Archive& ar, const unsigned int iFileVersion);
00140
00141 private:
00146     void serialisationImplementationExport() const;
00147     void serialisationImplementationImport();
00148
00149
00150 protected:
00151     // /////////// Constructors and destructors ///////////
00155     Bucket (const Key_T&);
00156
00160     virtual ~Bucket();
00161
00162 private:
00166     Bucket();
00167
00171     Bucket (const Bucket&);
00172
00173
00174 protected:
00175     // ////////////////////////////////// Children //////////////////////////////////
00179     Key_T _key;
00180
00184     BomAbstract* _parent;
00185
00189     HolderMap_T _holderMap;
00190
00191
00192 protected:
00193     // ////////////////////////////////// Attributes //////////////////////////////////
00197     Yield_T _yieldRangeUpperValue;
00198
00202     CabinCapacity_T _availability;
00203
00207     NbOfSeats_T _soldSeats;
00208 };
00209
00210 }
00211 #endif // __STDAIR_BOM_BUCKET_HPP
00212

```

## 33.249 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

- [stdair](#)

*Handle on the StdAir library context.*

## Functions

- template void [stdair::BucketKey::serialize< ba::text\\_oarchive >](#) (ba::text\_oarchive &, unsigned int)
- template void [stdair::BucketKey::serialize< ba::text\\_iarchive >](#) (ba::text\_iarchive &, unsigned int)

### 33.250 BucketKey.cpp

```
00001 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00002 // Import section
00003 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/access.hpp>
00011 // StdAir
00012 #include <stdair/bom/BucketKey.hpp>
00013
00014 namespace stdair {
00015
00016 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00017 BucketKey::BucketKey() {
00018     assert (false);
00019 }
00020
00021 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00022 BucketKey::BucketKey (const SeatIndex_T& iSeatIndex)
00023 : _seatIndex (iSeatIndex) {
00024 }
00025
00026 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00027 BucketKey::BucketKey (const BucketKey& iBucketKey)
00028 : _seatIndex (iBucketKey._seatIndex) {
00029 }
00030
00031 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00032 BucketKey::~~BucketKey() {
00033 }
00034
00035 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00036 void BucketKey::toStream (std::ostream& ioOut) const {
00037     ioOut << "BucketKey: " << toString() << std::endl;
00038 }
00039
00040 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00041 void BucketKey::fromStream (std::istream& ioIn) {
00042 }
00043
00044 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00045 const std::string BucketKey::toString() const {
00046     std::ostringstream oStr;
00047     oStr << _seatIndex;
00048     return oStr.str();
00049 }
00050
00051 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00052 void BucketKey::serialisationImplementationExport() const {
00053     std::ostringstream oStr;
00054     boost::archive::text_oarchive oa (oStr);
00055     oa << *this;
```

```

00056     }
00057
00058     // //////////////////////////////////////
00059     void BucketKey::serialisationImplementationImport() {
00060         std::istringstream iStr;
00061         boost::archive::text_iarchive ia (iStr);
00062         ia >> *this;
00063     }
00064
00065     // //////////////////////////////////////
00066     template<class Archive>
00067     void BucketKey::serialize (Archive& ioArchive,
00068                               const unsigned int iFileVersion) {
00069         ioArchive & _seatIndex;
00070     }
00071
00072     // //////////////////////////////////////
00073     // Explicit template instantiation
00074     namespace ba = boost::archive;
00075     template void BucketKey::serialize<ba::text_oarchive> (ba::text_oarchive&,
00076                                                           unsigned int);
00077     template void BucketKey::serialize<ba::text_iarchive> (ba::text_iarchive&,
00078                                                           unsigned int);
00079     // //////////////////////////////////////
00080
00081 }

```

### 33.251 stdair/bom/BucketKey.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>

```

#### Classes

- struct [stdair::BucketKey](#)  
*Key of booking-class.*

#### Namespaces

- [boost](#)  
*Forward declarations.*
- [boost::serialization](#)
- [stdair](#)  
*Handle on the StdAir library context.*

### 33.252 BucketKey.hpp

```

00001 #ifndef __STDAIR_BOM_BUCKETKEY_HPP
00002 #define __STDAIR_BOM_BUCKETKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/bom/KeyAbstract.hpp>
00013
00015 namespace boost {
00016     namespace serialization {
00017         class access;
00018     }
00019 }
00020

```

```

00021 namespace stdair {
00022
00026     struct BucketKey : public KeyAbstract {
00027         friend class boost::serialization::access;
00028
00029         // ////////// Constructors and destructors //////////
00030     private:
00034         BucketKey();
00035
00036     public:
00040         BucketKey (const SeatIndex_T&);
00044         BucketKey (const BucketKey&);
00048         ~BucketKey();
00049
00050
00051     public:
00052         // ////////// Getters //////////
00054         const SeatIndex_T& getSeatIndex() const {
00055             return _seatIndex;
00056         }
00057
00058
00059     public:
00060         // ////////// Display support methods //////////
00066         void toStream (std::ostream& ioOut) const;
00067
00073         void fromStream (std::istream& ioIn);
00074
00084         const std::string toString() const;
00085
00086
00087     public:
00088         // ////////// (Boost) Serialisation support methods //////////
00092         template<class Archive>
00093         void serialize (Archive& ar, const unsigned int iFileVersion);
00094
00095     private:
00100         void serialisationImplementationExport() const;
00101         void serialisationImplementationImport();
00102
00103
00104     private:
00105         // ////////// Attributes //////////
00109         SeatIndex_T _seatIndex;
00110     };
00111
00112 }
00113 #endif // __STDAIR_BOM_BUCKETKEY_HPP

```

### 33.253 stdair/bom/BucketTypes.hpp File Reference

```

#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>

```

#### Namespaces

- [stdair](#)  
Handle on the StdAir library context.

#### Typedefs

- typedef std::list< Bucket \* > [stdair::BucketList\\_T](#)
- typedef std::map< const MapKey\_T, Bucket \* > [stdair::BucketMap\\_T](#)

### 33.254 BucketTypes.hpp

```

00001 // //////////////////////////////////////////
00002 #ifndef __STDAIR_BOM_BUCKETTYPES_HPP
00003 #define __STDAIR_BOM_BUCKETTYPES_HPP
00004

```

```

00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // StdAir
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016     // Forward declarations
00017     class Bucket;
00018
00020     typedef std::list<Bucket*> BucketList_T;
00021
00023     typedef std::map<const MapKey_T, Bucket*> BucketMap_T;
00024
00025 }
00026 #endif // __STDAIR_BOM_BUCKETTYPES_HPP
00027

```

### 33.255 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

- [stdair](#)

*Handle on the StdAir library context.*

### 33.256 CancellationStruct.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_BookingClass.hpp>
00009 #include <stdair/bom/CancellationStruct.hpp>
00010 #include <stdair/bom/BookingClass.hpp>
00011
00012 namespace stdair {
00013     // //////////////////////////////////////
00014     CancellationStruct::CancellationStruct (const
00015         SegmentPath_T& iSegPath,
00016                                     const ClassList_String_T& iList,
00017                                     const PartySize_T& iSize,
00018                                     const DateTime_T& iDateTime)
00019         : _segmentPath (iSegPath), _classList (iList), _partySize (iSize),
00020           _datetime (iDateTime) {
00021     }
00022
00023     CancellationStruct::CancellationStruct (const
00024         SegmentPath_T& iSegPath,
00025                                     const BookingClassIDList_T& iIDList,
00026                                     const PartySize_T& iSize,
00027                                     const DateTime_T& iDateTime)
00028         : _segmentPath (iSegPath), _classList (iIDList), _partySize (iSize),
00029           _datetime (iDateTime) {
00030     }
00031
00032     CancellationStruct::~CancellationStruct () {
00033     }
00034

```



```

00035 // //////////////////////////////////////
00036 void CancellationStruct::toStream (std::ostream& ioOut) const {
00037     ioOut << describe();
00038 }
00039
00040 // //////////////////////////////////////
00041 void CancellationStruct::fromStream (std::istream& ioIn) {
00042 }
00043
00044 // //////////////////////////////////////
00045 const std::string CancellationStruct::describe() const {
00046     std::ostringstream ostr;
00047
00048     ostr << "Segment path: ";
00049     unsigned short idx = 0;
00050     for (SegmentPath_T::const_iterator lItSegmentPath = _segmentPath.begin();
00051          lItSegmentPath != _segmentPath.end(); ++lItSegmentPath, ++idx) {
00052         if (idx != 0) {
00053             ostr << "-";
00054         }
00055         const std::string& lSegmentKey = *lItSegmentPath;
00056         ostr << lSegmentKey;
00057     }
00058     if (_classList == "") {
00059         ostr << ";";
00060         BookingClassIDList_T::const_iterator lItBookingClassIDList =
00061             _classList.begin();
00062         idx = 0;
00063         for (; lItBookingClassIDList != _classList.end();
00064              ++lItBookingClassIDList, ++idx) {
00065             if (idx != 0) {
00066                 ostr << "-";
00067             }
00068             const BookingClassID_T& lBookingClassID = *lItBookingClassIDList;
00069             const BookingClass& lBookingClass = lBookingClassID.
getObject();
00070             const ClassCode_T& lClassCode = lBookingClass.getClassCode();
00071             ostr << lClassCode;
00072         }
00073         ostr << ";" << _partySize << ";" << _datetime;
00074     } else {
00075         ostr << ";" << _classList << ";" << _partySize << ";" << _datetime;
00076     }
00077     return ostr.str();
00078 }
00079
00080 // //////////////////////////////////////
00081 const std::string CancellationStruct::display() const {
00082     std::ostringstream ostr;
00083
00084     // List of segment keys (one per segment)
00085     unsigned short idx = 0;
00086     for (SegmentPath_T::const_iterator itSegPath = _segmentPath.begin();
00087          itSegPath != _segmentPath.end(); ++itSegPath, ++idx) {
00088         if (idx != 0) {
00089             ostr << " ";
00090         }
00091         const std::string& lSegmentKey = *itSegPath;
00092         ostr << "[" << idx << " ] " << lSegmentKey;
00093     }
00094     if (_classList == "") {
00095         ostr << ";";
00096         BookingClassIDList_T::const_iterator lItBookingClassIDList =
00097             _classList.begin();
00098         idx = 0;
00099         for (; lItBookingClassIDList != _classList.end();
00100              ++lItBookingClassIDList, ++idx) {
00101             if (idx != 0) {
00102                 ostr << "-";
00103             }
00104             const BookingClassID_T& lBookingClassID = *lItBookingClassIDList;
00105             const BookingClass& lBookingClass = lBookingClassID.
getObject();
00106             const ClassCode_T& lClassCode = lBookingClass.getClassCode();
00107             ostr << lClassCode;
00108         }
00109         ostr << ";" << _partySize << ";" << _datetime;
00110     } else {
00111         ostr << ";" << _classList << ";" << _partySize << ";" << _datetime;
00112     }
00113     return ostr.str();
00114 }
00115 }

```

### 33.257 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>
```

#### Classes

- struct [stdair::CancellationStruct](#)  
*Structure holding the elements of a travel solution.*

#### Namespaces

- [stdair](#)  
*Handle on the StdAir library context.*

### 33.258 CancellationStruct.hpp

```
00001 #ifndef __STDAIR_BOM_CANCELLATIONSTRUCT_HPP
00002 #define __STDAIR_BOM_CANCELLATIONSTRUCT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 #include <vector>
00011 // StdAir
00012 #include <stdair/stdair_basic_types.hpp>
00013 #include <stdair/basic/StructAbstract.hpp>
00014 #include <stdair/bom/BookingClassTypes.hpp>
00015 #include <stdair/bom/TravelSolutionTypes.hpp>
00016 #include <stdair/bom/BomIDTypes.hpp>
00017
00018 namespace stdair {
00019
00023 struct CancellationStruct : public StructAbstract {
00024 public:
00025 // //////////// Getters ////////////
00027 const SegmentPath_T& getSegmentPath() const {
00028     return _segmentPath;
00029 }
00030
00032 const ClassList_String_T& getClassList() const {
00033     return _classList;
00034 }
00035
00037 const BookingClassIDList_T& getClassIDList() const {
00038     return _classIDList;
00039 }
00040
00042 const PartySize_T& getPartySize() const {
00043     return _partySize;
00044 }
00045
00047 const DateTime_T& getCancellationDateTime() const {
00048     return _datetime;
00049 }
00050
00051 public:
00052 // //////////// Display support method ////////////
00058 void toStream (std::ostream& ioOut) const;
00059
00064 void fromStream (std::istream& ioIn);
```

```

00065
00069     const std::string describe() const;
00070
00074     const std::string display() const;
00075
00076
00077 public:
00078     // ////////// Constructors & Destructor //////////
00082     CancellationStruct (const SegmentPath_T&, const
ClassList_String_T&,
00083                         const PartySize_T&, const DateTime_T&);
00084
00088     CancellationStruct (const SegmentPath_T&, const
BookingClassIDList_T&,
00089                         const PartySize_T&, const DateTime_T&);
00090
00094     ~CancellationStruct ();
00095
00096
00097 private:
00098     // ////////// Attributes //////////
00102     SegmentPath_T _segmentPath;
00103
00107     ClassList_String_T _classList;
00108
00112     BookingClassIDList_T _classList;
00113
00117     PartySize_T _partySize;
00118
00122     DateTime_T _datetime;
00123 };
00124
00125 }
00126 #endif // __STDAIR_BOM_CANCELLATIONSTRUCT_HPP

```

### 33.259 stdair/bom/CancellationTypes.hpp File Reference

```
#include <boost/shared_ptr.hpp>
```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

#### Typedefs

- typedef boost::shared\_ptr< CancellationStruct > [stdair::CancellationPtr\\_T](#)

### 33.260 CancellationTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_CANCELLATIONTYPES_HPP
00003 #define __STDAIR_BOM_CANCELLATIONTYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // Boost
00009 #include <boost/shared_ptr.hpp>
00010
00011 namespace stdair {
00012
00013     // Forward declarations
00014     struct CancellationStruct;
00015
00016     // ////////// Type definitions //////////
00018     typedef boost::shared_ptr<CancellationStruct> CancellationPtr_T;
00019
00020 }
00021 #endif // __STDAIR_BOM_CANCELLATIONTYPES_HPP
00022

```

## 33.261 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

- [stdair](#)

*Handle on the StdAir library context.*

## 33.262 ConfigHolderStruct.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 #if BOOST_VERSION >= 104100
00008 #include <boost/property_tree/ptree.hpp>
00009 #include <boost/property_tree/json_parser.hpp>
00010 #include <boost/foreach.hpp>
00011 #endif // BOOST_VERSION >= 104100
00012 // StdAir
00013 #include <stdair/stdair_exceptions.hpp>
00014 #include <stdair/basic/ForecastingMethod.hpp>
00015 #include <stdair/basic/UnconstrainingMethod.hpp>
00016 #include <stdair/basic/PartnershipTechnique.hpp>
00017 #include <stdair/basic/PreOptimisationMethod.hpp>
00018 #include <stdair/basic/OptimisationMethod.hpp>
00019 #include <stdair/bom/AirlineFeature.hpp>
00020 #include <stdair/bom/ConfigHolderStruct.hpp>
00021 #include <stdair/bom/BomRetriever.hpp>
00022 #include <stdair/service/Logger.hpp>
00023
00024 namespace stdair {
00025
00026 // //////////////////////////////////////
00027 ConfigHolderStruct::ConfigHolderStruct() {
00028 }
00029
00030 // //////////////////////////////////////
00031 ConfigHolderStruct::
00032 ConfigHolderStruct (const ConfigHolderStruct& iConfigHolderStruct)
00033 : _pt (iConfigHolderStruct._pt) {
00034 }
00035
00036 // //////////////////////////////////////
00037 ConfigHolderStruct::~ConfigHolderStruct() {
00038 }
00039
00040 // //////////////////////////////////////
00041 void ConfigHolderStruct::toStream (std::ostream& ioOut) const {
00042     ioOut << describe();
00043 }
00044
00045 // //////////////////////////////////////
00046 void ConfigHolderStruct::fromStream (std::istream& ioIn) {
00047 }
00048
00049 // //////////////////////////////////////
00050 const std::string ConfigHolderStruct::describe() const {

```

```

00051     std::ostringstream ostr;
00052     ostr << "Configuration Display:" << std::endl;
00053
00054     // Look for the start and end date values.
00055     stdair::Date_T lStartDate;
00056     const bool hasStartDateBeenRetrieved =
00057         exportValue<Date_T> (lStartDate, "date.start");
00058     if (hasStartDateBeenRetrieved == true) {
00059         ostr << "    Start date: " << lStartDate << std::endl;
00060     }
00061     stdair::Date_T lEndDate;
00062     const bool hasEndDateBeenRetrieved =
00063         exportValue<Date_T> (lEndDate, "date.end");
00064     if (hasEndDateBeenRetrieved == true) {
00065         ostr << "    End date: " << lEndDate << std::endl;
00066     }
00067
00068     // Look for the random seed value.
00069     RandomSeed_T lRandomSeed;
00070     const bool hasSeedBeenRetrieved =
00071         exportValue<RandomSeed_T> (lRandomSeed, "random.seed");
00072     if (hasSeedBeenRetrieved == true) {
00073         ostr << "    Random Seed: " << lRandomSeed << std::endl;
00074     }
00075
00076     // Look for the demand generation method.
00077     char lChar;
00078     const bool hasDemandGenMethodBeenRetrieved =
00079         exportValue<char> (lChar, "demand generation.method");
00080     if (hasDemandGenMethodBeenRetrieved == true) {
00081         ostr << "    Demand Generation method: " << lChar << std::endl;
00082     }
00083
00084     // Look for the number of runs value.
00085     Count_T lTotalNumberOfRuns;
00086     const bool hasNumberOfRunsBeenRetrieved =
00087         exportValue<Count_T> (lTotalNumberOfRuns, "runs.number");
00088     if (hasNumberOfRunsBeenRetrieved == true) {
00089         ostr << "    Number Of Runs: " << lTotalNumberOfRuns << std::endl;
00090     }
00091
00092     // Look for the input files.
00093     stdair::Filename_T lFilename ("");
00094     const bool hasScheduleFileBeenRetrieved =
00095         exportValue<stdair::Filename_T> (lFilename, "input.schedule");
00096     if (hasScheduleFileBeenRetrieved == true) {
00097         ostr << "    Schedule input file: " << lFilename << std::endl;
00098     }
00099     const bool hasODFileBeenRetrieved =
00100         exportValue<stdair::Filename_T> (lFilename, "input.ond");
00101     if (hasODFileBeenRetrieved == true) {
00102         ostr << "    OnD input file: " << lFilename << std::endl;
00103     }
00104     const bool hasFrat5FileBeenRetrieved =
00105         exportValue<stdair::Filename_T> (lFilename, "input.frat5");
00106     if (hasFrat5FileBeenRetrieved == true) {
00107         ostr << "    Frat5 input file: " << lFilename << std::endl;
00108     }
00109     const bool hasFFdisutilityFileBeenRetrieved =
00110         exportValue<stdair::Filename_T> (lFilename, "input.ffdisutility");
00111     if (hasFFdisutilityFileBeenRetrieved == true) {
00112         ostr << "    FFdisutility input file: " << lFilename << std::endl;
00113     }
00114     const bool hasYieldFileBeenRetrieved =
00115         exportValue<stdair::Filename_T> (lFilename, "input.yield");
00116     if (hasYieldFileBeenRetrieved == true) {
00117         ostr << "    Yield input file: " << lFilename << std::endl;
00118     }
00119     const bool hasFareFileBeenRetrieved =
00120         exportValue<stdair::Filename_T> (lFilename, "input.fare");
00121     if (hasFareFileBeenRetrieved == true) {
00122         ostr << "    Fare input file: " << lFilename << std::endl;
00123     }
00124     const bool hasDemandFileBeenRetrieved =
00125         exportValue<stdair::Filename_T> (lFilename, "input.demand");
00126     if (hasDemandFileBeenRetrieved == true) {
00127         ostr << "    Demand input file: " << lFilename << std::endl;
00128     }
00129
00130     return ostr.str();
00131 }
00132
00133 // //////////////////////////////////////
00134 const std::string ConfigHolderStruct::jsonExport() const {
00135     std::ostringstream ostr;
00136     #if BOOST_VERSION >= 104100
00137     // Write the property tree into the JSON stream.

```

```

00138     write_json (oStr, _pt);
00139 #endif // BOOST_VERSION >= 104100
00140     return oStr.str();
00141 }
00142
00143 // //////////////////////////////////////
00144 void ConfigHolderStruct::add (const bpt::ptree& iConfigPropertyTree) {
00145     // Call the dedicated recursive method with an empty path in order to merge
00146     // the config property tree with the given new one.
00147     std::string lEmptyPath ("");
00148     add (iConfigPropertyTree, lEmptyPath);
00149 }
00150
00151 // //////////////////////////////////////
00152 void ConfigHolderStruct::add (const bpt::ptree& iConfigPropertyTree,
00153     const std::string& iPath) {
00154
00155     // Are there any more children to browse?
00156     bool isThereAnyChild = false;
00157
00158 #if BOOST_VERSION >= 104100
00159
00160     // Browse the children nodes
00161     BOOST_FOREACH(bpt::ptree::value_type itChild, iConfigPropertyTree) {
00162
00163         isThereAnyChild = true;
00164
00165         // Build the current path
00166         std::ostringstream lCurrentPathStr;
00167         const bool isPathEmptyForNow = iPath.empty();
00168         if (isPathEmptyForNow == false) {
00169             lCurrentPathStr << iPath << ".";
00170         }
00171         // Add the current node name
00172         lCurrentPathStr << itChild.first.data();
00173         const std::string lCurrentPath (lCurrentPathStr.str());
00174
00175         // Get the child tree
00176         const bpt::ptree& lChildTree = itChild.second;
00177         add(lChildTree, lCurrentPath);
00178     }
00179
00180     // If there is no child for this node, create the specified path and add
00181     // the corresponding value
00182     if (isThereAnyChild == false) {
00183         std::string lValue (iConfigPropertyTree.data());
00184         const bool hasInsertionBeenSuccessful = addValue (lValue, iPath);
00185         assert (hasInsertionBeenSuccessful == true);
00186     }
00187 #endif // BOOST_VERSION >= 104100
00188 }
00189
00190 // //////////////////////////////////////
00191 bool ConfigHolderStruct::addValue (const std::string& iValue,
00192     const std::string& iPath) {
00193     bool hasInsertionBeenSuccessful = true;
00194     // Create the given specified path and add the corresponding given value,
00195     // or replace the value if the path already exists.
00196 #if BOOST_VERSION >= 104100
00197
00198     try {
00199         std::size_t found;
00200         const std::string lPrefix ("config");
00201         std::string lFinalPath;
00202         found = iPath.find(lPrefix);
00203         if (found == std::string::npos) {
00204             lFinalPath += lPrefix;
00205             lFinalPath += ".";
00206         }
00207         lFinalPath += iPath;
00208         if (lFinalPath != lPrefix) {
00209             _pt.put (lFinalPath, iValue);
00210         }
00211     } catch (bpt::ptree_bad_data& bptException) {
00212         hasInsertionBeenSuccessful = false;
00213     }
00214 #endif // BOOST_VERSION >= 104100
00215     return hasInsertionBeenSuccessful;
00216 }
00217
00218 // //////////////////////////////////////
00219 void ConfigHolderStruct::updateAirlineFeatures (
00220     BomRoot& iBomRoot) {
00221
00222     AirlineCode_T lAirlineCode ("");
00223

```

```

00224 // Browse the children nodes
00225 BOOST_FOREACH(bpt::ptree::value_type itChild, _pt) {
00226     std::ostringstream lPathStr;
00227     lPathStr << itChild.first.data() << ".airline_code";
00228     const bool hasAirlineCodeBeenRetrieved =
00229         exportValue<AirlineCode_T> (lAirlineCode , lPathStr.str());
00230     if (hasAirlineCodeBeenRetrieved == true) {
00231         AirlineFeature* lAirlineFeature_ptr =
00232             BomRetriever::retrieveAirlineFeatureFromKey (iBomRoot,
lAirlineCode);
00233         if (lAirlineFeature_ptr != NULL) {
00234             try {
00235                 try {
00236                     std::ostringstream lPathStr;
00237                     char lChar;
00238
00239                     // Try to extract the forecasting method from the config tree
00240                     lPathStr << itChild.first.data() << ".forecasting_method";
00241                     const bool hasForecastingMethodBeenRetrieved =
00242                         exportValue<char> (lChar, lPathStr.str());
00243                     if (hasForecastingMethodBeenRetrieved == true) {
00244                         const ForecastingMethod lForecastingMethod (lChar);
00245                         lAirlineFeature_ptr->setForecastingMethod(lForecastingMethod);
00246                     }
00247
00248                     // Try to extract the unconstraining method from the config tree
00249                     lPathStr.str("");
00250                     lPathStr << itChild.first.data() << ".unconstraining_method";
00251                     const bool hasUnconstrainingMethodBeenRetrieved =
00252                         exportValue<char> (lChar, lPathStr.str());
00253                     if (hasUnconstrainingMethodBeenRetrieved == true) {
00254                         const UnconstrainingMethod lUnconstrainingMethod (lChar);
00255                         lAirlineFeature_ptr->setUnconstrainingMethod(lUnconstrainingMethod);
00256                     }
00257
00258                     // Try to extract the partnership technique from the config tree
00259                     lPathStr.str("");
00260                     lPathStr << itChild.first.data() << ".partnership_technique";
00261                     const bool hasPartnershipTechniqueBeenRetrieved =
00262                         exportValue<char> (lChar, lPathStr.str());
00263                     if (hasPartnershipTechniqueBeenRetrieved == true) {
00264                         const PartnershipTechnique lPartnershipTechnique (lChar);
00265                         lAirlineFeature_ptr->setPartnershipTechnique(lPartnershipTechnique);
00266                     }
00267
00268                     // Try to extract the pre optimisation method from the config tree
00269                     lPathStr.str("");
00270                     lPathStr << itChild.first.data() << ".pre_optimisation_method";
00271                     const bool hasPreOptMethodBeenRetrieved =
00272                         exportValue<char> (lChar, lPathStr.str());
00273                     if (hasPreOptMethodBeenRetrieved == true) {
00274                         const PreOptimisationMethod lPreOptimisationMethod (lChar);
00275                         lAirlineFeature_ptr->setPreOptimisationMethod(lPreOptimisationMethod)
00276                     };
00277                 }
00278
00279                 // Try to extract the optimisation method from the config tree
00280                 lPathStr.str("");
00281                 lPathStr << itChild.first.data() << ".optimisation_method";
00282                 const bool hasOptMethodBeenRetrieved =
00283                     exportValue<char> (lChar, lPathStr.str());
00284                 if (hasOptMethodBeenRetrieved == true) {
00285                     const OptimisationMethod lOptimisationMethod (lChar);
00286                     lAirlineFeature_ptr->setOptimisationMethod(lOptimisationMethod);
00287                 }
00288             } catch (CodeConversionException& lCodeConversionException) {
00289                 std::ostringstream oMessage;
00290                 oMessage << "Wrong input features for the airline '"
00291                     << lAirlineCode << "' in the input configuration file: "
00292                     << lCodeConversionException.what();
00293                 STDAIR_LOG_ERROR (oMessage.str());
00294                 throw CodeConversionException (oMessage.str());
00295             }
00296         }
00297     }
00298 }
00299 }
00300 }
00301 }

```

### 33.263 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>
```

## Classes

- struct [stdair::ConfigHolderStruct](#)

## Namespaces

- [bpt](#)
- [stdair](#)

*Handle on the StdAir library context.*

## 33.264 ConfigHolderStruct.hpp

```
00001 #ifndef __STDAIR_BOM_CONFIGHOLDERSTRUCT_HPP
00002 #define __STDAIR_BOM_CONFIGHOLDERSTRUCT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // Boost
00011 #include <boost/static_assert.hpp>
00012 #include <boost/type_traits/is_same.hpp>
00013 #if BOOST_VERSION >= 104100
00014 // Boost Property Tree
00015 #include <boost/property_tree/ptree.hpp>
00016 #endif // BOOST_VERSION >= 104100
00017 // StdAir
00018 #include <stdair/stdair_file.hpp>
00019 #include <stdair/stdair_maths_types.hpp>
00020 #include <stdair/stdair_date_time_types.hpp>
00021 #include <stdair/basic/StructAbstract.hpp>
00022 #include <stdair/bom/ConfigHolderTypes.hpp>
00023
00024 #if BOOST_VERSION >= 104100
00025 namespace bpt = boost::property_tree;
00026 #else // BOOST_VERSION >= 104100
00027 namespace bpt {
00028     typedef char ptree;
00029 }
00030 #endif // BOOST_VERSION >= 104100
00031
00032 namespace stdair {
00033
00034     class BomRoot;
00035
00036     struct ConfigHolderStruct : public StructAbstract {
00037     public:
00038         // ////////////////////////////////// Getters //////////////////////////////////
00039
00040         // ////////////////////////////////// Business Methods //////////////////////////////////
00041         void add (const bpt::ptree&);
00042
00043         bool addValue (const std::string& iValue,
00044                       const std::string& iPath);
00045
00046         template <typename ValueType>
00047         bool exportValue (ValueType& ioValue, const std::string& iPath) const;
00048
00049         void updateAirlineFeatures (BomRoot&);
00050
00051     private:
```



```

00087     void add (const bpt::ptree&,
00088               const std::string&);
00089
00090 public:
00091     // //////////// Display support method ////////////
00096     void toStream (std::ostream& ioOut) const;
00097
00102     void fromStream (std::istream& ioIn);
00103
00107     const std::string describe() const;
00108
00112     const std::string jsonExport() const;
00113
00114
00115     // //////////// Constructors and Destructors ////////////
00116 public:
00120     ConfigHolderStruct ();
00121
00125     ConfigHolderStruct (const ConfigHolderStruct&);
00126
00127 public:
00131     ~ConfigHolderStruct ();
00132
00133
00134 private:
00135     // //////////// Attributes ////////////
00139     bpt::ptree _pt;
00140 };
00141
00142 // //////////////////////////////////////
00143 template <typename ValueType>
00144 bool ConfigHolderStruct::exportValue (ValueType& ioValue,
00145                                       const std::string& iPath) const {
00146
00147     bool hasValueBeenSuccessfullyRetrieved = true;
00148
00149 #if BOOST_VERSION >= 104100
00150     try {
00151         // Get the value.
00152         // If the path key is not found, an exception is thrown.
00153         const std::string lPrefix ("config.");
00154         const std::string lFinalPath = lPrefix + iPath;
00155         ioValue = _pt.get<ValueType> (lFinalPath);
00156     } catch (bpt::ptree_error& bptException) {
00157         hasValueBeenSuccessfullyRetrieved = false;
00158     }
00160 #endif // BOOST_VERSION >= 104100
00161
00162     return hasValueBeenSuccessfullyRetrieved;
00163
00164 }
00165
00166 // //////////////////////////////////////
00167 //
00168 // Specialization of the template method exportValue above for the type
00169 // Date_T.
00170 //
00171 // //////////////////////////////////////
00172
00173 template<>
00174 inline bool ConfigHolderStruct::exportValue<Date_T>
00175 (Date_T& ioValue,
00176  const std::string& iPath) const {
00177
00178     bool hasValueBeenSuccessfullyRetrieved = true;
00179
00180 #if BOOST_VERSION >= 104100
00181     try {
00182
00184         // Get the string date value.
00185         // If the path key is not found, an exception is thrown.
00186         const std::string lPrefix ("config.");
00187         const std::string lFinalPath = lPrefix + iPath;
00188         const std::string& lDateStr =
00189             _pt.get<std::string> (lFinalPath);
00190
00191         // Convert the string into a Date_T.
00192         ioValue =
00193             boost::gregorian::from_simple_string (lDateStr);
00194     } catch (bpt::ptree_error& bptException) {
00195         hasValueBeenSuccessfullyRetrieved = false;
00196     }
00197 #endif // BOOST_VERSION >= 104100
00198
00199

```

```

00200     return hasValueBeenSuccessfullyRetrieved;
00201
00202
00203 }
00204
00205 }
00206
00207 #endif // __STDAIR_BOM_CONFIGHOLDERSTRUCT_HPP

```

### 33.265 stdair/bom/ConfigHolderTypes.hpp File Reference

```

#include <list>
#include <boost/shared_ptr.hpp>

```

#### Namespaces

- [stdair](#)  
*Handle on the StdAir library context.*

#### Typedefs

- typedef boost::shared\_ptr< ConfigHolderStruct > [stdair::ConfigHolderPtr\\_T](#)

### 33.266 ConfigHolderTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_CONFIGHOLDERTYPES_HPP
00003 #define __STDAIR_BOM_CONFIGHOLDERTYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <list>
00010 // Boost
00011 #include <boost/shared_ptr.hpp>
00012
00013 namespace stdair {
00014
00015     // Forward declarations
00016     struct ConfigHolderStruct;
00017
00018     // ////////////////////////////////// Type definitions //////////////////////////////////
00020     typedef boost::shared_ptr<ConfigHolderStruct> ConfigHolderPtr_T;
00021
00022 }
00023 #endif // __STDAIR_BOM_CONFIGHOLDERTYPES_HPP
00024

```

### 33.267 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

- [stdair](#)  
*Handle on the StdAir library context.*

### 33.268 DatePeriod.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_Period_BOM.hpp>
00009 #include <stdair/service/Logger.hpp>
00010 #include <stdair/bom/DatePeriod.hpp>
00011
00012 namespace stdair {
00013
00014 // //////////////////////////////////////
00015 DatePeriod::DatePeriod()
00016 : _key (BOOST_DEFAULT_DATE_PERIOD),
00017   _parent (NULL) {
00018     // That constructor is used by the serialisation process
00019 }
00020
00021 // //////////////////////////////////////
00022 DatePeriod::DatePeriod (const DatePeriod& iDatePeriod)
00023 : _key (iDatePeriod.getKey()), _parent (NULL) {
00024 }
00025
00026 // //////////////////////////////////////
00027 DatePeriod::DatePeriod (const Key_T& iKey)
00028 : _key (iKey), _parent (NULL) {
00029 }
00030
00031 // //////////////////////////////////////
00032 DatePeriod::~DatePeriod () {
00033 }
00034
00035 // //////////////////////////////////////
00036 std::string DatePeriod::toString() const {
00037     std::ostringstream oStr;
00038     oStr << describeKey();
00039     return oStr.str();
00040 }
00041
00042 // //////////////////////////////////////
00043 bool DatePeriod::
00044 isDepartureDateValid (const Date_T& iFlightDate) const {
00045
00046     // Check if the departure date is within the date range.
00047     const DatePeriod_T& lPeriod = getDatePeriod ();
00048     if (lPeriod.contains (iFlightDate) == false) {
00049         return false;
00050     }
00051
00052     return true;
00053 }
00054
00055 }
00056

```

### 33.269 stdair/bom/DatePeriod.hpp File Reference

```

#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/DatePeriodKey.hpp>
#include <stdair/bom/DatePeriodTypes.hpp>

```

#### Classes

- class [stdair::DatePeriod](#)  
Class representing the actual attributes for a fare date-period.

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.270 DatePeriod.hpp

```

00001 #ifndef __STDAIR_BOM_DATEPERIOD_HPP
00002 #define __STDAIR_BOM_DATEPERIOD_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STDAIR
00008 #include <stdair/bom/BomAbstract.hpp>
00009 #include <stdair/bom/DatePeriodKey.hpp>
00010 #include <stdair/bom/DatePeriodTypes.hpp>
00011
00012 // Forward declaration
00013 namespace stdair {
00014
00015     class DatePeriod : public BomAbstract {
00016     public:
00017         template <typename BOM> friend class FacBom;
00018         template <typename BOM> friend class FacCloneBom;
00019         friend class FacBomManager;
00020
00021     public:
00022         // ////////////////////////////////// Type definitions //////////////////////////////////
00023         typedef DatePeriodKey Key_T;
00024
00025     public:
00026         // ////////////////////////////////// Display support methods //////////////////////////////////
00027         void toStream (std::ostream& ioOut) const {
00028             ioOut << toString();
00029         }
00030
00031         void fromStream (std::istream& ioIn) {
00032
00033         }
00034
00035         std::string toString() const;
00036
00037         const std::string describeKey() const {
00038             return _key.toString();
00039         }
00040
00041     public:
00042         // ////////////////////////////////// Getters //////////////////////////////////
00043         const Key_T& getKey() const {
00044             return _key;
00045         }
00046
00047         BomAbstract* const getParent() const {
00048             return _parent;
00049         }
00050
00051         const HolderMap_T& getHolderMap() const {
00052             return _holderMap;
00053         }
00054
00055         const DatePeriod_T& getDatePeriod() const {
00056             return _key.getDatePeriod();
00057         }
00058
00059     public:
00060         // ////////////////////////////////// Business methods //////////////////////////////////
00061         bool isDepartureDateValid (const Date_T&) const;
00062
00063     protected:
00064         // ////////////////////////////////// Constructors and destructors //////////////////////////////////
00065         DatePeriod (const Key_T&);
00066         virtual ~DatePeriod ();
00067
00068     private:
00069         DatePeriod ();
00070         DatePeriod (const DatePeriod&);
00071
00072     protected:
00073         // ////////////////////////////////// Attributes //////////////////////////////////
00074         Key_T _key;
00075
00076         BomAbstract* _parent;
00077
00078         HolderMap_T _holderMap;
00079
00080     };
00081
00082 
```

```

00140 }
00141 #endif // __STDAIR_BOM_DATEPERIOD_HPP
00142

```

### 33.271 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

- [stdair](#)

*Handle on the StdAir library context.*

### 33.272 DatePeriodKey.cpp

```

00001 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00002 // Import section
00003 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <ostream>
00006 #include <sstream>
00007 // Boost Date-Time
00008 #include <boost/date_time/gregorian/formatters.hpp>
00009 // STDAIR
00010 #include <stdair/basic/BasConst_Period_BOM.hpp>
00011 #include <stdair/bom/DatePeriodKey.hpp>
00012
00013 namespace stdair {
00014
00015 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00016 DatePeriodKey::DatePeriodKey()
00017 : _datePeriod (BOOST_DEFAULT_DATE_PERIOD) {
00018     assert (false);
00019 }
00020
00021 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00022 DatePeriodKey::DatePeriodKey (const stdair::DatePeriod_T& iDatePeriod)
00023 : _datePeriod (iDatePeriod) {
00024 }
00025
00026 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00027 DatePeriodKey::DatePeriodKey (const DatePeriodKey& iKey)
00028 : _datePeriod (iKey._datePeriod) {
00029 }
00030
00031 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00032 DatePeriodKey::~DatePeriodKey () {
00033 }
00034
00035 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00036 void DatePeriodKey::toStream (std::ostream& ioOut) const {
00037     ioOut << "DatePeriodKey: " << toString() << std::endl;
00038 }
00039
00040 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00041 void DatePeriodKey::fromStream (std::istream& ioIn) {
00042 }
00043
00044 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00045 const std::string DatePeriodKey::toString() const {
00046     std::ostringstream oStr;
00047     const stdair::Date_T lStart = _datePeriod.begin();
00048     const stdair::Date_T lEnd = _datePeriod.end();
00049     oStr << "[" << boost::gregorian::to_simple_string(lStart)
00050         << "/" << boost::gregorian::to_simple_string(lEnd)
00051         << "]";
00052     return oStr.str();
00053 }
00054
00055 }

```

### 33.273 stdair/bom/DatePeriodKey.hpp File Reference

```
#include <stdair/bom/KeyAbstract.hpp>
#include <stdair/stdair_date_time_types.hpp>
```

#### Classes

- struct [stdair::DatePeriodKey](#)

*Key of date-period.*

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.274 DatePeriodKey.hpp

```
00001 #ifndef __SIMFQT_BOM_DATEPERIODKEY_HPP
00002 #define __SIMFQT_BOM_DATEPERIODKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STDAIR
00008 #include <stdair/bom/KeyAbstract.hpp>
00009 #include <stdair/stdair_date_time_types.hpp>
00010
00011 namespace stdair {
00014     struct DatePeriodKey : public KeyAbstract {
00015
00016     public:
00017         // ////////////////////////////////// Construction //////////////////////////////////
00019         DatePeriodKey (const DatePeriod_T&);
00021         DatePeriodKey (const DatePeriodKey&);
00023         ~DatePeriodKey ();
00024
00025     private:
00027         DatePeriodKey ();
00028
00029     public:
00030         // ////////////////////////////////// Getters //////////////////////////////////
00032         const DatePeriod_T& getDatePeriod() const {
00033             return _datePeriod;
00034         }
00035
00036     public:
00037
00038         // ////////////////////////////////// Display support methods //////////////////////////////////
00044         void toStream (std::ostream& ioOut) const;
00045
00051         void fromStream (std::istream& ioIn);
00052
00058         const std::string toString() const;
00059
00060     private:
00061         // ////////////////////////////////// Attributes //////////////////////////////////
00065         DatePeriod_T _datePeriod;
00066
00067     };
00068
00069 }
00070 #endif // __SIMFQT_BOM_DATEPERIODKEY_HPP
```

### 33.275 stdair/bom/DatePeriodTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## Typedefs

- typedef std::list< DatePeriod \* > [stdair::DatePeriodList\\_T](#)
- typedef std::map< const MapKey\_T, DatePeriod \* > [stdair::DatePeriodMap\\_T](#)
- typedef std::pair< MapKey\_T, DatePeriod \* > [stdair::DatePeriodWithKey\\_T](#)
- typedef std::list< DatePeriodWithKey\_T > [stdair::DatePeriodDetailedList\\_T](#)

### 33.276 DatePeriodTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_DATEPERIODTYPES_HPP
00003 #define __STDAIR_BOM_DATEPERIODTYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // STDAIR
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016     // Forward declarations.
00017     class DatePeriod;
00018
00020     typedef std::list<DatePeriod*> DatePeriodList_T;
00021
00023     typedef std::map<const MapKey_T, DatePeriod*> DatePeriodMap_T;
00024
00026     typedef std::pair<MapKey_T, DatePeriod*> DatePeriodWithKey_T;
00027     typedef std::list<DatePeriodWithKey_T> DatePeriodDetailedList_T;
00028 }
00029 #endif // __STDAIR_BOM_DATEPERIODTYPES_HPP
00030

```

### 33.277 stdair/bom/DoWStruct.cpp File Reference

```

#include <sstream>
#include <cassert>
#include <stdair/basic/BasConst_Period_BOM.hpp>
#include <stdair/bom/DoWStruct.hpp>

```

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.278 DoWStruct.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <sstream>
00006 #include <cassert>
00007 // STDAIR
00008 #include <stdair/basic/BasConst_Period_BOM.hpp>

```

```

00009 #include <stdair/bom/DoWStruct.hpp>
00010
00011 namespace stdair {
00012
00013 ///////////////////////////////////////////////////////////////////
00014 DoWStruct::DoWStruct () {
00015     for (unsigned short i = 0; i < 7; ++i) {
00016         _dowList.push_back (false);
00017     }
00018 }
00019
00020 ///////////////////////////////////////////////////////////////////
00021 DoWStruct::DoWStruct (const std::string& iDowString) {
00022     const unsigned short lDowStringSize = iDowString.size();
00023     assert (lDowStringSize == 7);
00024
00025     _dowList.reserve (lDowStringSize);
00026     for (std::string::const_iterator itChar = iDowString.begin();
00027         itChar != iDowString.end(); ++itChar) {
00028         const bool isDoWSet = (*itChar == '1')?true:false;
00029         _dowList.push_back (isDoWSet);
00030     }
00031 }
00032
00033 ///////////////////////////////////////////////////////////////////
00034 DoWStruct::DoWStruct (const DoWStruct& iDowStruct) :
00035     _dowList (iDowStruct._dowList) {
00036 }
00037
00038 ///////////////////////////////////////////////////////////////////
00039 const std::string DoWStruct::describeShort () const {
00040     std::ostringstream ostr;
00041     short i = 0;
00042     for (BooleanList_T::const_iterator itDoW = _dowList.begin();
00043         itDoW != _dowList.end(); ++itDoW, ++i) {
00044         const char lDoW = (*itDoW == true)?'1':'0';
00045         ostr << lDoW;
00046     }
00047     return ostr.str();
00048 }
00049
00050 ///////////////////////////////////////////////////////////////////
00051 const std::string DoWStruct::describe () const {
00052     std::ostringstream ostr;
00053     short i = 0;
00054     for (BooleanList_T::const_iterator itDoW = _dowList.begin();
00055         itDoW != _dowList.end(); ++itDoW, ++i) {
00056         const bool lDoW = *itDoW;
00057         if (lDoW == true) {
00058             ostr << DOW_STR[i] << ".";
00059         }
00060     }
00061     return ostr.str();
00062 }
00063
00064 ///////////////////////////////////////////////////////////////////
00065 bool DoWStruct::getDayOfWeek (const unsigned short i) const {
00066     return _dowList.at (i);
00067 }
00068
00069 ///////////////////////////////////////////////////////////////////
00070 bool DoWStruct::getStandardDayOfWeek (const unsigned short i) const {
00071     unsigned short iStd = i;
00072     if (iStd == 0) {
00073         iStd = 6;
00074     } else {
00075         --iStd;
00076     }
00077     return _dowList.at (iStd);
00078 }
00079
00080 ///////////////////////////////////////////////////////////////////
00081 void DoWStruct::setDayOfWeek (const unsigned short i, const bool iBool) {
00082     assert (i < 7);
00083     _dowList.at (i) = iBool;
00084 }
00085
00086 ///////////////////////////////////////////////////////////////////
00087 DoWStruct DoWStruct::shift (const long& iNbOfDays) const {
00088     DoWStruct oDoW (DEFAULT_DOW_STRING);
00089
00090     for (short i = 0; i < 7; ++i) {
00091         const bool lDoWBool = _dowList.at (i);
00092         short lIndex = (i + iNbOfDays) % 7;
00093         if (lIndex < 0) {
00094             lIndex += 7;
00095         }
00096     }
00097 }

```



```

00096     }
00097     oDoW.setDayOfWeek (lIndex, lDoWBool);
00098 }
00099
00100     return oDoW;
00101 }
00102
00103 // //////////////////////////////////////
00104 DoWStruct DoWStruct::intersection (const
DoWStruct& iDoW) const {
00105     DoWStruct oDoW (DEFAULT_DOW_STRING);
00106     for (unsigned short i = 0; i < 7; ++i) {
00107         if (getDayOfWeek(i) && iDoW.getDayOfWeek(i)) {
00108             oDoW.setDayOfWeek (i, true);
00109         } else {
00110             oDoW.setDayOfWeek (i, false);
00111         }
00112     }
00113     return oDoW;
00114 }
00115
00116 // //////////////////////////////////////
00117 const bool DoWStruct::isValid () const {
00118     for (unsigned short i = 0; i < 7; ++i) {
00119         if (getDayOfWeek(i)) {
00120             return true;
00121         }
00122     }
00123     return false;
00124 }
00125
00126 }

```

### 33.279 stdair/bom/DoWStruct.hpp File Reference

```

#include <string>
#include <vector>
#include <stdair/basic/StructAbstract.hpp>

```

#### Classes

- struct [stdair::DoWStruct](#)

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.280 DoWStruct.hpp

```

00001 #ifndef __STDAIR_BOM_DOWSTRUCT_HPP
00002 #define __STDAIR_BOM_DOWSTRUCT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010 // STDAIR
00011 #include <stdair/basic/StructAbstract.hpp>
00012
00013 namespace stdair {
00014
00018     struct DoWStruct : public StructAbstract {
00019     public:
00021         typedef std::vector<bool> BooleanList_T;
00022
00023     public:
00024         // ////////////////////////////////// Getters //////////////////////////////////
00026         bool getDayOfWeek (const unsigned short i) const;

```

```

00027
00029     bool getStandardDayOfWeek (const unsigned short i) const;
00030
00031 public:
00032     // ////////////////////////////////// Setters //////////////////////////////////
00034     void setDayOfWeek (const unsigned short, const bool);
00035
00036 public:
00037     // ////////////////////////////////// Display methods //////////////////////////////////
00039     const std::string describe() const;
00040
00042     const std::string describeShort() const;
00043
00044 public:
00045     // ////////////////////////////////// Business Methods //////////////////////////////////
00047     DoWStruct shift (const long&) const;
00048
00050     DoWStruct intersection (const DoWStruct&) const;
00051
00053     const bool isValid () const;
00054
00055 public:
00058     DoWStruct (const std::string& iDowString);
00060     DoWStruct ();
00061     DoWStruct (const DoWStruct&);
00063     ~DoWStruct () { }
00064
00065 private:
00067     BooleanList_T _dowList;
00068 };
00069
00070 }
00071 #endif // __STDAIR_BOM_DOWSTRUCT_HPP

```

### 33.281 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

- [stdair](#)

*Handle on the StdAir library context.*

### 33.282 EventStruct.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // Boost
00007 #if BOOST_VERSION >= 103900
00008 #include <boost/make_shared.hpp>
00009 #else // BOOST_VERSION >= 103900
00010 #include <boost/shared_ptr.hpp>
00011 #endif // BOOST_VERSION >= 103900
00012 // StdAir
00013 #include <stdair/basic/BasConst_General.hpp>
00014 #include <stdair/basic/BasConst_Event.hpp>
00015 #include <stdair/bom/BookingRequestStruct.hpp>

```

```

00016 #include <stdair/bom/OptimisationNotificationStruct.hpp>
00017 #include <stdair/bom/SnapshotStruct.hpp>
00018 #include <stdair/bom/CancellationStruct.hpp>
00019 #include <stdair/bom/RMEventStruct.hpp>
00020 #include <stdair/bom/BreakPointStruct.hpp>
00021 #include <stdair/bom/EventStruct.hpp>
00022
00023 namespace stdair {
00024
00025 // //////////////////////////////////////
00026 EventStruct::EventStruct()
00027 : _eventType (EventType::BKG_REQ), _eventTimeStamp (0) {
00028 }
00029
00030 // //////////////////////////////////////
00031 EventStruct::EventStruct (const
EventStruct::EventStruct (const
EventType::EN_EventType& iEventType,
BookingRequestPtr_T ioRequestPtr)
00032 : _eventType (iEventType) {
00033
00034 //
00035 assert (ioRequestPtr != NULL);
00036 #if BOOST_VERSION >= 103900
00037 _bookingRequest = boost::make_shared<BookingRequestStruct> (*ioRequestPtr);
00038 #else // BOOST_VERSION >= 103900
00039 _bookingRequest = ioRequestPtr;
00040 #endif // BOOST_VERSION >= 103900
00041 assert (_bookingRequest != NULL);
00042
00043 const Duration_T lDuration =
00044 _bookingRequest->getRequestDateTime() - DEFAULT_EVENT_OLDEST_DATETIME;
00045 _eventTimeStamp = lDuration.total_milliseconds();
00046 }
00047
00048 // //////////////////////////////////////
00049 EventStruct::EventStruct (const
EventStruct::EventStruct (const
EventType::EN_EventType& iEventType,
CancellationPtr_T ioCancellationPtr)
00050 : _eventType (iEventType) {
00051
00052 //
00053 assert (ioCancellationPtr != NULL);
00054 #if BOOST_VERSION >= 103900
00055 _cancellation = boost::make_shared<CancellationStruct> (*ioCancellationPtr);
00056 #else // BOOST_VERSION >= 103900
00057 _cancellation = ioCancellationPtr;
00058 #endif // BOOST_VERSION >= 103900
00059 assert (_cancellation != NULL);
00060
00061 const Duration_T lDuration =
00062 _cancellation->getCancellationDateTime() - DEFAULT_EVENT_OLDEST_DATETIME;
00063
00064 ;
00065 _eventTimeStamp = lDuration.total_milliseconds();
00066 }
00067
00068 // //////////////////////////////////////
00069 EventStruct::
EventStruct (const EventType::EN_EventType& iEventType,
const DateTime_T& iDCPDate,
OptimisationNotificationPtr_T ioOptimisationNotificationPtr)
00070 : _eventType (iEventType) {
00071
00072 //
00073 assert (ioOptimisationNotificationPtr != NULL);
00074 #if BOOST_VERSION >= 103900
00075 _optimisationNotification =
boost::make_shared<OptimisationNotificationStruct> (*ioOptimisationNotificationPtr);
00076 #else // BOOST_VERSION >= 103900
00077 _optimisationNotification = ioOptimisationNotificationPtr;
00078 #endif // BOOST_VERSION >= 103900
00079 assert (_optimisationNotification != NULL);
00080
00081 const Duration_T lDuration = iDCPDate -
DEFAULT_EVENT_OLDEST_DATETIME;
00082 _eventTimeStamp = lDuration.total_milliseconds();
00083 }
00084
00085 // //////////////////////////////////////
00086 EventStruct::EventStruct (const
EventStruct::EventStruct (const
EventType::EN_EventType& iEventType,
SnapshotPtr_T ioSnapshotPtr)
00087 : _eventType (iEventType) {
00088
00089 //
00090 assert (ioSnapshotPtr != NULL);
00091 #if BOOST_VERSION >= 103900
00092
00093

```

Generated on Sun Jun 28 2015 18:08:36 for StdAir by Doxygen

```

00213
00214 //
00215 if (iEventStruct._snapshot != NULL) {
00216 #if BOOST_VERSION >= 103900
00217     _snapshot = boost::make_shared<SnapshotStruct> (*iEventStruct._snapshot);
00218 #else // BOOST_VERSION >= 103900
00219     _snapshot = iEventStruct._snapshot;
00220 #endif // BOOST_VERSION >= 103900
00221 }
00222
00223 //
00224 if (iEventStruct._rmEvent != NULL) {
00225 #if BOOST_VERSION >= 103900
00226     _rmEvent = boost::make_shared<RMEventStruct> (*iEventStruct._rmEvent);
00227 #else // BOOST_VERSION >= 103900
00228     _rmEvent = iEventStruct._rmEvent;
00229 #endif // BOOST_VERSION >= 103900
00230 }
00231
00232 //
00233 if (iEventStruct._breakPoint != NULL) {
00234 #if BOOST_VERSION >= 103900
00235     _breakPoint = boost::make_shared<BreakPointStruct> (*iEventStruct._breakPoint);
00236 #else // BOOST_VERSION >= 103900
00237     _breakPoint = iEventStruct._breakPoint;
00238 #endif // BOOST_VERSION >= 103900
00239 }
00240 }
00241
00242 // //////////////////////////////////////
00243 EventStruct::~EventStruct() {
00244 }
00245
00246 // //////////////////////////////////////
00247 void EventStruct::fromStream (std::istream& ioIn) {
00248 }
00249
00250 // //////////////////////////////////////
00251 const std::string EventStruct::describe() const {
00252     std::ostringstream oStr;
00253
00254     //
00255     const Duration_T lEventDateTimeDelta =
00256         boost::posix_time::milliseconds (_eventTimeStamp);
00257     const DateTime_T lEventDateTime (DEFAULT_EVENT_OLDEST_DATETIME
00258         + lEventDateTimeDelta);
00259
00260     oStr << lEventDateTime;
00261
00262     //
00263     switch (_eventType) {
00264     case EventType::BKG_REQ: {
00265         assert (_bookingRequest != NULL);
00266         oStr << ", " << EventType::getLabel (_eventType)
00267             << ", " << _bookingRequest->describe();
00268         break;
00269     }
00270     case EventType::CX: {
00271         assert (_cancellation != NULL);
00272         oStr << ", " << EventType::getLabel (_eventType)
00273             << ", " << _cancellation->describe();
00274         break;
00275     }
00276     case EventType::OPT_NOT_4_FD: {
00277         assert (_optimisationNotification != NULL);
00278         oStr << ", " << EventType::getLabel (_eventType)
00279             << ", " << _optimisationNotification->describe();
00280         break;
00281     }
00282     case EventType::SNAPSHOT: {
00283         assert (_snapshot != NULL);
00284         oStr << ", " << EventType::getLabel (_eventType)
00285             << ", " << _snapshot->describe();
00286         break;
00287     }
00288     case EventType::RM: {
00289         assert (_rmEvent != NULL);
00290         oStr << ", " << EventType::getLabel (_eventType)
00291             << ", " << _rmEvent->describe();
00292         break;
00293     }
00294     case EventType::BRK_PT: {
00295         assert (_breakPoint != NULL);
00296         oStr << ", " << EventType::getLabel (_eventType)
00297             << ", " << _breakPoint->describe();
00298         break;
00299     }

```

```

00300     default: {
00301         ostr << ", " << _eventType << " (not yet recognised)";
00302         break;
00303     }
00304 }
00305
00306 ostr << "\n";
00307 return ostr.str();
00308 }
00309
00310 // //////////////////////////////////////
00311 const DateTime_T& EventStruct::getEventTime() const {
00312     const DateTime_T& lDateTime (DEFAULT_EVENT_OLDEST_DATETIME);
00313
00314     //
00315     switch (_eventType) {
00316     case EventType::BKG_REQ: {
00317         assert (_bookingRequest != NULL);
00318         return _bookingRequest->getRequestDateTime();
00319         break;
00320     }
00321     case EventType::CX: {
00322         assert (_cancellation != NULL);
00323         return _cancellation->getCancellationDateTime();
00324         break;
00325     }
00326     case EventType::OPT_NOT_4_FD: {
00327         assert (_optimisationNotification != NULL);
00328         return _optimisationNotification->getNotificationDateTime();
00329         break;
00330     }
00331     case EventType::SNAPSHOT: {
00332         assert (_snapshot != NULL);
00333         return _snapshot->getSnapshotTime();
00334         break;
00335     }
00336     case EventType::RM: {
00337         assert (_rmEvent != NULL);
00338         return _rmEvent->getRMEventTime();
00339         break;
00340     }
00341     case EventType::BRK_PT: {
00342         assert (_breakPoint != NULL);
00343         return _breakPoint->getBreakPointTime();
00344         break;
00345     }
00346     default: {
00347         assert(false);
00348         return lDateTime;
00349         break;
00350     }
00351 }
00352
00353 return lDateTime;
00354 }
00355
00356 // //////////////////////////////////////
00357 void EventStruct::incrementEventTimeStamp() {
00358     // The date-time is counted in milliseconds (1e-3 second). Hence,
00359     // one thousand (1e3) of attempts correspond to 1 second.
00360     // Increment the time stamp of one millisecond.
00361     ++_eventTimeStamp;
00362 }
00363
00364 }

```

### 33.283 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>

```

## Classes

- struct [stdair::EventStruct](#)

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.284 EventStruct.hpp

```

00001 #ifndef __STDAIR_BAS_EVENTSTRUCT_HPP
00002 #define __STDAIR_BAS_EVENTSTRUCT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_basic_types.hpp>
00012 #include <stdair/stdair_date_time_types.hpp>
00013 #include <stdair/stdair_event_types.hpp>
00014 #include <stdair/basic/StructAbstract.hpp>
00015 #include <stdair/basic/EventType.hpp>
00016 #include <stdair/bom/EventTypes.hpp>
00017 #include <stdair/bom/BookingRequestTypes.hpp>
00018 #include <stdair/bom/OptimisationNotificationTypes.hpp>
00019 #include <stdair/bom/SnapshotTypes.hpp>
00020 #include <stdair/bom/CancellationTypes.hpp>
00021 #include <stdair/bom/RMEventTypes.hpp>
00022 #include <stdair/bom/BreakPointTypes.hpp>
00023
00024 namespace stdair {
00025
00036 struct EventStruct : public StructAbstract {
00037
00038     // ////////////////////////////////// Getters //////////////////////////////////
00039 public:
00041     const EventType::EN_EventType& getEventType() const {
00042         return _eventType;
00043     }
00044
00046     const LongDuration_T& getEventTimeStamp() const {
00047         return _eventTimeStamp;
00048     }
00049
00051     const DateTime_T& getEventTime () const;
00052
00059     const BookingRequestStruct& getBookingRequest () const {
00060         assert (_bookingRequest != NULL);
00061         return *_bookingRequest;
00062     }
00063
00070     const CancellationStruct& getCancellation() const {

```

```

00071     assert (_cancellation != NULL);
00072     return *_cancellation;
00073 }
00074
00082     const OptimisationNotificationStruct&
00083     getOptimisationNotificationStruct() const {
00084         assert (_optimisationNotification != NULL);
00085         return *_optimisationNotification;
00086     }
00087
00095     const SnapshotStruct& getSnapshotStruct() const {
00096         assert (_snapshot != NULL);
00097         return *_snapshot;
00098     }
00099
00107     const RMEventStruct& getRMEvent() const {
00108         assert (_rmEvent != NULL);
00109         return *_rmEvent;
00110     }
00111
00118     const BreakPointStruct& getBreakPoint() const {
00119         assert (_breakPoint != NULL);
00120         return *_breakPoint;
00121     }
00122
00123     // ////////// Display methods //////////
00124 public:
00127     void fromStream (std::istream& ioIn);
00128
00130     const std::string describe() const;
00131
00132     // ////////// Constructors and destructors //////////
00133 public:
00136     EventStruct ();
00138     EventStruct (const EventType::EN_EventType&,
BookingRequestPtr_T);
00140     EventStruct (const EventType::EN_EventType&,
CancellationPtr_T);
00142     EventStruct (const EventType::EN_EventType&, const
DateTime_T& idCPDate,
00143                 OptimisationNotificationPtr_T);
00145     EventStruct (const EventType::EN_EventType&,
SnapshotPtr_T);
00147     EventStruct (const EventType::EN_EventType&,
RMEventPtr_T);
00149     EventStruct (const EventType::EN_EventType&,
BreakPointPtr_T);
00151     EventStruct (const EventStruct&);
00152
00154     ~EventStruct ();
00155
00156     // ////////// Modifiers //////////
00157 public:
00166     void incrementEventTimeStamp();
00167
00168     // ////////////////// Attributes //////////////////
00169 private:
00173     EventType::EN_EventType _eventType;
00174
00180     LongDuration_T _eventTimeStamp;
00181
00185     BookingRequestPtr_T _bookingRequest;
00186
00190     CancellationPtr_T _cancellation;
00191
00195     OptimisationNotificationPtr_T _optimisationNotification;
00196
00200     SnapshotPtr_T _snapshot;
00201
00205     RMEventPtr_T _rmEvent;
00206
00210     BreakPointPtr_T _breakPoint;
00211 };
00212
00213 }
00214 #endif // __STDAIR_BAS_EVENTSTRUCT_HPP

```

## 33.285 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

- [stdair](#)

*Handle on the StdAir library context.*

## Typedefs

- typedef std::pair< const LongDuration\_T, EventStruct > [stdair::EventListElement\\_T](#)
- typedef std::map< const LongDuration\_T, EventStruct > [stdair::EventList\\_T](#)

## 33.286 EventTypees.hpp

```
00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_EVENTTYPES_HPP
00003 #define __STDAIR_BOM_EVENTTYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <map>
00010 // Boost Smart Pointers
00011 #include <boost/shared_ptr.hpp>
00012 // StdAir
00013 #include <stdair/stdair_basic_types.hpp>
00014 #include <stdair/stdair_date_time_types.hpp>
00015 #include <stdair/stdair_event_types.hpp>
00016 #include <stdair/basic/ProgressStatus.hpp>
00017 #include <stdair/bom/key_types.hpp>
00018
00019 namespace stdair {
00020
00021     struct EventStruct;
00022
00023     typedef std::pair<const LongDuration_T, EventStruct> EventListElement_T;
00024
00025     typedef std::map<const LongDuration_T, EventStruct> EventList_T;
00026 }
00027 #endif // __STDAIR_BOM_EVENTTYPES_HPP
00028
00029
```

## 33.287 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

- [stdair](#)

*Handle on the StdAir library context.*

## 33.288 FareFamily.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/access.hpp>
00011 // StdAir
00012 #include <stdair/basic/BasConst_Inventory.hpp>
00013 #include <stdair/bom/FareFamily.hpp>
00014
00015 namespace stdair {
00016
00017 // //////////////////////////////////////
00018 FareFamily::FareFamily() : _key (DEFAULT_FARE_FAMILY_CODE), _parent (NULL) {
00019     assert (false);
00020 }
00021
00022 // //////////////////////////////////////
00023 FareFamily::FareFamily (const FareFamily& iFareFamily)
00024 : _key (iFareFamily._key),
00025   _parent (NULL),
00026   _frat5Curve (iFareFamily._frat5Curve),
00027   _disutilityCurve (iFareFamily._disutilityCurve),
00028   _meanStdDev (iFareFamily._meanStdDev) {
00029 }
00030
00031 // //////////////////////////////////////
00032 FareFamily::FareFamily (const Key_T& iKey) : _key (iKey), _parent (NULL) {
00033 }
00034
00035 // //////////////////////////////////////
00036 FareFamily::~FareFamily() {
00037 }
00038
00039 // //////////////////////////////////////
00040 std::string FareFamily::toString() const {
00041     std::ostringstream oStr;
00042     oStr << describeKey();
00043     return oStr.str();
00044 }
00045
00046 // //////////////////////////////////////
00047 void FareFamily::serialisationImplementationExport() const {
00048     std::ostringstream oStr;
00049     boost::archive::text_oarchive oa (oStr);
00050     oa << *this;
00051 }
00052
00053 // //////////////////////////////////////
00054 void FareFamily::serialisationImplementationImport() {
00055     std::istringstream iStr;
00056     boost::archive::text_iarchive ia (iStr);
00057     ia >> *this;
00058 }
00059
00060 // //////////////////////////////////////
00061 template<class Archive>
00062 void FareFamily::serialize (Archive& ioArchive,
00063                             const unsigned int iFileVersion) {
00064     ioArchive & _key;
00065 }
00066
00067 }
00068
00069

```

## 33.289 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>

```

## Classes

- class `stdair::FareFamily`  
*Class representing the actual attributes for a family fare.*

## Namespaces

- `boost`  
*Forward declarations.*
- `boost::serialization`
- `stdair`  
*Handle on the StdAir library context.*

## 33.290 FareFamily.hpp

```

00001 #ifndef __STDAIR_BOM_FAREFAMILY_HPP
00002 #define __STDAIR_BOM_FAREFAMILY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_rm_types.hpp>
00012 #include <stdair/bom/BomAbstract.hpp>
00013 #include <stdair/bom/FareFamilyKey.hpp>
00014 #include <stdair/bom/FareFamilyTypes.hpp>
00015
00016 namespace boost {
00017     namespace serialization {
00018         class access;
00019     }
00020 }
00021
00022 namespace stdair {
00023
00024     class FareFamily : public BomAbstract {
00025     public:
00026         template <typename BOM> friend class FacBom;
00027         template <typename BOM> friend class FacCloneBom;
00028         friend class FacBomManager;
00029         friend class boost::serialization::access;
00030
00031         // ////////////////////////////////// Type definitions //////////////////////////////////
00032         typedef FareFamilyKey Key_T;
00033
00034     public:
00035         // ////////////////////////////////// Getters //////////////////////////////////
00036         const Key_T& getKey() const {
00037             return _key;
00038         }
00039
00040         BomAbstract* const getParent() const {
00041             return _parent;
00042         }
00043
00044         const FamilyCode_T& getFamilyCode() const {
00045             return _key.getFamilyCode();
00046         }
00047
00048         const HolderMap_T& getHolderMap() const {
00049             return _holderMap;
00050         }
00051
00052         const FRAT5Curve_T& getFrat5Curve () const {
00053             return _frat5Curve;
00054         }
00055
00056         const FFDisutilityCurve_T& getDisutilityCurve () const {
00057             return _disutilityCurve;
00058         }
00059     };
00060
00061 }
00062
00063 
```

```

00072     }
00073
00075     const MeanValue_T& getMean () const { return _mean; }
00076     const StdDevValue_T& getStdDev () const {return
_stdDev; }
00077
00079     const MeanStdDevPairVector_T& getMeanStdDev () const {return
_meanStdDev; }
00080
00081
00082 public:
00083     // //////////// Setters ////////////
00085     void setFrat5Curve (const FRAT5Curve_T& iFRAT5Curve) {
00086         _frat5Curve = iFRAT5Curve;
00087     }
00088
00090     void setDisutilityCurve (const FFDisutilityCurve_T&
iDisutilityCurve) {
00091         _disutilityCurve = iDisutilityCurve;
00092     }
00093
00095     void setMean (const MeanValue_T& iMean) { _mean = iMean; }
00096     void setStdDev (const StdDevValue_T& iStdDev) {
_stdDev = iStdDev; }
00097
00099     void setMeanStdDev (const MeanStdDevPairVector_T& iMeanStdDev){
00100         _meanStdDev = iMeanStdDev;
00101     }
00102
00103
00104 public:
00105     // //////////// Display support methods ////////////
00111     void toStream (std::ostream& ioOut) const {
00112         ioOut << toString();
00113     }
00114
00120     void fromStream (std::istream& ioIn) {
00121     }
00122
00126     std::string toString() const;
00127
00131     const std::string describeKey() const {
00132         return _key.toString();
00133     }
00134
00135
00136 public:
00137     // //////////// (Boost) Serialisation support methods ////////////
00141     template<class Archive>
00142     void serialize (Archive& ar, const unsigned int iFileVersion);
00143
00144
00145 private:
00150     void serialisationImplementationExport() const;
00151     void serialisationImplementationImport();
00152
00153
00154 protected:
00155     // //////////// Constructors and destructors ////////////
00159     FareFamily (const Key_T&);
00160
00164     virtual ~FareFamily();
00165
00166
00167 private:
00171     FareFamily();
00172
00176     FareFamily (const FareFamily&);
00177
00178
00179 public:
00180     // //////////// Attributes ////////////
00184     Key_T _key;
00185
00189     BomAbstract* _parent;
00190
00194     HolderMap_T _holderMap;
00195
00199     FRAT5Curve_T _frat5Curve;
00200
00204     FFDisutilityCurve_T _disutilityCurve;
00205
00207     MeanValue_T _mean;
00208     StdDevValue_T _stdDev;
00209
00213     MeanStdDevPairVector_T _meanStdDev;
00214 };

```

```

00215
00216 }
00217 #endif // __STDAIR_BOM_FAREFAMILY_HPP
00218

```

### 33.291 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

- [stdair](#)  
*Handle on the StdAir library context.*

#### Functions

- template void [stdair::FareFamilyKey::serialize< ba::text\\_oarchive >](#) (ba::text\_oarchive &, unsigned int)
- template void [stdair::FareFamilyKey::serialize< ba::text\\_iarchive >](#) (ba::text\_iarchive &, unsigned int)

### 33.292 FareFamilyKey.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/access.hpp>
00011 // StdAir
00012 #include <stdair/basic/BasConst_Inventory.hpp>
00013 #include <stdair/bom/FareFamilyKey.hpp>
00014
00015 namespace stdair {
00016
00017 // //////////////////////////////////////
00018 FareFamilyKey::FareFamilyKey() : _familyCode (DEFAULT_FARE_FAMILY_CODE) {
00019     assert (false);
00020 }
00021
00022 // //////////////////////////////////////
00023 FareFamilyKey::FareFamilyKey (const FareFamilyKey& iFareFamilyKey)
00024 : _familyCode (iFareFamilyKey._familyCode) {
00025 }
00026
00027 // //////////////////////////////////////
00028 FareFamilyKey::FareFamilyKey (const FamilyCode_T& iFamilyCode)
00029 : _familyCode (iFamilyCode) {
00030 }
00031
00032 // //////////////////////////////////////
00033 FareFamilyKey::~FareFamilyKey() {
00034 }
00035
00036 // //////////////////////////////////////
00037 void FareFamilyKey::toStream (std::ostream& ioOut) const {
00038     ioOut << "FareFamilyKey: " << toString();
00039 }
00040
00041 // //////////////////////////////////////
00042 void FareFamilyKey::fromStream (std::istream& ioIn) {

```

```

00043     }
00044
00045     // //////////////////////////////////////
00046     const std::string FareFamilyKey::toString() const {
00047         std::ostringstream ostr;
00048         ostr << _familyCode;
00049         return ostr.str();
00050     }
00051
00052     // //////////////////////////////////////
00053     void FareFamilyKey::serialisationImplementationExport() const {
00054         std::ostringstream ostr;
00055         boost::archive::text_oarchive oa (ostr);
00056         oa << *this;
00057     }
00058
00059     // //////////////////////////////////////
00060     void FareFamilyKey::serialisationImplementationImport() {
00061         std::istringstream istr;
00062         boost::archive::text_iarchive ia (istr);
00063         ia >> *this;
00064     }
00065
00066     // //////////////////////////////////////
00067     template<class Archive>
00068     void FareFamilyKey::serialize (Archive& ioArchive,
00069                                   const unsigned int iFileVersion) {
00070         ioArchive & _familyCode;
00071     }
00072
00073     // //////////////////////////////////////
00074     // Explicit template instantiation
00075     namespace ba = boost::archive;
00076     template void FareFamilyKey::serialize<ba::text_oarchive> (ba::text_oarchive&,
00077                                                             unsigned int);
00078     template void FareFamilyKey::serialize<ba::text_iarchive> (ba::text_iarchive&,
00079                                                             unsigned int);
00080     // //////////////////////////////////////
00081
00082 }

```

### 33.293 stdair/bom/FareFamilyKey.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>

```

#### Classes

- struct [stdair::FareFamilyKey](#)  
*Key of a given fare family, made of a fare family code.*

#### Namespaces

- [boost](#)  
*Forward declarations.*
- [boost::serialization](#)
- [stdair](#)  
*Handle on the StdAir library context.*

### 33.294 FareFamilyKey.hpp

```

00001 #ifndef __STDAIR_BOM_FAREFAMILYKEY_HPP
00002 #define __STDAIR_BOM_FAREFAMILYKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section

```

```

00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/bom/KeyAbstract.hpp>
00013
00015 namespace boost {
00016     namespace serialization {
00017         class access;
00018     }
00019 }
00020
00021 namespace stdair {
00022
00026     struct FareFamilyKey : public KeyAbstract {
00027         friend class boost::serialization::access;
00028
00029         // ////////////////////////////////// Constructors and destructors //////////////////////////////////
00030     private:
00031         FareFamilyKey();
00032
00033     public:
00034         FareFamilyKey (const FamilyCode_T& iFamilyCode);
00035
00036         FareFamilyKey (const FareFamilyKey&);
00037
00038         ~FareFamilyKey();
00039
00040     public:
00041         // ////////////////////////////////// Getters //////////////////////////////////
00042         const FamilyCode_T& getFamilyCode () const {
00043             return _familyCode;
00044         }
00045
00046     public:
00047         // ////////////////////////////////// Display support methods //////////////////////////////////
00048         void toStream (std::ostream& ioOut) const;
00049
00050         void fromStream (std::istream& ioIn);
00051
00052         const std::string toString() const;
00053
00054     public:
00055         // ////////////////////////////////// (Boost) Serialisation support methods //////////////////////////////////
00056         template<class Archive>
00057         void serialize (Archive& ar, const unsigned int iFileVersion);
00058
00059     private:
00060         void serialisationImplementationExport() const;
00061         void serialisationImplementationImport();
00062
00063     private:
00064         // ////////////////////////////////// Attributes //////////////////////////////////
00065         FamilyCode_T _familyCode;
00066     };
00067 }
00068
00069 #endif // __STDAIR_BOM_FAREFAMILYKEY_HPP

```

### 33.295 stdair/bom/FareFamilyTypes.hpp File Reference

```

#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>

```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## Typedefs

- typedef std::list< FareFamily \* > [stdair::FareFamilyList\\_T](#)
- typedef std::map< const MapKey\_T, FareFamily \* > [stdair::FareFamilyMap\\_T](#)

## 33.296 FareFamilyTypes.hpp

```

00001 ///////////////////////////////////////////////////////////////////
00002 #ifndef __STDAIR_BOM_FAREFAMILYTYPES_HPP
00003 #define __STDAIR_BOM_FAREFAMILYTYPES_HPP
00004
00005 ///////////////////////////////////////////////////////////////////
00006 // Import section
00007 ///////////////////////////////////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // StdAir
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016     // Forward declarations.
00017     class FareFamily;
00018
00019     typedef std::list<FareFamily*> FareFamilyList_T;
00020
00021     typedef std::map<const MapKey_T, FareFamily*> FareFamilyMap_T;
00022 }
00023
00024 #endif // __STDAIR_BOM_FAREFAMILYTYPES_HPP

```

## 33.297 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

- [stdair](#)  
*Handle on the StdAir library context.*

## 33.298 FareFeatures.cpp

```

00001 ///////////////////////////////////////////////////////////////////
00002 // Import section
00003 ///////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_DefaultObject.hpp>
00009 #include <stdair/basic/BasConst_Request.hpp>
00010 #include <stdair/service/Logger.hpp>
00011 #include <stdair/bom/FareFeatures.hpp>
00012
00013 namespace stdair {
00014
00015     ///////////////////////////////////////////////////////////////////
00016     FareFeatures::FareFeatures()
00017     : _key (TRIP_TYPE_ONE_WAY,
00018           NO_ADVANCE_PURCHASE,
00019           SATURDAY_STAY,
00020           CHANGE_FEES,
00021           NON_REFUNDABLE,
00022           NO_STAY_DURATION),

```



```

00023     _parent (NULL) {
00024         // That constructor is used by the serialisation process
00025     }
00026
00027     // //////////////////////////////////////
00028     FareFeatures::FareFeatures (const FareFeatures& iFeatures)
00029         : _key (iFeatures.getKey()), _parent (NULL) {
00030     }
00031
00032     // //////////////////////////////////////
00033     FareFeatures::FareFeatures (const Key_T& iKey)
00034         : _key (iKey), _parent (NULL) {
00035     }
00036
00037     // //////////////////////////////////////
00038     FareFeatures::~FareFeatures () {
00039     }
00040
00041     // //////////////////////////////////////
00042     std::string FareFeatures::toString() const {
00043         std::ostringstream ostr;
00044         ostr << describeKey();
00045         return ostr.str();
00046     }
00047
00048     // //////////////////////////////////////
00049     bool FareFeatures::
00050     isTripTypeValid (const TripType_T& iBookingRequestTripType) const {
00051         bool oIsTripTypeValidFlag = true;
00052
00053         const TripType_T lFareTripType = getTripType();
00054         // Check whether the fare trip type is the same as the booking request
00055         // trip type
00056         if (iBookingRequestTripType == lFareTripType) {
00057             // One way case
00058             return oIsTripTypeValidFlag;
00059         }
00060
00061         if (iBookingRequestTripType == TRIP_TYPE_INBOUND
00062             || iBookingRequestTripType == TRIP_TYPE_OUTBOUND) {
00063             // Round trip case
00064             if (lFareTripType == TRIP_TYPE_ROUND_TRIP) {
00065                 return oIsTripTypeValidFlag;
00066             }
00067         }
00068
00069         oIsTripTypeValidFlag = false;
00070         return oIsTripTypeValidFlag;
00071     }
00072
00073     // //////////////////////////////////////
00074     bool FareFeatures::
00075     isStayDurationValid (const DayDuration_T& iStayDuration) const {
00076
00077         // Check if the stay duration is lower or equal to the minimum one.
00078         const DayDuration_T lMinimumDayDuration = getMinimumStay();
00079         if (lMinimumDayDuration > iStayDuration) {
00080             return false;
00081         }
00082
00083         return true;
00084     }
00085
00086     // //////////////////////////////////////
00087     bool FareFeatures::
00088     isAdvancePurchaseValid (const DateTime_T& iBookingRequestDateTime,
00089                             const DateTime_T& iFlightDateTime) const {
00090         bool oIsAdvancePurchaseValidFlag = true;
00091
00092         // Check whether the departure date is within the date range.
00093         const DayDuration_T lAdvancedPurchase = getAdvancePurchase();
00094         const DateOffset_T lMinimumAdvancedPurchase (lAdvancedPurchase);
00095         const DateTime_T lCriticalDate = iFlightDateTime - lMinimumAdvancedPurchase;
00096
00097         if (lCriticalDate < iBookingRequestDateTime) {
00098             oIsAdvancePurchaseValidFlag = false;
00099             return oIsAdvancePurchaseValidFlag;
00100         }
00101
00102         return true;
00103     }
00104 }
00105 }
00106

```

## 33.299 stdair/bom/FareFeatures.hpp File Reference

```
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/FareFeaturesKey.hpp>
#include <stdair/bom/FareFeaturesTypes.hpp>
```

## Classes

- class [stdair::FareFeatures](#)  
Class representing the actual attributes for a fare date-period.

## Namespaces

- [stdair](#)  
Handle on the StdAir library context.

## 33.300 FareFeatures.hpp

```
00001 #ifndef __STDAIR_BOM_FAREFEATURES_HPP
00002 #define __STDAIR_BOM_FAREFEATURES_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/bom/BomAbstract.hpp>
00009 #include <stdair/bom/FareFeaturesKey.hpp>
00010 #include <stdair/bom/FareFeaturesTypes.hpp>
00011
00012 // Forward declaration
00013 namespace stdair {
00014
00018     class FareFeatures : public BomAbstract {
00019     public:
00020         template <typename BOM> friend class FacBom;
00021         template <typename BOM> friend class FacCloneBom;
00022         friend class FacBomManager;
00023
00024         // ////////////////////////////////// Type definitions //////////////////////////////////
00028         typedef FareFeaturesKey Key_T;
00029
00030     public:
00031         // ////////////////////////////////// Display support methods //////////////////////////////////
00037         void toStream (std::ostream& ioOut) const {
00038             ioOut << toString();
00039         }
00040
00046         void fromStream (std::istream& ioIn) {
00047             ;
00048         }
00052         std::string toString() const;
00053
00057         const std::string describeKey() const {
00058             return _key.toString();
00059         }
00060
00061     public:
00063         // ////////////////////////////////// Getters //////////////////////////////////
00067         const Key_T& getKey() const {
00068             return _key;
00069         }
00070
00074         BomAbstract* const getParent() const {
00075             return _parent;
00076         }
00077
00081         const HolderMap_T& getHolderMap() const {
00082             return _holderMap;
00083         }
00084
00088         const TripType_T& getTripType() const {
00089             return _key.getTripType();
00090         }
00091     };
00092 }
```

```

00090     }
00091
00095     const DayDuration_T& getAdvancePurchase() const {
00096         return _key.getAdvancePurchase();
00097     }
00098
00102     const SaturdayStay_T& getSaturdayStay() const {
00103         return _key.getSaturdayStay();
00104     }
00105
00109     const ChangeFees_T& getChangeFees() const {
00110         return _key.getChangeFees();
00111     }
00112
00116     const NonRefundable_T& getRefundableOption() const {
00117         return _key.getRefundableOption();
00118     }
00119
00123     const DayDuration_T& getMinimumStay() const {
00124         return _key.getMinimumStay();
00125     }
00126
00127
00128 public:
00129     // //////////// Business methods ////////////
00134     bool isTripTypeValid (const TripType_T&) const;
00135
00140     bool isStayDurationValid (const DayDuration_T&) const;
00141
00146     bool isAdvancePurchaseValid (const DateTime_T& iBookingRequestDateTime,
00147                                 const DateTime_T& iFlightDateTime) const;
00148
00149
00150 protected:
00151     // //////////// Constructors and destructors ////////////
00155     FareFeatures (const Key_T&);
00159     virtual ~FareFeatures ();
00160
00161 private:
00165     FareFeatures ();
00169     FareFeatures (const FareFeatures&);
00170
00171 protected:
00172     // //////////// Attributes ////////////
00176     Key_T _key;
00177
00181     BomAbstract* _parent;
00182
00186     HolderMap_T _holderMap;
00187 };
00188
00189 }
00190 #endif // __STDAIR_BOM_FAREFEATURES_HPP
00191

```

### 33.301 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

- [stdair](#)

*Handle on the StdAir library context.*

### 33.302 FareFeaturesKey.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////

```

```

00004 // STL
00005 #include <ostream>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_DefaultObject.hpp>
00009 #include <stdair/basic/BasConst_Request.hpp>
00010 #include <stdair/bom/FareFeaturesKey.hpp>
00011
00012 namespace stdair {
00013
00014 // //////////////////////////////////////
00015 FareFeaturesKey::FareFeaturesKey()
00016 : _tripType (TRIP_TYPE_ONE_WAY),
00017   _advancePurchase (NO_ADVANCE_PURCHASE),
00018   _saturdayStay (SATURDAY_STAY),
00019   _changeFees (CHANGE_FEES),
00020   _nonRefundable (NON_REFUNDABLE),
00021   _minimumStay (NO_STAY_DURATION) {
00022   assert (false);
00023 }
00024
00025 // //////////////////////////////////////
00026 FareFeaturesKey::FareFeaturesKey (const TripType_T& iTripType,
00027                                   const DayDuration_T& iAdvancePurchase,
00028                                   const SaturdayStay_T& iSaturdayStay,
00029                                   const ChangeFees_T& iChangeFees,
00030                                   const NonRefundable_T& iNonRefundable,
00031                                   const DayDuration_T& iMinimumStay)
00032 : _tripType (iTripType), _advancePurchase (iAdvancePurchase),
00033   _saturdayStay (iSaturdayStay), _changeFees (iChangeFees),
00034   _nonRefundable (iNonRefundable), _minimumStay (iMinimumStay) {
00035 }
00036
00037 // //////////////////////////////////////
00038 FareFeaturesKey::FareFeaturesKey (const FareFeaturesKey& iKey)
00039 : _tripType (iKey.getTripType()),
00040   _advancePurchase (iKey.getAdvancePurchase()),
00041   _saturdayStay (iKey.getSaturdayStay()),
00042   _changeFees (iKey.getChangeFees()),
00043   _nonRefundable (iKey.getRefundableOption()),
00044   _minimumStay (iKey.getMinimumStay()) {
00045 }
00046
00047 // //////////////////////////////////////
00048 FareFeaturesKey::~FareFeaturesKey() {
00049 }
00050
00051 // //////////////////////////////////////
00052 void FareFeaturesKey::toStream (std::ostream& ioOut) const {
00053   ioOut << "FareFeaturesKey: " << toString() << std::endl;
00054 }
00055
00056 // //////////////////////////////////////
00057 void FareFeaturesKey::fromStream (std::istream& ioIn) {
00058 }
00059
00060 // //////////////////////////////////////
00061 const std::string FareFeaturesKey::toString() const {
00062   std::ostringstream oStr;
00063   oStr << _tripType << " -- " << _advancePurchase << "-"
00064     << _saturdayStay << "-" << _changeFees << "-"
00065     << _nonRefundable << "-" << _minimumStay;
00066   return oStr.str();
00067 }
00068
00069 }

```

### 33.303 stdair/bom/FareFeaturesKey.hpp File Reference

```

#include <stdair/bom/KeyAbstract.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/stdair_inventory_types.hpp>

```

#### Classes

- struct [stdair::FareFeaturesKey](#)

*Key of date-period.*

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.304 FareFeaturesKey.hpp

```

00001 #ifndef __STDAIR_BOM_FAREFEATURESKEY_HPP
00002 #define __STDAIR_BOM_FAREFEATURESKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/bom/KeyAbstract.hpp>
00009 #include <stdair/stdair_date_time_types.hpp>
00010 #include <stdair/stdair_demand_types.hpp>
00011 #include <stdair/stdair_inventory_types.hpp>
00012
00013 namespace stdair {
00014
00018     struct FareFeaturesKey : public KeyAbstract {
00019     public:
00020         // ////////////////////////////////// Construction //////////////////////////////////
00022         FareFeaturesKey (const TripType_T&, const
DayDuration_T&,
00023                         const SaturdayStay_T&, const ChangeFees_T&,
00024                         const NonRefundable_T&, const DayDuration_T&);
00026         FareFeaturesKey (const FareFeaturesKey&);
00028         ~FareFeaturesKey ();
00029     private:
00031         FareFeaturesKey();
00032
00033     public:
00034         // ////////////////////////////////// Getters //////////////////////////////////
00035         const TripType_T& getTripType() const {
00039             return _tripType;
00040         }
00041     }
00042
00046         const DayDuration_T& getAdvancePurchase() const {
00047             return _advancePurchase;
00048         }
00049
00053         const SaturdayStay_T& getSaturdayStay() const {
00054             return _saturdayStay;
00055         }
00056
00060         const ChangeFees_T& getChangeFees() const {
00061             return _changeFees;
00062         }
00063
00067         const NonRefundable_T& getRefundableOption() const {
00068             return _nonRefundable;
00069         }
00070
00074         const DayDuration_T& getMinimumStay() const {
00075             return _minimumStay;
00076         }
00077
00078     public:
00079         // ////////////////////////////////// Display support methods //////////////////////////////////
00080         void toStream (std::ostream& ioOut) const;
00086
00087         void fromStream (std::istream& ioIn);
00093
00094         const std::string toString() const;
00100
00101     private:
00102         // ////////////////////////////////// Attributes //////////////////////////////////
00103         TripType_T _tripType;
00108
00109         DayDuration_T _advancePurchase;
00113
00114         SaturdayStay_T _saturdayStay;
00118
00119

```

```

00123     ChangeFees_T _changeFees;
00124
00128     NonRefundable_T _nonRefundable;
00129
00133     DayDuration_T _minimumStay;
00134 };
00135
00136 }
00137 #endif // __STDAIR_BOM_FAREFEATURESKEY_HPP

```

### 33.305 stdair/bom/FareFeaturesTypes.hpp File Reference

```

#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>

```

#### Namespaces

- [stdair](#)  
Handle on the StdAir library context.

#### Typedefs

- typedef std::list< FareFeatures \* > [stdair::FareFeaturesList\\_T](#)
- typedef std::map< const MapKey\_T, FareFeatures \* > [stdair::FareFeaturesMap\\_T](#)
- typedef std::pair< MapKey\_T, FareFeatures \* > [stdair::FareFeaturesWithKey\\_T](#)
- typedef std::list< FareFeaturesWithKey\_T > [stdair::FareFeaturesDetailedList\\_T](#)

### 33.306 FareFeaturesTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_FAREFEATURESTYPES_HPP
00003 #define __STDAIR_BOM_FAREFEATURESTYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // STDAIR
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016     // Forward declarations.
00017     class FareFeatures;
00018
00020     typedef std::list<FareFeatures*> FareFeaturesList_T;
00021
00023     typedef std::map<const MapKey_T, FareFeatures*> FareFeaturesMap_T;
00024
00026     typedef std::pair<MapKey_T, FareFeatures*> FareFeaturesWithKey_T;
00027     typedef std::list<FareFeaturesWithKey_T> FareFeaturesDetailedList_T;
00028 }
00029 #endif // __STDAIR_BOM_FAREFEATURESTYPES_HPP
00030

```

### 33.307 stdair/bom/FareOptionStruct.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_BookingClass.hpp>
#include <stdair/bom/FareOptionStruct.hpp>

```

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.308 FareOptionStruct.cpp

```

00001 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00002 // Import section
00003 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_BookingClass.hpp>
00009 #include <stdair/bom/FareOptionStruct.hpp>
00010
00011 namespace stdair {
00012
00013 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00014 FareOptionStruct::FareOptionStruct()
00015 : _fare (DEFAULT_FARE_VALUE), _avl (DEFAULT_AVAILABILITY) {
00016 }
00017
00018 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00019 FareOptionStruct::FareOptionStruct (const
FareOptionStruct& iFO)
00020 : _classPath (iFO._classPath),
00021   _fare (iFO._fare), _avl (iFO._avl), _changeFee (iFO._changeFee),
00022   _nonRefundable (iFO._nonRefundable), _saturdayStay (iFO._saturdayStay) {
00023 }
00024
00025 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00026 FareOptionStruct::FareOptionStruct (const std::string& iClassPath,
00027                                     const Fare_T& iFare,
00028                                     const ChangeFees_T& iChangeFee,
00029                                     const NonRefundable_T& iNonRefundable,
00030                                     const SaturdayStay_T& iSaturdayNightStay)
00031 : _fare (iFare), _avl (DEFAULT_AVAILABILITY),
00032   _changeFee (iChangeFee), _nonRefundable (iNonRefundable),
00033   _saturdayStay (iSaturdayNightStay) {
00034   _classPath.push_back (iClassPath);
00035 }
00036
00037 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00038 FareOptionStruct::~FareOptionStruct () {
00039 }
00040
00041 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00042 void FareOptionStruct::toStream (std::ostream& ioOut) const {
00043   ioOut << describe();
00044 }
00045
00046 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00047 void FareOptionStruct::fromStream (std::istream& ioIn) {
00048 }
00049
00050 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00051 const std::string FareOptionStruct::describe() const {
00052   std::ostringstream oStr;
00053
00054   oStr << "Class path: ";
00055   unsigned short idx = 0;
00056   for (ClassList_StringList_T::const_iterator itClassPath =
00057         _classPath.begin(); itClassPath != _classPath.end();
00058        ++itClassPath, ++idx) {
00059     if (idx != 0) {
00060       oStr << "-";
00061     }
00062     const std::string& lClassPath = *itClassPath;
00063     oStr << lClassPath;
00064   }
00065
00066   oStr << "; " << _fare << " EUR";
00067   oStr << "; conditions: " << _changeFee << " " << _nonRefundable
00068     << " " << _saturdayStay;
00069   return oStr.str();
00070 }
00071
00072 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00073 const std::string FareOptionStruct::display() const {
00074   std::ostringstream oStr;
00075 }

```

```

00076     unsigned short idx = 0;
00077     for (ClassList_StringList_T::const_iterator itClassPath =
00078         _classPath.begin(); itClassPath != _classPath.end();
00079         ++itClassPath, ++idx) {
00080         if (idx != 0) {
00081             ostr << "-";
00082         }
00083         const std::string& lClassPath = *itClassPath;
00084         ostr << lClassPath;
00085     }
00086
00087     ostr << ", " << _fare << ", " << _changeFee << " " << _nonRefundable
00088     << " " << _saturdayStay;
00089     return ostr.str();
00090 }
00091
00092 // //////////////////////////////////////
00093 void FareOptionStruct::addClassList (const std::string iClassCodeList) {
00094     _classPath.push_back (iClassCodeList);
00095 }
00096
00097 // //////////////////////////////////////
00098 void FareOptionStruct::emptyClassList () {
00099     _classPath.clear();
00100 }
00101
00102 }

```

### 33.309 stdair/bom/FareOptionStruct.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/stdair_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/BookingClassTypes.hpp>

```

#### Classes

- struct [stdair::FareOptionStruct](#)  
Structure holding the elements of a fare option.

#### Namespaces

- [stdair](#)  
Handle on the StdAir library context.

### 33.310 FareOptionStruct.hpp

```

00001 #ifndef __STDAIR_BOM_FAREOPTIONSTRUCT_HPP
00002 #define __STDAIR_BOM_FAREOPTIONSTRUCT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_types.hpp>
00012 #include <stdair/basic/StructAbstract.hpp>
00013 #include <stdair/bom/BookingClassTypes.hpp>
00014
00015 namespace stdair {
00016
00020     struct FareOptionStruct : public StructAbstract {
00021     public:
00022         // //////////////////////////////////////
00024         const ClassList_StringList_T& getClassPath() const {
00025             return _classPath;
00026         }

```



```

00027
00029     const Fare_T& getFare() const {
00030         return _fare;
00031     }
00032
00034     const Availability_T& getAvailability() const {
00035         return _avl;
00036     }
00037
00039     const ChangeFees_T getChangeFees() const {
00040         return _changeFee;
00041     }
00042
00044     const NonRefundable_T getNonRefundable() const {
00045         return _nonRefundable;
00046     }
00047
00049     const SaturdayStay_T getSaturdayStay() const {
00050         return _saturdayStay;
00051     }
00052
00053
00054 public:
00055     // //////////// Setters ////////////
00057     void addClassList (const std::string);
00058
00060     void emptyClassList ();
00061
00063     void setFare (const Fare_T& iFare) {
00064         _fare = iFare;
00065     }
00066
00068     void setAvailability (const Availability_T& iAvl) {
00069         _avl = iAvl;
00070     }
00071
00073     void setChangeFees (const ChangeFees_T iRes) {
00074         _changeFee = iRes;
00075     }
00076
00078     void setNonRefundable (const NonRefundable_T iRes) {
00079         _nonRefundable = iRes;
00080     }
00081
00083     void setSaturdayStay (const SaturdayStay_T iRes) {
00084         _saturdayStay = iRes;
00085     }
00086
00087
00088 public:
00089     // //////////// Display support method ////////////
00095     void toStream (std::ostream& ioOut) const;
00096
00102     void fromStream (std::istream& ioIn);
00103
00107     const std::string describe() const;
00108
00112     const std::string display() const;
00113
00114
00115 public:
00116     // //////////// Constructors & Destructor ////////////
00120     FareOptionStruct();
00121
00125     FareOptionStruct (const std::string& iClassPath,
00126                     const Fare_T&, const ChangeFees_T&,
00127                     const NonRefundable_T&, const
00128 SaturdayStay_T&);
00132
00133     FareOptionStruct (const FareOptionStruct&);
00137
00138     ~FareOptionStruct();
00139
00140 private:
00141     // //////////// Attributes ////////////
00145     ClassList_StringList_T _classPath;
00146
00150     Fare_T _fare;
00151
00155     Availability_T _avl;
00156
00160     ChangeFees_T _changeFee;
00161
00165     NonRefundable_T _nonRefundable;
00166
00170     SaturdayStay_T _saturdayStay;

```

```

00171     };
00172
00173 }
00174 #endif // __STDAIR_BOM_FAREOPTIONSTRUCT_HPP

```

### 33.311 stdair/bom/FareOptionTypes.hpp File Reference

```

#include <list>
#include <map>
#include <stdair/stdair_types.hpp>
#include <stdair/bom/key_types.hpp>

```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

#### Typedefs

- typedef std::list< FareOptionStruct > [stdair::FareOptionList\\_T](#)

### 33.312 FareOptionTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_FAREOPTIONTYPES_HPP
00003 #define __STDAIR_BOM_FAREOPTIONTYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <list>
00010 #include <map>
00011 // STDAIR
00012 #include <stdair/stdair_types.hpp>
00013 #include <stdair/bom/key_types.hpp>
00014
00015 namespace stdair {
00016
00017     // Forward declarations.
00018     struct FareOptionStruct;
00019
00021     typedef std::list<FareOptionStruct> FareOptionList_T;
00022
00023 }
00024 #endif // __STDAIR_BOM_FAREOPTIONTYPES_HPP
00025

```

### 33.313 stdair/bom/FFDisutilityCurveHolderStruct.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/FFDisutilityCurveHolderStruct.hpp>

```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.314 FFDisutilityCurveHolderStruct.cpp

```

00001 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00002 // Import section
00003 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/service/Logger.hpp>
00009 #include <stdair/bom/FFDisutilityCurveHolderStruct.hpp>
00010
00011 namespace stdair {
00012
00013     //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00014     FFDisutilityCurveHolderStruct::FFDisutilityCurveHolderStruct
00015     () {
00016     }
00017     //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00018     FFDisutilityCurveHolderStruct::
00019     FFDisutilityCurveHolderStruct (const
00020     FFDisutilityCurveHolderStruct& iHolder)
00021     : _disutilityCurveHolder (iHolder._disutilityCurveHolder) {
00022     }
00023     //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00024     FFDisutilityCurveHolderStruct::~FFDisutilityCurveHolderStruct
00025     () {
00026     }
00027     //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00028     const FFDisutilityCurve_T& FFDisutilityCurveHolderStruct::
00029     getFFDisutilityCurve (const std::string& iKey) const {
00030         FFDisutilityCurveHolder_T::const_iterator itCurve = _disutilityCurveHolder.find (iKey);
00031         if (itCurve == _disutilityCurveHolder.end()) {
00032             STDAIR_LOG_DEBUG ("Cannot find the FFDisutility curve corresponding to the "
00033             << "given key: " << iKey);
00034             assert (false);
00035         }
00036         return itCurve->second;
00037     }
00038     //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00039     void FFDisutilityCurveHolderStruct::
00040     addCurve (const std::string& iKey, const FFDisutilityCurve_T& iCurve) {
00041         bool insert = _disutilityCurveHolder.insert (FFDisutilityCurveHolder_T::value_type(iKey, iCurve)).
00042         second;
00043         if (insert == false) {
00044             STDAIR_LOG_DEBUG ("Cannot add the FFDisutility curve corresponding to the "
00045             << "given key: " << iKey
00046             << ", the key may already exist.");
00047             assert (false);
00048         }
00049     }
00050     //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00051     void FFDisutilityCurveHolderStruct::toStream (std::ostream& ioOut)
00052     const {
00053         ioOut << describe();
00054     }
00055     //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00056     void FFDisutilityCurveHolderStruct::fromStream (std::istream&
00057     ioIn) {
00058     }
00059     //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00060     const std::string FFDisutilityCurveHolderStruct::describe() const
00061     {
00062         std::ostringstream oStr;
00063         for (FFDisutilityCurveHolder_T::const_iterator itCurve = _disutilityCurveHolder.begin();
00064             itCurve != _disutilityCurveHolder.end(); ++itCurve) {
00065             const std::string& lKey = itCurve->first;
00066             const FFDisutilityCurve_T& lCurve = itCurve->second;
00067             oStr << lKey << " ";
00068             for (FFDisutilityCurve_T::const_reverse_iterator itFFDisutility =
00069                 lCurve.rbegin(); itFFDisutility != lCurve.rend(); ++itFFDisutility) {
00070                 const DTD_T& lDTD = itFFDisutility->first;
00071                 const double& lFFDisutility = itFFDisutility->second;
00072                 oStr << lDTD << ":" << lFFDisutility << " ";
00073             }
00074             oStr << std::endl;
00075         }
00076         return oStr.str();
00077     }
00078 }

```

```
00079
00080 }
```

### 33.315 stdair/bom/FFDisutilityCurveHolderStruct.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_rm_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
```

#### Classes

- struct [stdair::FFDisutilityCurveHolderStruct](#)

#### Namespaces

- [stdair](#)  
*Handle on the StdAir library context.*

#### Typedefs

- typedef std::map< const std::string, FFDisutilityCurve\_T > [stdair::FFDisutilityCurveHolder\\_T](#)

### 33.316 FFDisutilityCurveHolderStruct.hpp

```
00001 #ifndef __STDAIR_BOM_FFDisUTILITYCURVEHOLDERSTRUCT_HPP
00002 #define __STDAIR_BOM_FFDisUTILITYCURVEHOLDERSTRUCT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_rm_types.hpp>
00012 #include <stdair/basic/StructAbstract.hpp>
00013
00014 namespace stdair {
00015     // Type definition for the holder of disutility curves.
00016     typedef std::map<const std::string, FFDisutilityCurve_T>
00017         FFDisutilityCurveHolder_T;
00018
00019     struct FFDisutilityCurveHolderStruct : public
00020         StructAbstract {
00021     public:
00022         // ////////////////////////////////// Getters //////////////////////////////////
00023         const FFDisutilityCurve_T& getFFDisutilityCurve (const
00024             std::string&) const;
00025
00026         // ////////////////////////////////// Business Methods //////////////////////////////////
00027         void addCurve (const std::string&, const FFDisutilityCurve_T&);
00028
00029         // ////////////////////////////////// Display support method //////////////////////////////////
00030         void toStream (std::ostream& ioOut) const;
00031
00032         void fromStream (std::istream& ioIn);
00033
00034         const std::string describe() const;
00035
00036         // ////////////////////////////////// Constructors and Destructors //////////////////////////////////
00037     public:
00038         FFDisutilityCurveHolderStruct ();
00039
00040         FFDisutilityCurveHolderStruct (const
00041             FFDisutilityCurveHolderStruct&);
00042
00043     }
```

```

00050     public:
00051         ~FFDisutilityCurveHolderStruct();
00052
00053     private:
00054         // ////////////////////////////////// Attributes //////////////////////////////////
00055         FFDisutilityCurveHolder_T _disutilityCurveHolder;
00056     };
00057
00058 }
00059 #endif // __STDAIR_BOM_FFDISUTILITYCURVEHOLDERSTRUCT_HPP

```

### 33.317 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

- [stdair](#)

*Handle on the StdAir library context.*

### 33.318 FlightDate.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_Inventory.hpp>
00009 #include <stdair/bom/BomManager.hpp>
00010 #include <stdair/bom/Inventory.hpp>
00011 #include <stdair/bom/FlightDate.hpp>
00012 #include <stdair/bom/LegDate.hpp>
00013 #include <stdair/bom/SegmentDate.hpp>
00014
00015 namespace stdair {
00016
00017     // //////////////////////////////////////
00018     FlightDate::FlightDate()
00019         : _key (DEFAULT_FLIGHT_NUMBER, DEFAULT_DEPARTURE_DATE),
00020         _parent (NULL) {
00021         // That constructor is used by the serialisation process
00022     }
00023
00024     // //////////////////////////////////////
00025     FlightDate::FlightDate (const FlightDate& iFlightDate)
00026         : _key (iFlightDate._key), _parent (NULL) {
00027     }
00028
00029     // //////////////////////////////////////
00030     FlightDate::FlightDate (const Key_T& iKey) : _key (iKey), _parent (NULL) {
00031     }
00032
00033     // //////////////////////////////////////
00034     FlightDate::~FlightDate() {
00035     }
00036
00037     // //////////////////////////////////////
00038     const AirlineCode_T& FlightDate::getAirlineCode() const {
00039         const Inventory* lInventory_ptr =
00040             static_cast<const Inventory*> (getParent());
00041         assert (lInventory_ptr != NULL);
00042         return lInventory_ptr->getAirlineCode();
00043     }

```

```

00042     }
00043
00044     // //////////////////////////////////////
00045     std::string FlightDate::toString() const {
00046         std::ostringstream ostr;
00047         ostr << describeKey();
00048         return ostr.str();
00049     }
00050
00051     // //////////////////////////////////////
00052     LegDate* FlightDate::getLegDate (const std::string& iLegDateKeyStr) const {
00053         LegDate* oLegDate_ptr =
00054             BomManager::getObjectPtr<LegDate> (*this, iLegDateKeyStr);
00055         return oLegDate_ptr;
00056     }
00057
00058     // //////////////////////////////////////
00059     LegDate* FlightDate::getLegDate (const LegDateKey& iLegDateKey)
00060     const {
00061         return getLegDate (iLegDateKey.toString());
00062     }
00063
00064     // //////////////////////////////////////
00065     SegmentDate* FlightDate::
00066     getSegmentDate (const std::string& iSegmentDateKeyStr) const {
00067         SegmentDate* oSegmentDate_ptr =
00068             BomManager::getObjectPtr<SegmentDate> (*this, iSegmentDateKeyStr);
00069         return oSegmentDate_ptr;
00070     }
00071
00072     // //////////////////////////////////////
00073     SegmentDate* FlightDate::
00074     getSegmentDate (const SegmentDateKey& iSegmentDateKey) const {
00075         return getSegmentDate (iSegmentDateKey.toString());
00076     }
00077 }
00078

```

### 33.319 stdair/bom/FlightDate.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/FlightDateKey.hpp>
#include <stdair/bom/FlightDateTypes.hpp>

```

#### Classes

- class [stdair::FlightDate](#)  
Class representing the actual attributes for an airline flight-date.

#### Namespaces

- [boost](#)  
Forward declarations.
- [boost::serialization](#)
- [stdair](#)  
Handle on the StdAir library context.

### 33.320 FlightDate.hpp

```

00001 #ifndef __STDAIR_BOM_FLIGHTDATE_HPP
00002 #define __STDAIR_BOM_FLIGHTDATE_HPP
00003
00004 // //////////////////////////////////////

```

```

00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/bom/BomAbstract.hpp>
00013 #include <stdair/bom/FlightDateKey.hpp>
00014 #include <stdair/bom/FlightDateTypes.hpp>
00015
00017 namespace boost {
00018     namespace serialization {
00019         class access;
00020     }
00021 }
00022
00023 namespace stdair {
00024
00026     struct LegDateKey;
00027     class LegDate;
00028     struct SegmentDateKey;
00029     class SegmentDate;
00030
00035     class FlightDate : public BomAbstract {
00036     public:
00037         template <typename BOM> friend class FacBom;
00038         template <typename BOM> friend class FacCloneBom;
00039         friend class FacBomManager;
00040         friend class boost::serialization::access;
00041
00042         // ////////////////////////////////// Type definitions //////////////////////////////////
00043         typedef FlightDateKey Key_T;
00044
00045     public:
00046         // ////////////////////////////////// Getters //////////////////////////////////
00047         const Key_T& getKey() const {
00048             return _key;
00049         }
00050
00051         BomAbstract* getParent() const {
00052             return _parent;
00053         }
00054
00055         const FlightNumber_T& getFlightNumber() const {
00056             return _key.getFlightNumber();
00057         }
00058
00059         const Date_T& getDepartureDate() const {
00060             return _key.getDepartureDate();
00061         }
00062
00063         const AirlineCode_T& getAirlineCode() const;
00064
00065         const HolderMap_T& getHolderMap() const {
00066             return _holderMap;
00067         }
00068
00069         LegDate* getLegDate (const std::string& iLegDateKeyStr) const;
00070
00071         LegDate* getLegDate (const LegDateKey&) const;
00072
00073         SegmentDate* getSegmentDate (const std::string& iSegmentDateKeyStr) const;
00074
00075         SegmentDate* getSegmentDate (const SegmentDateKey&) const;
00076
00077     public:
00078         // ////////////////////////////////// Display support methods //////////////////////////////////
00079         void toStream (std::ostream& ioOut) const {
00080             ioOut << toString();
00081         }
00082
00083         void fromStream (std::istream& ioIn) {
00084
00085         }
00086
00087         std::string toString() const;
00088
00089         const std::string describeKey() const {
00090             return _key.toString();
00091         }
00092
00093     public:
00094         // ////////////////////////////////// (Boost) Serialisation support methods //////////////////////////////////
00095         template<class Archive>
00096         void serialize (Archive& ar, const unsigned int iFileVersion);

```

```

00174
00175 private:
00183     void serialisationImplementationExport() const;
00184     void serialisationImplementationImport();
00185
00186
00187 protected:
00188     // ////////// Constructors and destructors //////////
00192     FlightDate (const Key_T&);
00193
00197     virtual ~FlightDate();
00198
00199 private:
00203     FlightDate();
00204
00208     FlightDate (const FlightDate&);
00209
00210
00211 protected:
00212     // ////////// Attributes //////////
00216     Key_T _key;
00217
00221     BomAbstract* _parent;
00222
00226     HolderMap_T _holderMap;
00227 };
00228
00229 }
00230 #endif // __STDAIR_BOM_FLIGHTDATE_HPP
00231

```

### 33.321 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

- [stdair](#)  
Handle on the StdAir library context.

#### Functions

- template void [stdair::FlightDateKey::serialize< ba::text\\_oarchive >](#) (ba::text\_oarchive &, unsigned int)
- template void [stdair::FlightDateKey::serialize< ba::text\\_iarchive >](#) (ba::text\_iarchive &, unsigned int)

### 33.322 FlightDateKey.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost Date-Time
00008 #include <boost/date_time/gregorian/formatters.hpp>
00009 // Boost.Serialization
00010 #include <boost/archive/text_iarchive.hpp>
00011 #include <boost/archive/text_oarchive.hpp>
00012 #include <boost/serialization/access.hpp>
00013 // StdAir

```



```

00014 #include <stdair/basic/BasConst_Inventory.hpp>
00015 #include <stdair/basic/BasConst_BomDisplay.hpp>
00016 #include <stdair/bom/FlightDateKey.hpp>
00017
00018 namespace stdair {
00019
00020 // //////////////////////////////////////
00021 FlightDateKey::FlightDateKey()
00022 : _flightNumber (DEFAULT_FLIGHT_NUMBER),
00023   _departureDate (DEFAULT_DEPARTURE_DATE) {
00024     assert (false);
00025 }
00026
00027 // //////////////////////////////////////
00028 FlightDateKey::FlightDateKey (const FlightNumber_T& iFlightNumber,
00029                               const Date_T& iFlightDate)
00030 : _flightNumber (iFlightNumber), _departureDate (iFlightDate) {
00031 }
00032
00033 // //////////////////////////////////////
00034 FlightDateKey::FlightDateKey (const FlightDateKey& iKey)
00035 : _flightNumber (iKey._flightNumber), _departureDate (iKey._departureDate) {
00036 }
00037
00038 // //////////////////////////////////////
00039 FlightDateKey::~FlightDateKey() {
00040 }
00041
00042 // //////////////////////////////////////
00043 void FlightDateKey::toStream (std::ostream& ioOut) const {
00044     ioOut << "FlightDateKey: " << toString();
00045 }
00046
00047 // //////////////////////////////////////
00048 void FlightDateKey::fromStream (std::istream& ioIn) {
00049 }
00050
00051 // //////////////////////////////////////
00052 const std::string FlightDateKey::toString() const {
00053     std::ostringstream oStr;
00054     const std::string& lDepartureDateStr =
00055         boost::gregorian::to_iso_extended_string (_departureDate);
00056     oStr << _flightNumber
00057         << DEFAULT_KEY_SUB_FLD_DELIMITER << " " << lDepartureDateStr;
00058     return oStr.str();
00059 }
00060
00061 // //////////////////////////////////////
00062 void FlightDateKey::serialisationImplementationExport() const {
00063     std::ostringstream oStr;
00064     boost::archive::text_oarchive oa (oStr);
00065     oa << *this;
00066 }
00067
00068 // //////////////////////////////////////
00069 void FlightDateKey::serialisationImplementationImport() {
00070     std::istringstream iStr;
00071     boost::archive::text_iarchive ia (iStr);
00072     ia >> *this;
00073 }
00074
00075 // //////////////////////////////////////
00076 template<class Archive>
00077 void FlightDateKey::serialize (Archive& ioArchive,
00078                                const unsigned int iFileVersion) {
00079     std::string lDepartureDateStr =
00080         boost::gregorian::to_simple_string (_departureDate);
00081     ioArchive & _flightNumber & lDepartureDateStr;
00082 }
00083
00084 // //////////////////////////////////////
00085 // Explicit template instantiation
00086 namespace ba = boost::archive;
00087 template void FlightDateKey::serialize<ba::text_oarchive> (ba::text_oarchive&,
00088                                                            unsigned int);
00089 template void FlightDateKey::serialize<ba::text_iarchive> (ba::text_iarchive&,
00090                                                            unsigned int);
00091 // //////////////////////////////////////
00092 }
00093

```

### 33.323 stdair/bom/FlightDateKey.hpp File Reference

```
#include <iosfwd>
```

```
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

## Classes

- struct [stdair::FlightDateKey](#)

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

## Namespaces

- [boost](#)

*Forward declarations.*

- [boost::serialization](#)
- [stdair](#)

*Handle on the StdAir library context.*

## 33.324 FlightDateKey.hpp

```
00001 #ifndef __STDAIR_BOM_FLIGHTDATEKEY_HPP
00002 #define __STDAIR_BOM_FLIGHTDATEKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_basic_types.hpp>
00012 #include <stdair/stdair_date_time_types.hpp>
00013 #include <stdair/bom/KeyAbstract.hpp>
00014
00016 namespace boost {
00017     namespace serialization {
00018         class access;
00019     }
00020 }
00021
00022 namespace stdair {
00023
00028     struct FlightDateKey : public KeyAbstract {
00029         friend class boost::serialization::access;
00030
00031         // ////////// Constructors and destructors //////////
00032     private:
00033         FlightDateKey();
00034
00035     public:
00042         FlightDateKey (const FlightNumber_T&, const
Date_T&);
00043
00047         FlightDateKey (const FlightDateKey&);
00048
00052         ~FlightDateKey();
00053
00054     public:
00056         // ////////// Getters //////////
00058         const FlightNumber_T& getFlightNumber() const {
00059             return _flightNumber;
00060         }
00061
00063         const Date_T& getDepartureDate() const {
00064             return _departureDate;
00065         }
00066
00067     public:
00069         // ////////// Display support methods //////////
00075         void toStream (std::ostream& ioOut) const;
```

```

00076
00082     void fromStream (std::istream& ioIn);
00083
00093     const std::string toString() const;
00094
00095
00096 public:
00097     // ////////// (Boost) Serialisation support methods //////////
00101     template<class Archive>
00102     void serialize (Archive& ar, const unsigned int iFileVersion);
00103
00104 private:
00109     void serialisationImplementationExport() const;
00110     void serialisationImplementationImport();
00111
00112
00113 private:
00114     // ////////////////////////////////// Attributes //////////////////////////////////
00118     FlightNumber_T _flightNumber;
00119
00123     Date_T _departureDate;
00124 };
00125
00126 }
00127 #endif // __STDAIR_BOM_FLIGHTDATEKEY_HPP

```

### 33.325 stdair/bom/FlightDateTypes.hpp File Reference

```

#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>

```

#### Namespaces

- [stdair](#)  
Handle on the StdAir library context.

#### Typedefs

- typedef std::list< FlightDate \* > [stdair::FlightDateList\\_T](#)
- typedef std::map< const MapKey\_T, FlightDate \* > [stdair::FlightDateMap\\_T](#)

### 33.326 FlightDateTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_FLIGHTDATETYPES_HPP
00003 #define __STDAIR_BOM_FLIGHTDATETYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // StdAir
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016     // Forward declarations
00017     class FlightDate;
00018
00019     // ////////////////////////////////// Type definitions //////////////////////////////////
00021     typedef std::list<FlightDate*> FlightDateList_T;
00022
00024     typedef std::map<const MapKey_T, FlightDate*> FlightDateMap_T;
00025
00026 }
00027 #endif // __STDAIR_BOM_FLIGHTDATETYPES_HPP
00028

```

## 33.327 stdair/bom/FlightPeriod.cpp File Reference

```
#include <cassert>
#include <stdair/bom/FlightPeriod.hpp>
```

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.328 FlightPeriod.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // STDAIR
00007 #include <stdair/bom/FlightPeriod.hpp>
00008
00009 namespace stdair {
00010
00011 // //////////////////////////////////////
00012 FlightPeriod::FlightPeriod (const Key_T& iKey)
00013     : _key (iKey), _parent (NULL) {
00014 }
00015
00016 // //////////////////////////////////////
00017 FlightPeriod::FlightPeriod (const FlightPeriod& iFlightPeriod)
00018     : _key (iFlightPeriod.getKey()), _parent (NULL) {
00019 }
00020
00021 // //////////////////////////////////////
00022 FlightPeriod::~FlightPeriod () {
00023 }
00024
00025 // //////////////////////////////////////
00026 std::string FlightPeriod::toString() const {
00027     std::ostringstream ostr;
00028     ostr << describeKey();
00029     return ostr.str();
00030 }
00031
00032 }
00033
```

## 33.329 stdair/bom/FlightPeriod.hpp File Reference

```
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/FlightPeriodKey.hpp>
#include <stdair/bom/FlightPeriodTypes.hpp>
```

## Classes

- class [stdair::FlightPeriod](#)

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.330 FlightPeriod.hpp

```

00001 #ifndef __STDAIR_BOM_FLIGHTPERIOD_HPP
00002 #define __STDAIR_BOM_FLIGHTPERIOD_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STDAIR
00008 #include <stdair/bom/BomAbstract.hpp>
00009 #include <stdair/bom/FlightPeriodKey.hpp>
00010 #include <stdair/bom/FlightPeriodTypes.hpp>
00011
00012 namespace stdair {
00013
00014     class FlightPeriod : public BomAbstract {
00015     public:
00016         template <typename BOM> friend class FacBom;
00017         template <typename BOM> friend class FacCloneBom;
00018         friend class FacBomManager;
00019
00020     public:
00021         // Type definitions.
00022         typedef FlightPeriodKey Key_T;
00023
00024     public:
00025         // ////////////////////////////////// Getters //////////////////////////////////
00026         const Key_T& getKey () const { return _key; }
00027
00028         BomAbstract* const getParent() const { return _parent; }
00029
00030         const FlightNumber_T& getFlightNumber () const {
00031             return _key.getFlightNumber();
00032         }
00033
00034         const PeriodStruct& getPeriod () const { return _key.
00035             getPeriod(); }
00036
00037         const HolderMap_T& getHolderMap() const { return
00038             _holderMap; }
00039
00040     public:
00041         // ////////////////////////////////// Display support methods //////////////////////////////////
00042         void toStream (std::ostream& ioOut) const { ioOut << toString(); }
00043
00044         void fromStream (std::istream& ioIn) { }
00045
00046         std::string toString() const;
00047
00048         const std::string describeKey() const { return _key.toString(); }
00049
00050     protected:
00051         FlightPeriod (const Key_T&);
00052
00053         ~FlightPeriod ();
00054
00055     private:
00056         FlightPeriod ();
00057
00058         FlightPeriod (const FlightPeriod&);
00059
00060     protected:
00061         // Attributes
00062         Key_T _key;
00063         BomAbstract* _parent;
00064         HolderMap_T _holderMap;
00065     };
00066
00067 }
00068 #endif // __STDAIR_BOM_FLIGHTPERIOD_HPP
00069

```

### 33.331 stdair/bom/FlightPeriodKey.cpp File Reference

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

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.332 FlightPeriodKey.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STDAIR
00005 #include <stdair/bom/FlightPeriodKey.hpp>
00006
00007 namespace stdair {
00008
00009     // //////////////////////////////////////
00010     FlightPeriodKey::FlightPeriodKey (const FlightNumber_T& iFlightNumber,
00011                                     const PeriodStruct& iPeriod)
00012         : _flightNumber (iFlightNumber), _period (iPeriod) {}
00013 }
00014
00015     // //////////////////////////////////////
00016     FlightPeriodKey::FlightPeriodKey (const FlightPeriodKey& iKey)
00017         : _flightNumber (iKey._flightNumber), _period (iKey._period) {}
00018 }
00019
00020     // //////////////////////////////////////
00021     FlightPeriodKey::~FlightPeriodKey () {}
00022 }
00023
00024     // //////////////////////////////////////
00025     void FlightPeriodKey::toStream (std::ostream& ioOut) const {
00026         ioOut << "FlightPeriodKey: " << toString() << std::endl;
00027     }
00028
00029     // //////////////////////////////////////
00030     void FlightPeriodKey::fromStream (std::istream& ioIn) {}
00031 }
00032
00033     // //////////////////////////////////////
00034     const std::string FlightPeriodKey::toString() const {
00035         std::ostringstream oStr;
00036         oStr << _flightNumber << ", " << _period.describeShort();
00037         return oStr.str();
00038     }
00039 }
00040 }

```

## 33.333 stdair/bom/FlightPeriodKey.hpp File Reference

```

#include <stdair/bom/KeyAbstract.hpp>
#include <stdair/bom/PeriodStruct.hpp>

```

## Classes

- struct [stdair::FlightPeriodKey](#)

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.334 FlightPeriodKey.hpp

```

00001 #ifndef __STDAIR_BOM_FLIGHTPERIODKEY_HPP
00002 #define __STDAIR_BOM_FLIGHTPERIODKEY_HPP
00003

```

```

00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STDAIR
00008 #include <stdair/bom/KeyAbstract.hpp>
00009 #include <stdair/bom/PeriodStruct.hpp>
00010
00011 namespace stdair {
00013     struct FlightPeriodKey : public KeyAbstract {
00014
00015     private:
00016         // /////////// Default constructor ///////////
00017         FlightPeriodKey ();
00018     public:
00019         // /////////// Construction ///////////
00021         FlightPeriodKey (const FlightNumber_T&, const
PeriodStruct&);
00022         FlightPeriodKey (const FlightPeriodKey&);
00024         ~FlightPeriodKey ();
00025
00026         // /////////// Getters ///////////
00028         const FlightNumber_T& getFlightNumber() const {
00029             return _flightNumber;
00030         }
00031
00033         const PeriodStruct& getPeriod () const {
00034             return _period;
00035         }
00036
00037         // /////////// Display support methods ///////////
00040         void toStream (std::ostream& ioOut) const;
00041
00044         void fromStream (std::istream& ioIn);
00045
00051         const std::string toString() const;
00052
00053     private:
00054         // Attributes
00056         FlightNumber_T _flightNumber;
00057
00059         PeriodStruct _period;
00060
00061     };
00062 }
00063 }
00064 #endif // __STDAIR_BOM_FLIGHTPERIODKEY_HPP

```

### 33.335 stdair/bom/FlightPeriodTypes.hpp File Reference

```

#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>

```

#### Namespaces

- [stdair](#)  
*Handle on the StdAir library context.*

#### Typedefs

- typedef std::list< FlightPeriod \* > [stdair::FlightPeriodList\\_T](#)
- typedef std::map< const MapKey\_T, FlightPeriod \* > [stdair::FlightPeriodMap\\_T](#)

### 33.336 FlightPeriodTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_FLIGHTPERIODTYPES_HPP
00003 #define __STDAIR_BOM_FLIGHTPERIODTYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section

```

```

00007 // //////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // StdAir
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016     // Forward declarations.
00017     class FlightPeriod;
00018
00019     typedef std::list<FlightPeriod*> FlightPeriodList_T;
00020
00021     typedef std::map<const MapKey_T, FlightPeriod*> FlightPeriodMap_T;
00022 }
00023 #endif // __STDAIR_BOM_FLIGHTPERIODTYPES_HPP
00024

```

### 33.337 stdair/bom/FRAT5CurveHolderStruct.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/FRAT5CurveHolderStruct.hpp>

```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.338 FRAT5CurveHolderStruct.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/service/Logger.hpp>
00009 #include <stdair/bom/FRAT5CurveHolderStruct.hpp>
00010
00011 namespace stdair {
00012
00013     // //////////////////////////////////////
00014     FRAT5CurveHolderStruct::FRAT5CurveHolderStruct() {
00015     }
00016
00017     // //////////////////////////////////////
00018     FRAT5CurveHolderStruct::
00019     FRAT5CurveHolderStruct (const FRAT5CurveHolderStruct&
00020     iHolder)
00021     : _frat5CurveHolder (iHolder._frat5CurveHolder) {
00022     }
00023
00024     // //////////////////////////////////////
00025     FRAT5CurveHolderStruct::~FRAT5CurveHolderStruct() {
00026     }
00027
00028     // //////////////////////////////////////
00029     const FRAT5Curve_T& FRAT5CurveHolderStruct::
00030     getFRAT5Curve (const std::string& iKey) const {
00031         FRAT5CurveHolder_T::const_iterator itCurve = _frat5CurveHolder.find (iKey);
00032         if (itCurve == _frat5CurveHolder.end()) {
00033             STDAIR_LOG_DEBUG ("Cannot find the FRAT5 curve corresponding to the "
00034             << "given key: " << iKey);
00035             assert (false);
00036         }
00037         return itCurve->second;
00038     }
00039
00040 // //////////////////////////////////////

```



```

00041 void FRAT5CurveHolderStruct::
00042 addCurve (const std::string& iKey, const FRAT5Curve_T& iCurve) {
00043     bool insert = _frat5CurveHolder.insert (FRAT5CurveHolder_T::value_type(iKey, iCurve)).second;
00044     if (insert == false) {
00045         STDAIR_LOG_DEBUG ("Cannot add the FRAT5 curve corresponding to the "
00046             << "given key: " << iKey
00047             << ", the key may already exist.");
00048         assert (false);
00049     }
00050 }
00051
00052 // //////////////////////////////////////
00053 void FRAT5CurveHolderStruct::toStream (std::ostream& ioOut) const {
00054     ioOut << describe();
00055 }
00056
00057 // //////////////////////////////////////
00058 void FRAT5CurveHolderStruct::fromStream (std::istream& ioIn) {
00059 }
00060
00061 // //////////////////////////////////////
00062 const std::string FRAT5CurveHolderStruct::describe() const {
00063     std::ostringstream oStr;
00064     for (FRAT5CurveHolder_T::const_iterator itCurve = _frat5CurveHolder.begin();
00065         itCurve != _frat5CurveHolder.end(); ++itCurve) {
00066         const std::string& lKey = itCurve->first;
00067         const FRAT5Curve_T& lCurve = itCurve->second;
00068         oStr << lKey << " ";
00069         for (FRAT5Curve_T::const_reverse_iterator itFRAT5 = lCurve.rbegin();
00070             itFRAT5 != lCurve.rend(); ++itFRAT5) {
00071             const DTD_T& lDTD = itFRAT5->first;
00072             const double& lFRAT5 = itFRAT5->second;
00073             oStr << lDTD << ":" << lFRAT5 << " ";
00074         }
00075         oStr << std::endl;
00076     }
00077     return oStr.str();
00078 }
00079
00080 }

```

### 33.339 stdair/bom/FRAT5CurveHolderStruct.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/stdair_rm_types.hpp>
#include <stdair/basic/StructAbstract.hpp>

```

#### Classes

- struct [stdair::FRAT5CurveHolderStruct](#)

#### Namespaces

- [stdair](#)  
*Handle on the StdAir library context.*

#### Typedefs

- typedef std::map< const std::string, FRAT5Curve\_T > [stdair::FRAT5CurveHolder\\_T](#)

### 33.340 FRAT5CurveHolderStruct.hpp

```

00001 #ifndef __STDAIR_BOM_FRAT5CURVEHOLDERSTRUCT_HPP
00002 #define __STDAIR_BOM_FRAT5CURVEHOLDERSTRUCT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section

```

```

00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_rm_types.hpp>
00012 #include <stdair/basic/StructAbstract.hpp>
00013
00014 namespace stdair {
00015     // Type definition for the holder of Frat5 curves.
00016     typedef std::map<const std::string, FRAT5Curve_T> FRAT5CurveHolder_T;
00017
00018     struct FRAT5CurveHolderStruct : public StructAbstract {
00019     public:
00020         // ////////////////////////////////// Getters //////////////////////////////////
00021         const FRAT5Curve_T& getFRAT5Curve (const std::string&) const;
00022
00023         // ////////////////////////////////// Business Methods //////////////////////////////////
00024         void addCurve (const std::string&, const FRAT5Curve_T&);
00025
00026         // ////////////////////////////////// Display support method //////////////////////////////////
00027         void toStream (std::ostream& ioOut) const;
00028
00029         void fromStream (std::istream& ioIn);
00030
00031         const std::string describe() const;
00032
00033         // ////////////////////////////////// Constructors and Destructors //////////////////////////////////
00034     public:
00035         FRAT5CurveHolderStruct ();
00036
00037         FRAT5CurveHolderStruct (const FRAT5CurveHolderStruct&);
00038
00039     public:
00040         ~FRAT5CurveHolderStruct ();
00041
00042     private:
00043         // ////////////////////////////////// Attributes //////////////////////////////////
00044         FRAT5CurveHolder_T _frat5CurveHolder;
00045     };
00046 }
00047 #endif // __STDAIR_BOM_FRAT5CURVEHOLDERSTRUCT_HPP

```

### 33.341 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

- [stdair](#)

*Handle on the StdAir library context.*

### 33.342 Inventory.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_Inventory.hpp>
00009 #include <stdair/bom/BomManager.hpp>
00010 #include <stdair/bom/Inventory.hpp>
00011 #include <stdair/bom/FlightDate.hpp>

```

```

00012
00013 namespace stdair {
00014
00015 // //////////////////////////////////////
00016 Inventory::Inventory() :
00017     _key (DEFAULT_AIRLINE_CODE),
00018     _parent (NULL),
00019     _airlineFeature (NULL) {
00020     // That constructor is used by the serialisation process
00021 }
00022
00023 // //////////////////////////////////////
00024 Inventory::Inventory (const Inventory& iInventory)
00025 : _key (iInventory._key),
00026   _parent (NULL),
00027   _airlineFeature (NULL) {
00028 }
00029
00030 // //////////////////////////////////////
00031 Inventory::Inventory (const Key_T& iKey) :
00032     _key (iKey),
00033     _parent (NULL),
00034     _airlineFeature (NULL) {
00035 }
00036
00037 // //////////////////////////////////////
00038 Inventory::~Inventory() {
00039 }
00040
00041 // //////////////////////////////////////
00042 std::string Inventory::toString() const {
00043     std::ostringstream oStr;
00044     oStr << describeKey();
00045     return oStr.str();
00046 }
00047
00048 // //////////////////////////////////////
00049 FlightDate* Inventory::
00050 getFlightDate (const std::string& iFlightDateKeyStr) const {
00051     FlightDate* oFlightDate_ptr =
00052         BomManager::getObjectPtr<FlightDate> (*this, iFlightDateKeyStr);
00053     return oFlightDate_ptr;
00054 }
00055
00056 // //////////////////////////////////////
00057 FlightDate* Inventory::
00058 getFlightDate (const FlightDateKey& iFlightDateKey) const {
00059     return getFlightDate (iFlightDateKey.toString());
00060 }
00061
00062 // //////////////////////////////////////
00063 ForecastingMethod::EN_ForecastingMethod
Inventory::
00064 getForecastingMethod() const {
00065     assert (_airlineFeature != NULL);
00066     return _airlineFeature->getForecastingMethod();
00067 }
00068
00069 // //////////////////////////////////////
00070 UnconstrainingMethod::EN_UnconstrainingMethod
Inventory::
00071 getUnconstrainingMethod() const {
00072     assert (_airlineFeature != NULL);
00073     return _airlineFeature->getUnconstrainingMethod();
00074 }
00075
00076 // //////////////////////////////////////
00077 PreOptimisationMethod::EN_PreOptimisationMethod
Inventory::
00078 getPreOptimisationMethod() const {
00079     assert (_airlineFeature != NULL);
00080     return _airlineFeature->getPreOptimisationMethod();
00081 }
00082
00083 // //////////////////////////////////////
00084 OptimisationMethod::EN_OptimisationMethod
Inventory::
00085 getOptimisationMethod() const {
00086     assert (_airlineFeature != NULL);
00087     return _airlineFeature->getOptimisationMethod();
00088 }
00089
00090 // //////////////////////////////////////
00091 PartnershipTechnique::EN_PartnershipTechnique
Inventory::
00092 getPartnershipTechnique() const {
00093     assert (_airlineFeature != NULL);

```

```

00094     return _airlineFeature->getPartnershipTechnique();
00095 }
00096
00097 }
00098

```

### 33.343 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>

```

#### Classes

- class [stdair::Inventory](#)  
*Class representing the actual attributes for an airline inventory.*

#### Namespaces

- [boost](#)  
*Forward declarations.*
- [boost::serialization](#)
- [stdair](#)  
*Handle on the StdAir library context.*

### 33.344 Inventory.hpp

```

00001 #ifndef __STDAIR_BOM_INVENTORY_HPP
00002 #define __STDAIR_BOM_INVENTORY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/basic/BasConst_Inventory.hpp>
00013 #include <stdair/bom/BomAbstract.hpp>
00014 #include <stdair/bom/AirlineFeature.hpp>
00015 #include <stdair/bom/InventoryKey.hpp>
00016 #include <stdair/bom/InventoryTypes.hpp>
00017
00019 namespace boost {
00020     namespace serialization {
00021         class access;
00022     }
00023 }
00024
00025 namespace stdair {
00026
00028     struct FlightDateKey;
00029     class FlightDate;
00030
00034     class Inventory : public BomAbstract {
00035     template <typename BOM> friend class FacBom;
00036     template <typename BOM> friend class FacCloneBom;
00037     friend class FacBomManager;
00038     friend class boost::serialization::access;
00039

```

```

00040 public :
00041 // ////////// Type definitions //////////
00045 typedef InventoryKey Key_T;
00046
00047
00048 public:
00049 // ////////// Getters //////////
00051 const Key_T& getKey() const {
00052     return _key;
00053 }
00054
00056 const AirlineCode_T& getAirlineCode() const {
00057     return _key.getAirlineCode();
00058 }
00059
00061 ForecastingMethod::EN_ForecastingMethod
getForecastingMethod() const;
00062
00064 UnconstrainingMethod::EN_UnconstrainingMethod
getUnconstrainingMethod() const;
00065
00067 PreOptimisationMethod::EN_PreOptimisationMethod
getPreOptimisationMethod() const;
00068
00070 OptimisationMethod::EN_OptimisationMethod
getOptimisationMethod() const;
00071
00073 PartnershipTechnique::EN_PartnershipTechnique
getPartnershipTechnique() const;
00074
00076 BomAbstract* const getParent() const {
00077     return _parent;
00078 }
00079
00081 const HolderMap_T& getHolderMap() const {
00082     return _holderMap;
00083 }
00084
00095 FlightDate* getFlightDate (const std::string& iFlightDateKeyStr) const;
00096
00107 FlightDate* getFlightDate (const FlightDateKey&) const;
00108
00112 AirlineFeature* getAirlineFeature () const {
00113     return _airlineFeature;
00114 }
00115
00116
00117 private:
00118 // ////////// Setters //////////
00120 void setAirlineFeature (AirlineFeature& iAirlineFeature) {
00121     _airlineFeature = &iAirlineFeature;
00122 }
00123
00124
00125 public:
00126 // ////////// Display support methods //////////
00132 void toStream (std::ostream& ioOut) const {
00133     ioOut << toString();
00134 }
00135
00141 void fromStream (std::istream& ioIn) {
00142 }
00143
00147 std::string toString() const;
00148
00152 const std::string describeKey() const {
00153     return _key.toString();
00154 }
00155
00156
00157 public:
00158 // ////////// (Boost) Serialisation support methods //////////
00162 template<class Archive>
00163 void serialize (Archive& ar, const unsigned int iFileVersion);
00164
00165 private:
00173 void serialisationImplementationExport() const;
00174 void serialisationImplementationImport();
00175
00176
00177 protected:
00178 // ////////// Constructors and destructors //////////
00182 Inventory (const Key_T&);
00186 ~Inventory();
00187
00188 private:
00192 Inventory();

```

```

00196     Inventory (const Inventory&);
00197
00198
00199 protected:
00200     // ////////// Attributes //////////
00204     Key_T _key;
00205
00209     BomAbstract* _parent;
00210
00214     AirlineFeature* _airlineFeature;
00215
00219     HolderMap_T _holderMap;
00220 };
00221
00222 }
00223 #endif // __STDAIR_BOM_INVENTORY_HPP
00224

```

### 33.345 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

- [stdair](#)  
*Handle on the StdAir library context.*

#### Functions

- template void [stdair::InventoryKey::serialize< ba::text\\_oarchive >](#) (ba::text\_oarchive &, unsigned int)
- template void [stdair::InventoryKey::serialize< ba::text\\_iarchive >](#) (ba::text\_iarchive &, unsigned int)

### 33.346 InventoryKey.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/access.hpp>
00011 // StdAir
00012 #include <stdair/basic/BasConst_Inventory.hpp>
00013 #include <stdair/bom/InventoryKey.hpp>
00014
00015 namespace stdair {
00016
00017     // //////////////////////////////////////
00018     InventoryKey::InventoryKey() : _airlineCode (DEFAULT_AIRLINE_CODE) {
00019         assert (false);
00020     }
00021
00022     // //////////////////////////////////////
00023     InventoryKey::InventoryKey (const AirlineCode_T& iAirlineCode)
00024         : _airlineCode (iAirlineCode) {
00025     }
00026
00027     // //////////////////////////////////////
00028     InventoryKey::InventoryKey (const InventoryKey& iKey)
00029         : _airlineCode (iKey._airlineCode) {

```

```

00030     }
00031
00032     // //////////////////////////////////////
00033     InventoryKey::~InventoryKey() {
00034     }
00035
00036     // //////////////////////////////////////
00037     void InventoryKey::toStream (std::ostream& ioOut) const {
00038         ioOut << "InventoryKey: " << toString();
00039     }
00040
00041     // //////////////////////////////////////
00042     void InventoryKey::fromStream (std::istream& ioIn) {
00043     }
00044
00045     // //////////////////////////////////////
00046     const std::string InventoryKey::toString() const {
00047         std::ostringstream ostr;
00048         ostr << _airlineCode;
00049         return ostr.str();
00050     }
00051
00052     // //////////////////////////////////////
00053     void InventoryKey::serialisationImplementationExport() const {
00054         std::ostringstream ostr;
00055         boost::archive::text_oarchive oa (ostr);
00056         oa << *this;
00057     }
00058
00059     // //////////////////////////////////////
00060     void InventoryKey::serialisationImplementationImport() {
00061         std::istringstream istr;
00062         boost::archive::text_iarchive ia (istr);
00063         ia >> *this;
00064     }
00065
00066     // //////////////////////////////////////
00067     template<class Archive>
00068     void InventoryKey::serialize (Archive& ioArchive,
00069                                   const unsigned int iFileVersion) {
00070         ioArchive & _airlineCode;
00071     }
00072
00073     // //////////////////////////////////////
00074     // Explicit template instantiation
00075     namespace ba = boost::archive;
00076     template void InventoryKey::serialize<ba::text_oarchive> (ba::text_oarchive&,
00077                                                             unsigned int);
00078     template void InventoryKey::serialize<ba::text_iarchive> (ba::text_iarchive&,
00079                                                             unsigned int);
00080     // //////////////////////////////////////
00081
00082 }

```

### 33.347 stdair/bom/InventoryKey.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>

```

#### Classes

- struct [stdair::InventoryKey](#)  
Key of a given inventory, made of the airline code.

#### Namespaces

- [boost](#)  
Forward declarations.
- [boost::serialization](#)
- [stdair](#)

*Handle on the StdAir library context.*

### 33.348 InventoryKey.hpp

```

00001 #ifndef __STDAIR_BOM_INVENTORYKEY_HPP
00002 #define __STDAIR_BOM_INVENTORYKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/bom/KeyAbstract.hpp>
00013
00015 namespace boost {
00016     namespace serialization {
00017         class access;
00018     }
00019 }
00020
00021 namespace stdair {
00022
00026     struct InventoryKey : public KeyAbstract {
00027         friend class boost::serialization::access;
00028
00029         // ////////// Constructors and destructors //////////
00030     private:
00031         InventoryKey();
00032
00033     public:
00034         // ////////// Construction //////////
00035         InventoryKey (const AirlineCode_T& iAirlineCode);
00036
00037         InventoryKey (const InventoryKey&);
00038
00039         ~InventoryKey();
00040
00041         // ////////// Getters //////////
00042         const AirlineCode_T& getAirlineCode() const {
00043             return _airlineCode;
00044         }
00045
00046     public:
00047         // ////////// Display support methods //////////
00048         void toStream (std::ostream& ioOut) const;
00049
00050         void fromStream (std::istream& ioIn);
00051
00052         const std::string toString() const;
00053
00054     public:
00055         // ////////// (Boost) Serialisation support methods //////////
00056         template<class Archive>
00057         void serialize (Archive& ar, const unsigned int iFileVersion);
00058
00059     private:
00060         void serialisationImplementationExport() const;
00061         void serialisationImplementationImport();
00062
00063     private:
00064         // ////////// Attributes //////////
00065         AirlineCode_T _airlineCode;
00066     };
00067 }
00068 #endif // __STDAIR_BOM_INVENTORYKEY_HPP

```

### 33.349 stdair/bom/InventoryTypes.hpp File Reference

```

#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>

```



## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## Typedefs

- typedef std::list< Inventory \* > [stdair::InventoryList\\_T](#)
- typedef std::map< const MapKey\_T, Inventory \* > [stdair::InventoryMap\\_T](#)

### 33.350 InventoryTypes.hpp

```
00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_INVENTORYTYPES_HPP
00003 #define __STDAIR_BOM_INVENTORYTYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // Stdair
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016     // Forward declarations.
00017     class Inventory;
00018
00020     typedef std::list<Inventory*> InventoryList_T;
00021
00023     typedef std::map<const MapKey_T, Inventory*> InventoryMap_T;
00024
00025 }
00026 #endif // __STDAIR_BOM_INVENTORYTYPES_HPP
```

### 33.351 stdair/bom/key\_types.hpp File Reference

```
#include <string>
#include <list>
```

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## Typedefs

- typedef std::string [stdair::MapKey\\_T](#)
- typedef std::list< std::string > [stdair::KeyList\\_T](#)

### 33.352 key\_types.hpp

```
00001 #ifndef __STDAIR_BOM_KEY_TYPES_HPP
00002 #define __STDAIR_BOM_KEY_TYPES_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
```

```

00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <list>
00010
00011 namespace stdair {
00012
00013 // ////////////////////////////////// Type definitions //////////////////////////////////
00015 typedef std::string MapKey_T;
00016
00018 typedef std::list<std::string> KeyList_T;
00019
00020 }
00021 #endif // __STDAIR_BOM_KEY_TYPES_HPP

```

### 33.353 stdair/bom/KeyAbstract.hpp File Reference

```

#include <iosfwd>
#include <string>

```

#### Classes

- struct [stdair::KeyAbstract](#)  
Base class for the keys of Business Object Model (BOM) layer.

#### Namespaces

- [stdair](#)  
Handle on the StdAir library context.

#### Functions

- template<class charT, class traits >  
std::basic\_ostream< charT, traits > & [operator<<](#) (std::basic\_ostream< charT, traits > &ioOut, const [stdair::KeyAbstract](#) &iKey)
- template<class charT, class traits >  
std::basic\_istream< charT, traits > & [operator>>](#) (std::basic\_istream< charT, traits > &ioIn, [stdair::KeyAbstract](#) &ioKey)

#### 33.353.1 Function Documentation

**33.353.1.1** 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](#).

**33.353.1.2** 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\(\)](#).

### 33.354 KeyAbstract.hpp

```

00001
00007 #ifndef __STDAIR_BOM_KEYABSTRACT_HPP
00008 #define __STDAIR_BOM_KEYABSTRACT_HPP
00009
00010 // //////////////////////////////////////
00011 // Import section
00012 // //////////////////////////////////////
00013 // STL
00014 #include <iosfwd>
00015 #include <string>
00016
00017 namespace stdair {
00018
00027     struct KeyAbstract {
00028     public:
00029
00030         // ////////////////////////////////// Display support methods //////////////////////////////////
00036         virtual void toStream (std::ostream& ioOut) const {}
00037
00043         virtual void fromStream (std::istream& ioIn) {}
00044
00056         virtual const std::string toString() const { return std::string("Hello!"); }
00057
00061         virtual ~KeyAbstract() {}
00062     };
00063
00064 }
00065
00071 template <class charT, class traits>
00072 inline
00073 std::basic_ostream<charT, traits>&
00074 operator<< (std::basic_ostream<charT, traits>& ioOut,
00075           const stdair::KeyAbstract& iKey) {
00081     std::basic_ostringstream<charT,traits> ostr;
00082     ostr.copyfmt (ioOut);
00083     ostr.width (0);
00084
00085     // Fill string stream
00086     iKey.toStream (ostr);
00087
00088     // Print string stream
00089     ioOut << ostr.str();
00090
00091     return ioOut;
00092 }
00093
00099 template <class charT, class traits>
00100 inline
00101 std::basic_istream<charT, traits>&
00102 operator>> (std::basic_istream<charT, traits>& ioIn,
00103            stdair::KeyAbstract& ioKey) {
00104     // Fill Key object with input stream
00105     ioKey.fromStream (ioIn);
00106     return ioIn;
00107 }
00108
00109 #endif // __STDAIR_BOM_KEYABSTRACT_HPP

```

### 33.355 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

- [stdair](#)

*Handle on the StdAir library context.*

## 33.356 LegCabin.cpp

```

00001 ///////////////////////////////////////////////////////////////////
00002 // Import section
00003 ///////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_BookingClass.hpp>
00009 #include <stdair/basic/BasConst_Inventory.hpp>
00010 #include <stdair/basic/BasConst_BomDisplay.hpp>
00011 #include <stdair/bom/BomManager.hpp>
00012 #include <stdair/bom/LegDate.hpp>
00013 #include <stdair/bom/LegCabin.hpp>
00014
00015
00016 namespace stdair {
00017
00018 ///////////////////////////////////////////////////////////////////
00019 LegCabin::LegCabin() : _key (DEFAULT_CABIN_CODE), _parent (NULL) {
00020     assert (false);
00021 }
00022
00023 ///////////////////////////////////////////////////////////////////
00024 LegCabin::LegCabin (const LegCabin& iLegCabin)
00025 : _key (iLegCabin._key), _parent (NULL),
00026   _offeredCapacity (iLegCabin._offeredCapacity),
00027   _physicalCapacity (iLegCabin._physicalCapacity),
00028   _soldSeat (iLegCabin._soldSeat),
00029   _committedSpace (iLegCabin._committedSpace),
00030   _availabilityPool (iLegCabin._availabilityPool),
00031   _availability (iLegCabin._availability),
00032   _currentBidPrice (iLegCabin._currentBidPrice),
00033   _dcsRegrade (iLegCabin._dcsRegrade),
00034   _au (iLegCabin._au),
00035   _upr (iLegCabin._upr),
00036   _nav (iLegCabin._nav),
00037   _gav (iLegCabin._gav),
00038   _acp (iLegCabin._acp),
00039   _etb (iLegCabin._etb),
00040   _staffNbOfBookings (iLegCabin._staffNbOfBookings),
00041   _wlNbOfBookings (iLegCabin._wlNbOfBookings),
00042   _groupNbOfBookings (iLegCabin._groupNbOfBookings) {
00043 }
00044
00045 ///////////////////////////////////////////////////////////////////
00046 LegCabin::LegCabin (const Key_T& iKey)
00047 : _key (iKey), _parent (NULL),
00048   _offeredCapacity (DEFAULT_CABIN_CAPACITY),
00049   _physicalCapacity (DEFAULT_CABIN_CAPACITY),
00050   _soldSeat (DEFAULT_CLASS_NB_OF_BOOKINGS),
00051   _committedSpace (DEFAULT_COMMITTED_SPACE),
00052   _availabilityPool (DEFAULT_AVAILABILITY),
00053   _availability (DEFAULT_AVAILABILITY),
00054   _currentBidPrice (DEFAULT_BID_PRICE),
00055   _bidPriceVector (DEFAULT_BID_PRICE_VECTOR),
00056   _dcsRegrade (DEFAULT_NULL_CAPACITY_ADJUSTMENT),
00057   _au (DEFAULT_CLASS_AUTHORIZATION_LEVEL),
00058   _upr (DEFAULT_NULL_UPR),
00059   _nav (DEFAULT_AVAILABILITY),
00060   _gav (DEFAULT_AVAILABILITY),
00061   _acp (DEFAULT_CLASS_OVERBOOKING_RATE),
00062   _etb (DEFAULT_NULL_BOOKING_NUMBER),
00063   _staffNbOfBookings (DEFAULT_NULL_BOOKING_NUMBER),
00064   _wlNbOfBookings (DEFAULT_NULL_BOOKING_NUMBER),
00065   _groupNbOfBookings (DEFAULT_NULL_BOOKING_NUMBER) {
00066 }
00067
00068 ///////////////////////////////////////////////////////////////////
00069 LegCabin::~LegCabin() {
00070 }
00071
00072 ///////////////////////////////////////////////////////////////////
00073 void LegCabin::setCapacities (const CabinCapacity_T& iCapacity) {
00074     _offeredCapacity = iCapacity;
00075     _physicalCapacity = iCapacity;
00076     setAvailabilityPool (iCapacity - _committedSpace);
00077 }
00078
00079 ///////////////////////////////////////////////////////////////////
00080 const MapKey_T LegCabin::getFullerKey() const {
00081     const LegDate& lLegDate = BomManager::getParent<LegDate> (*this);
00082
00083     const MapKey_T oFullKey =
00084         lLegDate.describeKey() + DEFAULT_KEY_FLD_DELIMITER +
00085         getCabinCode();

```

```

00085     return oFullKey;
00086 }
00087
00088 // //////////////////////////////////////
00089 std::string LegCabin::toString() const {
00090     std::ostringstream oStr;
00091     oStr << describeKey();
00092     return oStr.str();
00093 }
00094
00095 // //////////////////////////////////////
00096 const std::string LegCabin::displayVirtualClassList () const {
00097     std::ostringstream oStr;
00098
00099     for (VirtualClassList_T::const_iterator itVC = _virtualClassList.begin();
00100          itVC != _virtualClassList.end(); ++itVC) {
00101         const VirtualClassStruct& lVC = *itVC;
00102         oStr << std::endl << "Yield: " << std::fixed << std::setprecision (2)
00103             << lVC.getYield()
00104             << ", Protection: " << std::fixed << std::setprecision (2)
00105             << lVC.getCumulatedProtection()
00106             << ", Booking limit: " << std::fixed << std::setprecision (2)
00107             << lVC.getCumulatedBookingLimit();
00108     }
00109
00110     return oStr.str();
00111 }
00112
00113 // //////////////////////////////////////
00114 void LegCabin::updateFromReservation (const
NbOfBookings_T& iNbOfBookings) {
00115     _committedSpace += iNbOfBookings;
00116     _availabilityPool = _offeredCapacity -
_committedSpace;
00117 }
00118
00119 // //////////////////////////////////////
00120 void LegCabin::updateCurrentBidPrice() {
00121     const unsigned short lAvailabilityPool =
00122         static_cast<unsigned short> (std::floor (_availabilityPool));
00123
00124     if (lAvailabilityPool >= 1) {
00125         const unsigned short lBidPriceVectorSize = _bidPriceVector.size();
00126         if (lBidPriceVectorSize >= lAvailabilityPool) {
00127             _currentBidPrice = _bidPriceVector.at (lAvailabilityPool - 1);
00128         }
00129     }
00130 }
00131
00132 // //////////////////////////////////////
00133 void LegCabin::addDemandInformation (const
YieldValue_T& iYield,
                                const MeanValue_T& iMeanValue,
                                const StdDevValue_T& iStdDevValue) {
00134
00135     //
00136     const int lYieldLevel =
00137         static_cast<int> (std::floor (iYield + 0.5));
00138
00139     //
00140     YieldLevelDemandMap_T::iterator itDemand =
00141         _yieldLevelDemandMap.find (lYieldLevel);
00142
00143     if (itDemand == _yieldLevelDemandMap.end()) {
00144         MeanStdDevPair_T lMeanStdDevPair (iMeanValue, iStdDevValue);
00145         const bool hasInsertBeenSuccessful = _yieldLevelDemandMap.
00146             insert (YieldLevelDemandMap_T::value_type (lYieldLevel,
00147                                                         lMeanStdDevPair)).second;
00148         assert (hasInsertBeenSuccessful == true);
00149     } else {
00150         //
00151         MeanStdDevPair_T& lMeanStdDevPair = itDemand->second;
00152         MeanValue_T lMeanValue = iMeanValue + lMeanStdDevPair.first;
00153         StdDevValue_T lStdDevValue = iStdDevValue * iStdDevValue + lMeanStdDevPair.second *
lMeanStdDevPair.second;
00154         lStdDevValue = std::sqrt (lStdDevValue);
00155
00156         //
00157         lMeanStdDevPair = MeanStdDevPair_T (lMeanValue, lStdDevValue);
00158     }
00159 }
00160 }
00161 }
00162
00163 }
00164

```

## 33.357 stdair/bom/LegCabin.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/LegCabinKey.hpp>
#include <stdair/bom/LegCabinTypes.hpp>
#include <stdair/bom/VirtualClassStruct.hpp>
#include <stdair/bom/VirtualClassTypes.hpp>
```

## Classes

- class [stdair::LegCabin](#)  
Class representing the actual attributes for an airline leg-cabin.

## Namespaces

- [stdair](#)  
Handle on the StdAir library context.

## 33.358 LegCabin.hpp

```
00001 #ifndef __STDAIR_BOM_LEGCABIN_HPP
00002 #define __STDAIR_BOM_LEGCABIN_HPP
00003 // //////////////////////////////////////
00004 // Import section
00005 // //////////////////////////////////////
00006 // STL
00007 #include <iosfwd>
00008 #include <string>
00009 // StdAir
00010 #include <stdair/stdair_inventory_types.hpp>
00011 #include <stdair/stdair_maths_types.hpp>
00012 #include <stdair/bom/BomAbstract.hpp>
00013 #include <stdair/bom/LegCabinKey.hpp>
00014 #include <stdair/bom/LegCabinTypes.hpp>
00015 #include <stdair/bom/VirtualClassStruct.hpp>
00016 #include <stdair/bom/VirtualClassTypes.hpp>
00017
00018
00019 namespace stdair {
00020
00025     class LegCabin : public BomAbstract {
00026     public:
00027         template <typename BOM> friend class FacBom;
00028         template <typename BOM> friend class FacCloneBom;
00029         friend class FacBomManager;
00030
00031         // ////////////////////////////////// Type definitions //////////////////////////////////
00032         typedef LegCabinKey Key_T;
00033
00034     public:
00035         // ////////////////////////////////// Getters //////////////////////////////////
00036         const Key_T& getKey() const {
00037             return _key;
00038         }
00039
00040         BomAbstract* const getParent() const {
00041             return _parent;
00042         }
00043
00044         const CabinCode_T& getCabinCode() const {
00045             return _key.getCabinCode();
00046         }
00047
00048         const MapKey_T getFullerKey() const;
00049
00050         const HolderMap_T& getHolderMap() const {
```

```

00073         return _holderMap;
00074     }
00075
00077     const CabinCapacity_T& getOfferedCapacity() const {
00078         return _offeredCapacity;
00079     }
00080
00082     const CabinCapacity_T& getPhysicalCapacity() const {
00083         return _physicalCapacity;
00084     }
00085
00087     const NbOfSeats_T& getSoldSeat() const {
00088         return _soldSeat;
00089     }
00090
00092     const CommittedSpace_T& getCommittedSpace() const {
00093         return _committedSpace;
00094     }
00095
00097     const Availability_T& getAvailabilityPool() const {
00098         return _availabilityPool;
00099     }
00100
00102     const Availability_T& getAvailability() const {
00103         return _availability;
00104     }
00105
00107     const BidPrice_T& getCurrentBidPrice() const {
00108         return _currentBidPrice;
00109     }
00110
00112     const BidPrice_T& getPreviousBidPrice() const {
00113         return _previousBidPrice;
00114     }
00115
00117     const BidPriceVector_T& getBidPriceVector() const {
00118         return _bidPriceVector;
00119     }
00120
00122     const CapacityAdjustment_T& getRegradeAdjustment() const {
00123         return _dcsRegrade;
00124     }
00125
00127     const AuthorizationLevel_T& getAuthorizationLevel() const {
00128         return _au;
00129     }
00130
00132     const UPR_T& getUPR() const {
00133         return _upr;
00134     }
00135
00137     const Availability_T& getNetAvailability() const {
00138         return _nav;
00139     }
00140
00142     const Availability_T& getGrossAvailability() const {
00143         return _gav;
00144     }
00145
00147     const OverbookingRate_T& getAvgCancellationPercentage()
const {
00148         return _acp;
00149     }
00150
00152     const NbOfSeats_T& getETB() const {
00153         return _etb;
00154     }
00155
00157     const NbOfSeats_T& getStaffNbOfSeats() const {
00158         return _staffNbOfBookings;
00159     }
00160
00162     const NbOfSeats_T& getWLNbOfSeats() const {
00163         return _wlNbOfBookings;
00164     }
00165
00167     const NbOfSeats_T& getGroupNbOfSeats() const {
00168         return _groupNbOfBookings;
00169     }
00170
00172     VirtualClassList_T& getVirtualClassList() {
00173         return _virtualClassList;
00174     }
00175
00177     BidPriceVector_T& getBidPriceVector() {
00178         return _bidPriceVector;
00179     }

```

```

00180
00181
00183     const YieldLevelDemandMap_T& getYieldLevelDemandMap() {
00184         return _yieldLevelDemandMap;
00185     }
00186
00187
00188 public:
00189     // ////////////////////////////////// Setters //////////////////////////////////
00191     void setCapacities (const CabinCapacity_T& iCapacity);
00192
00194     void setSoldSeat (const NbOfSeats_T& iSoldSeat) {
00195         _soldSeat = iSoldSeat;
00196     }
00197
00199     void setCommittedSpace (const CommittedSpace_T& iCommittedSpace) {
00200         _committedSpace = iCommittedSpace;
00201     }
00202
00204     void setAvailabilityPool (const Availability_T& iAvailabilityPool) {
00205         _availabilityPool = iAvailabilityPool;
00206     }
00207
00209     void setAvailability (const Availability_T& iAvailability) {
00210         _availability = iAvailability;
00211     }
00212
00214     void setCurrentBidPrice (const BidPrice_T& iBidPrice) {
00215         _currentBidPrice = iBidPrice;
00216     }
00217
00219     void setPreviousBidPrice (const BidPrice_T& iBidPrice) {
00220         _previousBidPrice = iBidPrice;
00221     }
00222
00224     void updatePreviousBidPrice () {
00225         _previousBidPrice = _currentBidPrice;
00226     }
00227
00229     void setRegradeAdjustment (const CapacityAdjustment_T&
00230 iRegradeAdjustment) {
00231         _dcsRegrade = iRegradeAdjustment;
00232     }
00233
00234     void setAuthorizationLevel (const AuthorizationLevel_T& iAU) {
00235         _au = iAU;
00236     }
00237
00239     void setUPR (const UPR_T& iUPR) {
00240         _upr = iUPR;
00241     }
00242
00244     void setNetAvailability (const Availability_T& iNAV) {
00245         _nav = iNAV;
00246     }
00247
00249     void setGrossAvailability (const Availability_T& iGAV) {
00250         _gav = iGAV;
00251     }
00252
00254     void setAvgCancellationPercentage (const
00255 OverbookingRate_T& iACP) {
00256         _acp = iACP;
00257     }
00258
00259     void setETB (const NbOfSeats_T& iETB) {
00260         _etb = iETB;
00261     }
00262
00264     void setStaffNbOfSeats (const NbOfSeats_T& iStaffSeats) {
00265         _staffNbOfBookings = iStaffSeats;
00266     }
00267
00269     void setWLNbOfSeats (const NbOfSeats_T& iWLSeats) {
00270         _wlNbOfBookings = iWLSeats;
00271     }
00272
00274     void setGroupNbOfSeats (const NbOfSeats_T& iGroupSeats) {
00275         _groupNbOfBookings = iGroupSeats;
00276     }
00277
00279     void updateCurrentBidPrice();
00280
00281
00282 public:
00283     // ////////////////////////////////// Display support methods //////////////////////////////////
00288     void toStream (std::ostream& ioOut) const {

```



```

00289         ioOut << toString();
00290     }
00291
00296     void fromStream (std::istream& ioIn) {
00297     }
00298
00302     std::string toString() const;
00303
00307     const std::string describeKey() const {
00308         return _key.toString();
00309     }
00310
00314     const std::string displayVirtualClassList() const;
00315
00316
00317 public:
00318     // ////////// Business methods //////////
00322     void updateFromReservation (const NbOfBookings_T&);
00323
00327     void addVirtualClass (const VirtualClassStruct& iVC) {
00328         _virtualClassList.push_back (iVC);
00329     }
00330
00334     void emptyVirtualClassList() {
00335         _virtualClassList.clear();
00336     }
00337
00341     void emptyBidPriceVector() {
00342         _bidPriceVector.clear();
00343     }
00344
00348     void addDemandInformation (const YieldValue_T&, const
MeanValue_T&,
                                const StdDevValue_T&);
00349
00350
00354     void emptyYieldLevelDemandMap() {
00355         _yieldLevelDemandMap.clear();
00356     }
00357
00358
00359 protected:
00360     // ////////// Constructors and destructors //////////
00364     LegCabin (const Key_T&);
00368     ~LegCabin();
00369
00370
00371 private:
00375     LegCabin();
00379     LegCabin (const LegCabin&);
00380
00381
00382 protected:
00383     // ////////// Attributes //////////
00387     Key_T _key;
00388
00392     BomAbstract* _parent;
00393
00397     HolderMap_T _holderMap;
00398
00400     CabinCapacity_T _offeredCapacity;
00401
00403     CabinCapacity_T _physicalCapacity;
00404
00406     NbOfSeats_T _soldSeat;
00407
00408     /* Committed space. */
00409     CommittedSpace_T _committedSpace;
00410
00412     Availability_T _availabilityPool;
00413
00415     Availability_T _availability;
00416
00418     BidPrice_T _currentBidPrice;
00419
00421     BidPrice_T _previousBidPrice;
00422
00424     BidPriceVector_T _bidPriceVector;
00425
00427     VirtualClassList_T _virtualClassList;
00428
00430     YieldLevelDemandMap_T _yieldLevelDemandMap;
00431
00432
00433 public:
00435     CapacityAdjustment_T _dcsRegrade;
00436
00438     AuthorizationLevel_T _au;

```

```

00439
00441     UPR_T _upr;
00442
00444     Availability_T _nav;
00445
00447     Availability_T _gav;
00448
00450     OverbookingRate_T _acp;
00451
00453     NbOfSeats_T _etb;
00454
00456     NbOfSeats_T _staffNbOfBookings;
00457
00459     NbOfSeats_T _wlNbOfBookings;
00460
00462     NbOfSeats_T _groupNbOfBookings;
00463 };
00464
00465 }
00466 #endif // __STDAIR_BOM_LEG_CABIN_HPP
00467

```

### 33.359 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

- [stdair](#)

*Handle on the StdAir library context.*

### 33.360 LegCabinKey.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/access.hpp>
00011 // StdAir
00012 #include <stdair/basic/BasConst_Inventory.hpp>
00013 #include <stdair/bom/LegCabinKey.hpp>
00014
00015 namespace stdair {
00016
00017 // //////////////////////////////////////
00018 LegCabinKey::LegCabinKey() : _cabinCode (DEFAULT_CABIN_CODE) {
00019     assert (false);
00020 }
00021
00022 // //////////////////////////////////////
00023 LegCabinKey::LegCabinKey (const CabinCode_T& iCabinCode)
00024 : _cabinCode (iCabinCode) {
00025 }
00026
00027 // //////////////////////////////////////
00028 LegCabinKey::LegCabinKey (const LegCabinKey& iKey)
00029 : _cabinCode (iKey._cabinCode) {
00030 }
00031
00032 // //////////////////////////////////////
00033 LegCabinKey::~LegCabinKey () {

```

```

00034     }
00035
00036     // //////////////////////////////////////
00037 void LegCabinKey::toStream (std::ostream& ioOut) const {
00038     ioOut << "LegCabinKey: " << toString() << std::endl;
00039 }
00040
00041     // //////////////////////////////////////
00042 void LegCabinKey::fromStream (std::istream& ioIn) {
00043 }
00044
00045     // //////////////////////////////////////
00046 const std::string LegCabinKey::toString() const {
00047     std::ostringstream oStr;
00048     oStr << _cabinCode;
00049     return oStr.str();
00050 }
00051
00052     // //////////////////////////////////////
00053 void LegCabinKey::serialisationImplementationExport() const {
00054     std::ostringstream oStr;
00055     boost::archive::text_oarchive oa (oStr);
00056     oa << *this;
00057 }
00058
00059     // //////////////////////////////////////
00060 void LegCabinKey::serialisationImplementationImport() {
00061     std::istringstream iStr;
00062     boost::archive::text_iarchive ia (iStr);
00063     ia >> *this;
00064 }
00065
00066     // //////////////////////////////////////
00067 template<class Archive>
00068 void LegCabinKey::serialize (Archive& ioArchive,
00069                             const unsigned int iFileVersion) {
00074     ioArchive & _cabinCode;
00075 }
00076
00077 }

```

### 33.361 stdair/bom/LegCabinKey.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>

```

#### Classes

- struct [stdair::LegCabinKey](#)  
*Key of a given leg-cabin, made of a cabin code (only).*

#### Namespaces

- [boost](#)  
*Forward declarations.*
- [boost::serialization](#)
- [stdair](#)  
*Handle on the StdAir library context.*

### 33.362 LegCabinKey.hpp

```

00001 #ifndef __STDAIR_BOM_LEGCABINKEY_HPP
00002 #define __STDAIR_BOM_LEGCABINKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section

```

```

00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_basic_types.hpp>
00012 #include <stdair/bom/KeyAbstract.hpp>
00013
00015 namespace boost {
00016     namespace serialization {
00017         class access;
00018     }
00019 }
00020
00021 namespace stdair {
00022
00026     struct LegCabinKey : public KeyAbstract {
00027         friend class boost::serialization::access;
00028
00029         // ////////////////////////////////// Constructors and destructors //////////////////////////////////
00030     private:
00031         LegCabinKey();
00032
00033     public:
00034         LegCabinKey (const CabinCode_T& iCabinCode);
00035
00036         LegCabinKey (const LegCabinKey&);
00037
00038         ~LegCabinKey();
00039
00040     public:
00041         // ////////////////////////////////// Getters //////////////////////////////////
00042         const CabinCode_T& getCabinCode() const {
00043             return _cabinCode;
00044         }
00045
00046     public:
00047         // ////////////////////////////////// Display support methods //////////////////////////////////
00048         void toStream (std::ostream& ioOut) const;
00049
00050         void fromStream (std::istream& ioIn);
00051
00052         const std::string toString() const;
00053
00054     public:
00055         // ////////////////////////////////// (Boost) Serialisation support methods //////////////////////////////////
00056         template<class Archive>
00057         void serialize (Archive& ar, const unsigned int iFileVersion);
00058
00059     private:
00060         void serialisationImplementationExport() const;
00061         void serialisationImplementationImport();
00062
00063     private:
00064         // ////////////////////////////////// Attributes //////////////////////////////////
00065         CabinCode_T _cabinCode;
00066     };
00067 }
00068
00069 #endif // __STDAIR_BOM_LEGCABINKEY_HPP

```

### 33.363 stdair/bom/LegCabinTypes.hpp File Reference

```

#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>

```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## Typedefs

- typedef std::list< LegCabin \* > [stdair::LegCabinList\\_T](#)
- typedef std::map< const MapKey\_T, LegCabin \* > [stdair::LegCabinMap\\_T](#)

### 33.364 LegCabinTypes.hpp

```

00001 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00002 #ifndef __STDAIR_BOM_LEGCABINTYPES_HPP
00003 #define __STDAIR_BOM_LEGCABINTYPES_HPP
00004
00005 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00006 // Import section
00007 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // StdAir
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016     // Forward declarations.
00017     class LegCabin;
00018
00020     typedef std::list<LegCabin*> LegCabinList_T;
00021
00023     typedef std::map<const MapKey_T, LegCabin*> LegCabinMap_T;
00024
00025 }
00026 #endif // __STDAIR_BOM_LEGCABINTYPES_HPP
00027

```

### 33.365 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

- [stdair](#)  
*Handle on the StdAir library context.*

### 33.366 LegDate.cpp

```

00001 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00002 // Import section
00003 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_General.hpp>
00009 #include <stdair/basic/BasConst_Inventory.hpp>
00010 #include <stdair/basic/BasConst_BomDisplay.hpp>
00011 #include <stdair/bom/BomManager.hpp>
00012 #include <stdair/bom/FlightDate.hpp>
00013 #include <stdair/bom/LegCabin.hpp>
00014 #include <stdair/bom/LegDate.hpp>
00015

```

```

00016 namespace stdair {
00017
00018 // //////////////////////////////////////
00019 LegDate::LegDate() : _key (DEFAULT_ORIGIN), _parent (NULL) {
00020     assert (false);
00021 }
00022
00023 // //////////////////////////////////////
00024 LegDate::LegDate (const LegDate& iLegDate) :
00025     _key (iLegDate._key),
00026     _parent (NULL),
00027     _offPoint (iLegDate._offPoint),
00028     _boardingDate (iLegDate._boardingDate),
00029     _boardingTime (iLegDate._boardingTime),
00030     _offDate (iLegDate._offDate),
00031     _offTime (iLegDate._offTime),
00032     _elapsedTime (iLegDate._elapsedTime),
00033     _distance (iLegDate._distance),
00034     _capacity (iLegDate._capacity) {
00035 }
00036
00037 // //////////////////////////////////////
00038 LegDate::LegDate (const Key_T& iKey)
00039 : _key (iKey), _parent (NULL), _distance (DEFAULT_DISTANCE_VALUE),
00040   _capacity (DEFAULT_CABIN_CAPACITY) {
00041 }
00042
00043 // //////////////////////////////////////
00044 LegDate::~LegDate () {
00045 }
00046
00047 // //////////////////////////////////////
00048 const AirlineCode_T& LegDate::getAirlineCode() const {
00049     const FlightDate* lFlightDate_ptr =
00050         static_cast<const FlightDate*> (getParent());
00051     assert (lFlightDate_ptr != NULL);
00052     return lFlightDate_ptr->getAirlineCode();
00053 }
00054
00055 // //////////////////////////////////////
00056 std::string LegDate::toString() const {
00057     std::ostringstream oStr;
00058     oStr << describeKey();
00059     return oStr.str();
00060 }
00061
00062 // //////////////////////////////////////
00063 const std::string LegDate::describeRoutingKey() const {
00064     const FlightDate* lFlightDate_ptr =
00065         static_cast<const FlightDate*> (getParent());
00066     assert (lFlightDate_ptr != NULL);
00067     std::ostringstream oStr;
00068     oStr << _operatingAirlineCode <<
00069         DEFAULT_KEY_FLD_DELIMITER
00070         << _operatingFlightNumber <<
00071         DEFAULT_KEY_FLD_DELIMITER
00072         << lFlightDate_ptr->getDepartureDate() <<
00073         DEFAULT_KEY_FLD_DELIMITER
00074         << describeKey();
00075     return oStr.str();
00076 }
00077
00078 // //////////////////////////////////////
00079 LegCabin* LegDate::getLegCabin (const std::string& iLegCabinKeyStr) const {
00080     LegCabin* oLegCabin_ptr =
00081         BomManager::getObjectPtr<LegCabin> (*this, iLegCabinKeyStr);
00082     return oLegCabin_ptr;
00083 }
00084
00085 // //////////////////////////////////////
00086 LegCabin* LegDate::getLegCabin (const LegCabinKey& iLegCabinKey)
00087 const {
00088     return getLegCabin (iLegCabinKey.toString());
00089 }
00090
00091 // //////////////////////////////////////
00092 const Duration_T LegDate::getTimeOffset() const {
00093     // TimeOffset = (OffTime - BoardingTime) + (OffDate - BoardingDate) * 24
00094     // - ElapsedTime
00095     Duration_T oTimeOffset = (_offTime - _boardingTime);
00096
00097     const DateOffset_T lDateOffset = getDateOffset();
00098
00099     const Duration_T lDateOffsetInHours (lDateOffset.days() * 24, 0, 0);
00100
00101     oTimeOffset += lDateOffsetInHours - _elapsedTime;
00102 }

```

```

00099     return oTimeOffset;
00100 }
00101
00102 // //////////////////////////////////////
00103 void LegDate::setElapsedTime (const Duration_T& iElapsedTime) {
00104     // Set Elapsed time
00105     _elapsedTime = iElapsedTime;
00106
00107     // Update distance according to the mean plane speed
00108     updateDistanceFromElapsedTime();
00109 }
00110
00111 // //////////////////////////////////////
00112 void LegDate::updateDistanceFromElapsedTime() {
00113     //
00114     const double lElapseInHours =
00115         static_cast<const double> (_elapsedTime.hours());
00116
00117     // Normally, Distance_T is an unsigned long int
00118     const Distance_T lDistance =
00119         static_cast<const Distance_T> (DEFAULT_FLIGHT_SPEED * lElapseInHours);
00120
00121     _distance = lDistance;
00122 }
00123
00124 }
00125

```

### 33.367 stdair/bom/LegDate.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/LegDateKey.hpp>
#include <stdair/bom/LegDateTypes.hpp>

```

#### Classes

- class [stdair::LegDate](#)

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.368 LegDate.hpp

```

00001 #ifndef __STDAIR_BOM_LEGDATE_HPP
00002 #define __STDAIR_BOM_LEGDATE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/bom/BomAbstract.hpp>
00013 #include <stdair/bom/LegDateKey.hpp>
00014 #include <stdair/bom/LegDateTypes.hpp>
00015
00016 namespace stdair {
00017
00018     struct LegCabinKey;
00019     class LegCabin;
00020
00021     class LegDate : public BomAbstract {
00022     public:
00023         template <typename BOM> friend class FacBom;
00024     };
00025
00026 }

```

```

00027     template <typename BOM> friend class FacCloneBom;
00028     friend class FacBomManager;
00029
00030 public:
00031     // ////////// Type definitions //////////
00032     typedef LegDateKey Key_T;
00033
00034
00035
00036 public:
00037     // ////////// Getters //////////
00038     const Key_T& getKey() const {
00039         return _key;
00040     }
00041
00042     BomAbstract* const getParent() const {
00043         return _parent;
00044     }
00045
00046     const AirportCode_T& getBoardingPoint() const {
00047         return _key.getBoardingPoint();
00048     }
00049
00050     const AirlineCode_T& getAirlineCode() const;
00051
00052     const HolderMap_T& getHolderMap() const {
00053         return _holderMap;
00054     }
00055
00056     LegCabin* getLegCabin (const std::string& iLegCabinKeyStr) const;
00057
00058     LegCabin* getLegCabin (const LegCabinKey&) const;
00059
00060     const AirportCode_T& getOffPoint() const {
00061         return _offPoint;
00062     }
00063
00064     const Date_T& getBoardingDate() const {
00065         return _boardingDate;
00066     }
00067
00068     const Duration_T& getBoardingTime() const {
00069         return _boardingTime;
00070     }
00071
00072     const Date_T& getOffDate() const {
00073         return _offDate;
00074     }
00075
00076     const Duration_T& getOffTime() const {
00077         return _offTime;
00078     }
00079
00080     const Duration_T& getElapsedTime() const {
00081         return _elapsedTime;
00082     }
00083
00084     const Distance_T& getDistance() const {
00085         return _distance;
00086     }
00087
00088     const CabinCapacity_T& getCapacity() const {
00089         return _capacity;
00090     }
00091
00092     const DateOffset_T getDateOffset() const {
00093         return _offDate - _boardingDate;
00094     }
00095
00096     const Duration_T getTimeOffset() const;
00097
00098 public:
00099     // ////////// Setters //////////
00100     void setOffPoint (const AirportCode_T& iOffPoint) {
00101         _offPoint = iOffPoint;
00102     }
00103
00104     void setBoardingDate (const Date_T& iBoardingDate) {
00105         _boardingDate = iBoardingDate;
00106     }
00107
00108     void setBoardingTime (const Duration_T& iBoardingTime) {
00109         _boardingTime = iBoardingTime;
00110     }
00111
00112     void setOffDate (const Date_T& iOffDate) {
00113         _offDate = iOffDate;
00114     }

```



```

00165     }
00166
00168     void setOffTime (const Duration_T& iOffTime) {
00169         _offTime = iOffTime;
00170     }
00171
00173     void setElapsedTime (const Duration_T&);
00174
00176     void setOperatingAirlineCode (const AirlineCode_T& iAirlineCode) {
00177         _operatingAirlineCode = iAirlineCode;
00178     }
00179
00181     void setOperatingFlightNumber (const FlightNumber_T&
iFlightNumber) {
00182         _operatingFlightNumber = iFlightNumber;
00183     }
00184
00185 private:
00187     void updateDistanceFromElapsedTime();
00188
00189
00190 public:
00191     // /////////// Display support methods ///////////
00194     void toStream (std::ostream& ioOut) const {
00195         ioOut << toString();
00196     }
00197
00200     void fromStream (std::istream& ioIn) {
00201     }
00202
00204     std::string toString() const;
00205
00207     const std::string describeKey() const {
00208         return _key.toString();
00209     }
00210
00212     const std::string describeRoutingKey() const;
00213
00214 protected:
00215     // /////////// Constructors and destructors ///////////
00217     LegDate (const Key_T&);
00219     virtual ~LegDate();
00220
00221 private:
00223     LegDate();
00225     LegDate (const LegDate&);
00226
00227
00228 protected:
00229     // /////////// Attributes ///////////
00231     Key_T _key;
00232
00234     BomAbstract* _parent;
00235
00237     HolderMap_T _holderMap;
00238
00240     AirportCode_T _offPoint;
00241
00243     Date_T _boardingDate;
00244
00246     Duration_T _boardingTime;
00247
00249     Date_T _offDate;
00250
00252     Duration_T _offTime;
00253
00255     Duration_T _elapsedTime;
00256
00258     Distance_T _distance;
00259
00261     CabinCapacity_T _capacity;
00262
00264     AirlineCode_T _operatingAirlineCode;
00265
00267     FlightNumber_T _operatingFlightNumber;
00268 };
00269
00270 }
00271 #endif // __STDAIR_BOM_LEGDATE_HPP
00272

```

### 33.369 stdair/bom/LegDateKey.cpp File Reference

```
#include <cassert>
```

```
#include <sstream>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/LegDateKey.hpp>
```

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.370 LegDateKey.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_Inventory.hpp>
00009 #include <stdair/bom/LegDateKey.hpp>
00010
00011 namespace stdair {
00012
00013 // //////////////////////////////////////
00014 LegDateKey::LegDateKey() : _boardingPoint (DEFAULT_ORIGIN) {
00015     assert (false);
00016 }
00017
00018 // //////////////////////////////////////
00019 LegDateKey::LegDateKey (const AirportCode_T& iBoardingPoint)
00020     : _boardingPoint (iBoardingPoint) {
00021 }
00022
00023 // //////////////////////////////////////
00024 LegDateKey::LegDateKey (const LegDateKey& iKey)
00025     : _boardingPoint (iKey._boardingPoint) {
00026 }
00027
00028 // //////////////////////////////////////
00029 LegDateKey::~LegDateKey () {
00030 }
00031
00032 // //////////////////////////////////////
00033 void LegDateKey::toStream (std::ostream& ioOut) const {
00034     ioOut << "LegDateKey: " << toString();
00035 }
00036
00037 // //////////////////////////////////////
00038 void LegDateKey::fromStream (std::istream& ioIn) {
00039 }
00040
00041 // //////////////////////////////////////
00042 const std::string LegDateKey::toString() const {
00043     std::ostringstream oStr;
00044     oStr << _boardingPoint;
00045     return oStr.str();
00046 }
00047
00048 }
```

## 33.371 stdair/bom/LegDateKey.hpp File Reference

```
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

## Classes

- struct [stdair::LegDateKey](#)

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.372 LegDateKey.hpp

```

00001 #ifndef __STDAIR_BOM_LEGDATEKEY_HPP
00002 #define __STDAIR_BOM_LEGDATEKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009 #include <stdair/bom/KeyAbstract.hpp>
00010
00011 namespace stdair {
00012
00016     struct LegDateKey : public KeyAbstract {
00017
00018         // ////////////////////////////////// Constructors and destructors //////////////////////////////////
00019     private:
00021         LegDateKey();
00022
00023     public:
00025         LegDateKey (const AirportCode_T& iBoardingPoint);
00027         LegDateKey (const LegDateKey&);
00029         ~LegDateKey ();
00030
00031
00032         // ////////////////////////////////// Getters //////////////////////////////////
00034         const AirportCode_T& getBoardingPoint() const {
00035             return _boardingPoint;
00036         }
00037
00038
00039         // ////////////////////////////////// Display support methods //////////////////////////////////
00042         void toStream (std::ostream& ioOut) const;
00043
00046         void fromStream (std::istream& ioIn);
00047
00053         const std::string toString() const;
00054
00055     private:
00057         // ////////////////////////////////// Attributes //////////////////////////////////
00059         AirportCode_T _boardingPoint;
00060     };
00061
00062 }
00063 #endif // __STDAIR_BOM_LEGDATEKEY_HPP

```

## 33.373 stdair/bom/LegDateTypes.hpp File Reference

```

#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>

```

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## Typedefs

- typedef std::list< LegDate \* > [stdair::LegDateList\\_T](#)
- typedef std::map< const MapKey\_T, LegDate \* > [stdair::LegDateMap\\_T](#)

## 33.374 LegDateTypes.hpp

```

00001 ///////////////////////////////////////////////////////////////////
00002 #ifndef __STDAIR_BOM_LEGDATEYPES_HPP
00003 #define __STDAIR_BOM_LEGDATEYPES_HPP
00004
00005 ///////////////////////////////////////////////////////////////////
00006 // Import section
00007 ///////////////////////////////////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // StdAir
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016     // Forward declarations.
00017     class LegDate;
00018
00019     typedef std::list<LegDate*> LegDateList_T;
00020
00021     typedef std::map<const MapKey_T, LegDate*> LegDateMap_T;
00022 }
00023 #endif // __STDAIR_BOM_LEGDATEYPES_HPP
00024
00025
00026

```

## 33.375 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

- [stdair](#)

*Handle on the StdAir library context.*

## 33.376 NestingNode.cpp

```

00001 ///////////////////////////////////////////////////////////////////
00002 // Import section
00003 ///////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <sstream>
00006 #include <cassert>
00007 #include <iomanip>
00008 #include <iostream>
00009 //STDAIR
00010 #include <stdair/basic/BasConst_Inventory.hpp>
00011 #include <stdair/bom/BomManager.hpp>
00012 #include <stdair/bom/BookingClass.hpp>
00013 #include <stdair/bom/BookingClassTypes.hpp>
00014 #include <stdair/bom/NestingNode.hpp>
00015
00016 namespace stdair {
00017
00018     ///////////////////////////////////////////////////////////////////
00019     NestingNode::NestingNode () :
00020         _key (DEFAULT_NESTING_NODE_CODE), _parent (NULL) {
00021         assert (false);
00022     }
00023
00024     ///////////////////////////////////////////////////////////////////
00025     NestingNode::NestingNode (const NestingNode& iNestingNode)

```

```

00026 : _key (DEFAULT_NESTING_NODE_CODE), _parent (NULL) {
00027     assert (false);
00028 }
00029
00030 // //////////////////////////////////////
00031 NestingNode::NestingNode (const Key_T& iKey) : _key (iKey), _parent (NULL) {
00032 }
00033
00034 // //////////////////////////////////////
00035 NestingNode::~NestingNode() {
00036 }
00037
00038 // //////////////////////////////////////
00039 std::string NestingNode::toString () const {
00040     std::ostringstream oStr;
00041     oStr << describeKey();
00042
00043     oStr << _yield << std::endl;
00044
00045     return oStr.str();
00046 }
00047
00048 }

```

### 33.377 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>

```

#### Classes

- class [stdair::NestingNode](#)

#### Namespaces

- [boost](#)

*Forward declarations.*

- [boost::serialization](#)
- [stdair](#)

*Handle on the StdAir library context.*

### 33.378 NestingNode.hpp

```

00001 #ifndef __STDAIR_BOM_NESTINGNODE_HPP
00002 #define __STDAIR_BOM_NESTINGNODE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <cmath>
00009 // StdAir
00010 #include <stdair/stdair_basic_types.hpp>
00011 #include <stdair/stdair_rm_types.hpp>
00012 #include <stdair/bom/BomAbstract.hpp>
00013 #include <stdair/bom/BookingClassTypes.hpp>
00014 #include <stdair/bom/NestingNodeKey.hpp>
00015
00016 namespace boost {
00017     namespace serialization {
00018         class access;
00019     }
00020 }
00021
00022

```

```

00023 namespace stdair {
00024
00029     class NestingNode : public BomAbstract {
00030     template <typename BOM> friend class FacBom;
00031     friend class FacBomManager;
00032     friend class boost::serialization::access;
00033
00034     public:
00035         // //////////// Type definitions ////////////
00039         typedef NestingNodeKey Key_T;
00040
00041     public:
00042         // //////////// Getters ////////////
00044         const Key_T& getKey() const {
00045             return _key;
00046         }
00047
00049         BomAbstract* const getParent() const {
00050             return _parent;
00051         }
00052
00056         const HolderMap_T& getHolderMap() const {
00057             return _holderMap;
00058         }
00059
00061         const Yield_T& getYield() const {
00062             return _yield;
00063         }
00064
00065     public:
00066         // //////////// Setters ////////////
00068         void setYield (const Yield_T& iYield) {
00069             _yield = iYield;
00070         }
00071
00072
00073     public:
00074         // //////////// Display support methods ////////////
00080         void toStream (std::ostream& ioOut) const {
00081             ioOut << toString();
00082         }
00083
00089         void fromStream (std::istream& ioIn) {
00090         }
00091
00095         std::string toString() const;
00096
00100         const std::string describeKey() const {
00101             return _key.toString();
00102         }
00103
00104
00105     public:
00106         // //////////// (Boost) Serialisation support methods ////////////
00110         template<class Archive>
00111         void serialize (Archive& ar, const unsigned int iFileVersion);
00112
00113     private:
00121         void serialisationImplementationExport() const;
00122         void serialisationImplementationImport();
00123
00124
00125     protected:
00126         // //////////// Constructors and destructor. ////////////
00130         NestingNode (const Key_T&);
00131
00135         virtual ~NestingNode();
00136
00137     private:
00141         NestingNode();
00142
00146         NestingNode (const NestingNode&);
00147
00148
00149     private:
00150         // //////////// Attributes ////////////
00154         Key_T _key;
00155
00159         BomAbstract* _parent;
00160
00164         HolderMap_T _holderMap;
00165
00169         Yield_T _yield;
00170
00171     };
00172 }
00173 #endif // __STDAIR_BOM_NESTINGNODE_HPP

```

### 33.379 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

- [stdair](#)  
*Handle on the StdAir library context.*

#### 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)

### 33.380 NestingNodeKey.cpp

```
00001 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00002 // Import section
00003 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/access.hpp>
00011 // StdAir
00012 #include <stdair/basic/BasConst_Inventory.hpp>
00013 #include <stdair/bom/NestingNodeKey.hpp>
00014
00015 namespace stdair {
00016
00017 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00018 NestingNodeKey::NestingNodeKey() : _nestingNodeCode (DEFAULT_POLICY_CODE) {
00019     assert (false);
00020 }
00021
00022 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00023 NestingNodeKey::NestingNodeKey (const NestingNodeKey& iNestingNodeKey)
00024     : _nestingNodeCode (iNestingNodeKey._nestingNodeCode) {
00025 }
00026
00027 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00028 NestingNodeKey::NestingNodeKey (const NestingNodeCode_T& iNestingNodeCode)
00029     : _nestingNodeCode (iNestingNodeCode) {
00030 }
00031
00032 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00033 NestingNodeKey::~NestingNodeKey() {
00034 }
00035
00036 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00037 void NestingNodeKey::toStream (std::ostream& ioOut) const {
00038     ioOut << "NestingNodeKey: " << toString();
00039 }
00040
00041 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00042 void NestingNodeKey::fromStream (std::istream& ioIn) {
00043 }
00044
00045 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00046 const std::string NestingNodeKey::toString() const {
00047     std::ostringstream oStr;
00048     oStr << _nestingNodeCode;
00049     return oStr.str();
00050 }
```

```

00050     }
00051
00052     // //////////////////////////////////////
00053     void NestingNodeKey::serialisationImplementationExport() const {
00054         std::ostringstream ostr;
00055         boost::archive::text_oarchive oa (ostr);
00056         oa << *this;
00057     }
00058
00059     // //////////////////////////////////////
00060     void NestingNodeKey::serialisationImplementationImport() {
00061         std::istringstream istr;
00062         boost::archive::text_iarchive ia (istr);
00063         ia >> *this;
00064     }
00065
00066     // //////////////////////////////////////
00067     template<class Archive>
00068     void NestingNodeKey::serialize (Archive& ioArchive,
00069                                     const unsigned int iFileVersion) {
00070         ioArchive & _nestingNodeCode;
00071     }
00072
00073     // //////////////////////////////////////
00074     // Explicit template instantiation
00075     namespace ba = boost::archive;
00076     template void NestingNodeKey::serialize<ba::text_oarchive> (ba::text_oarchive&,
00077                                                                unsigned int);
00078     template void NestingNodeKey::serialize<ba::text_iarchive> (ba::text_iarchive&,
00079                                                                unsigned int);
00080     // //////////////////////////////////////
00081 }

```

### 33.381 stdair/bom/NestingNodeKey.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>

```

#### Classes

- struct [stdair::NestingNodeKey](#)  
*Key of a given policy, made of a policy code.*

#### Namespaces

- [boost](#)  
*Forward declarations.*
- [boost::serialization](#)
- [stdair](#)  
*Handle on the StdAir library context.*

### 33.382 NestingNodeKey.hpp

```

00001 #ifndef __STDAIR_BOM_NESTINGNODEKEY_HPP
00002 #define __STDAIR_BOM_NESTINGNODEKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/bom/KeyAbstract.hpp>

```



```

00013
00015 namespace boost {
00016     namespace serialization {
00017         class access;
00018     }
00019 }
00020
00021 namespace stdair {
00022
00026     struct NestingNodeKey : public KeyAbstract {
00027         friend class boost::serialization::access;
00028
00029         // ////////// Constructors and destructors //////////
00030     private:
00031         NestingNodeKey();
00032
00033     public:
00034         NestingNodeKey (const NestingNodeCode_T& iNestingNodeCode);
00035
00036         NestingNodeKey (const NestingNodeKey&);
00037
00038         ~NestingNodeKey();
00039
00040     public:
00041         // ////////// Getters //////////
00042         const NestingNodeCode_T& getNestingNodeCode () const {
00043             return _nestingNodeCode;
00044         }
00045
00046     public:
00047         // ////////// Display support methods //////////
00048         void toStream (std::ostream& ioOut) const;
00049
00050         void fromStream (std::istream& ioIn);
00051
00052         const std::string toString() const;
00053
00054     public:
00055         // ////////// (Boost) Serialisation support methods //////////
00056         template<class Archive>
00057         void serialize (Archive& ar, const unsigned int iFileVersion);
00058
00059     private:
00060         void serialisationImplementationExport() const;
00061         void serialisationImplementationImport();
00062
00063     private:
00064         // ////////// Attributes //////////
00065         NestingNodeCode_T _nestingNodeCode;
00066     };
00067 }
00068 #endif // __STDAIR_BOM_NESTINGNODEKEY_HPP

```

### 33.383 stdair/bom/NestingNodeTypes.hpp File Reference

```

#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>

```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

#### Typedefs

- typedef std::list< NestingNode \* > [stdair::NestingNodeList\\_T](#)
- typedef std::map< const MapKey\_T, NestingNode \* > [stdair::NestingNodeMap\\_T](#)

**33.384 NestingNodeTypes.hpp**

```

00001 ///////////////////////////////////////////////////////////////////
00002 #ifndef __STDAIR_BOM_NESTINGNODETYPES_HPP
00003 #define __STDAIR_BOM_NESTINGNODETYPES_HPP
00004
00005 ///////////////////////////////////////////////////////////////////
00006 // Import section
00007 ///////////////////////////////////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // StdAir
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016     // Forward declarations.
00017     class NestingNode;
00018
00019     typedef std::list<NestingNode*> NestingNodeList_T;
00020
00021     typedef std::map<const MapKey_T, NestingNode*> NestingNodeMap_T;
00022
00023 }
00024
00025 #endif // __STDAIR_BOM_NESTINGNODETYPES_HPP

```

**33.385 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**

- [stdair](#)

*Handle on the StdAir library context.*

**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)

**33.386 NestingStructureKey.cpp**

```

00001 ///////////////////////////////////////////////////////////////////
00002 // Import section
00003 ///////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/access.hpp>
00011 // StdAir
00012 #include <stdair/basic/BasConst_Inventory.hpp>
00013 #include <stdair/bom/NestingStructureKey.hpp>
00014
00015 namespace stdair {
00016
00017     ///////////////////////////////////////////////////////////////////
00018     NestingStructureKey::NestingStructureKey() : _nestingStructureCode (
        DEFAULT_NESTING_STRUCTURE_CODE) {

```

```

00019     assert (false);
00020 }
00021
00022 // //////////////////////////////////////
00023 NestingStructureKey::NestingStructureKey (const NestingStructureKey&
iNestingStructureKey)
00024 : _nestingStructureCode (iNestingStructureKey._nestingStructureCode) {
00025 }
00026
00027 // //////////////////////////////////////
00028 NestingStructureKey::NestingStructureKey (const NestingStructureCode_T&
iNestingStructureCode)
00029 : _nestingStructureCode (iNestingStructureCode) {
00030 }
00031
00032 // //////////////////////////////////////
00033 NestingStructureKey::~NestingStructureKey() {
00034 }
00035
00036 // //////////////////////////////////////
00037 void NestingStructureKey::toStream (std::ostream& ioOut) const {
00038     ioOut << "NestingStructureKey: " << toString();
00039 }
00040
00041 // //////////////////////////////////////
00042 void NestingStructureKey::fromStream (std::istream& ioIn) {
00043 }
00044
00045 // //////////////////////////////////////
00046 const std::string NestingStructureKey::toString() const {
00047     std::ostringstream oStr;
00048     oStr << _nestingStructureCode;
00049     return oStr.str();
00050 }
00051
00052 // //////////////////////////////////////
00053 void NestingStructureKey::serialisationImplementationExport() const {
00054     std::ostringstream oStr;
00055     boost::archive::text_oarchive oa (oStr);
00056     oa << *this;
00057 }
00058
00059 // //////////////////////////////////////
00060 void NestingStructureKey::serialisationImplementationImport() {
00061     std::istringstream iStr;
00062     boost::archive::text_iarchive ia (iStr);
00063     ia >> *this;
00064 }
00065
00066 // //////////////////////////////////////
00067 template<class Archive>
00068 void NestingStructureKey::serialize (Archive& ioArchive,
const unsigned int iFileVersion) {
00069     ioArchive & _nestingStructureCode;
00070 }
00071
00072 // //////////////////////////////////////
00073 // Explicit template instantiation
00074 namespace ba = boost::archive;
00075 template void NestingStructureKey::serialize<ba::text_oarchive> (ba::text_oarchive&,
unsigned int);
00076 template void NestingStructureKey::serialize<ba::text_iarchive> (ba::text_iarchive&,
unsigned int);
00077 // //////////////////////////////////////
00078
00085 }
00086 }

```

### 33.387 stdair/bom/NestingStructureKey.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>

```

#### Classes

- struct [stdair::NestingStructureKey](#)

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

## Namespaces

- [boost](#)

*Forward declarations.*

- [boost::serialization](#)
- [stdair](#)

*Handle on the StdAir library context.*

## 33.388 NestingStructureKey.hpp

```

00001 #ifndef __STDAIR_BOM_NESTINGSTRUCTUREKEY_HPP
00002 #define __STDAIR_BOM_NESTINGSTRUCTUREKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/bom/KeyAbstract.hpp>
00013
00015 namespace boost {
00016     namespace serialization {
00017         class access;
00018     }
00019 }
00020
00021 namespace stdair {
00022
00026     struct NestingStructureKey : public KeyAbstract {
00027         friend class boost::serialization::access;
00028
00029         // ////////// Constructors and destructors //////////
00030     private:
00034         NestingStructureKey();
00035
00036     public:
00040         NestingStructureKey (const NestingStructureCode_T&
iNestingStructureCode);
00041
00045         NestingStructureKey (const NestingStructureKey&);
00046
00050         ~NestingStructureKey();
00051
00052     public:
00054         // ////////// Getters //////////
00056         const NestingStructureCode_T& getNestingStructureCode ()
const {
00057             return _nestingStructureCode;
00058         }
00059
00060     public:
00062         // ////////// Display support methods //////////
00068         void toStream (std::ostream& ioOut) const;
00069
00075         void fromStream (std::istream& ioIn);
00076
00086         const std::string toString() const;
00087
00088     public:
00089         // ////////// (Boost) Serialisation support methods //////////
00094         template<class Archive>
00095         void serialize (Archive& ar, const unsigned int iFileVersion);
00096
00097     private:
00102         void serialisationImplementationExport() const;
00103         void serialisationImplementationImport();
00104
00105     private:
00107         // ////////// Attributes //////////

```

```

00111     NestingStructureCode_T _nestingStructureCode;
00112 };
00113
00114 }
00115 #endif // __STDAIR_BOM_NESTINGSTRUCTUREKEY_HPP

```

### 33.389 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

- [stdair](#)

*Handle on the StdAir library context.*

### 33.390 OnDDate.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_Inventory.hpp>
00009 #include <stdair/basic/BasConst_General.hpp>
00010 #include <stdair/bom/BomManager.hpp>
00011 #include <stdair/bom/Inventory.hpp>
00012 #include <stdair/bom/OnDDate.hpp>
00013
00014 namespace stdair {
00015
00016 // //////////////////////////////////////
00017 OnDDate::OnDDate()
00018 : _key (DEFAULT_OND_STRING_LIST), _parent (NULL) {
00019     assert (false);
00020 }
00021
00022 // //////////////////////////////////////
00023 OnDDate::OnDDate (const OnDDate& iOnDDate)
00024 : _key (iOnDDate.getKey()), _parent (NULL) {
00025 }
00026
00027 // //////////////////////////////////////
00028 OnDDate::OnDDate (const Key_T& iKey)
00029 : _key (iKey), _parent (NULL) {
00030 }
00031
00032 // //////////////////////////////////////
00033 OnDDate::~OnDDate() {
00034 }
00035
00036 // //////////////////////////////////////
00037 std::string OnDDate::toString() const {
00038     std::ostringstream oStr;
00039     oStr << describeKey();
00040     return oStr.str();
00041 }
00042
00043 // //////////////////////////////////////
00044 const AirlineCode_T& OnDDate::getAirlineCode() const {
00045     const Inventory* lInventory_ptr =
00046         static_cast<const Inventory*> (getParent());
00047     assert (lInventory_ptr != NULL);
00048     return lInventory_ptr->getAirlineCode();
00049 }

```

```

00050
00051 ///////////////////////////////////////////////////////////////////
00052 void OnDDate::
00053 setDemandInformation (const CabinClassPairList_T&
iCabinClassPairList,
00054                     const YieldDemandPair_T& iYieldDemandPair) {
00055     std::ostringstream oStr;
00056     for(CabinClassPairList_T::const_iterator itCCP = iCabinClassPairList.begin();
00057         itCCP != iCabinClassPairList.end(); ++itCCP) {
00058         oStr << itCCP->first << ":" << itCCP->second << ";";
00059     }
00060     std::string lCabinClassPath = oStr.str();
00061     StringDemandStructMap_T::iterator it =
00062     _classPathDemandMap.find(lCabinClassPath);
00063     if (it == _classPathDemandMap.end()) {
00064         const StringDemandStructPair_T lPairStringDemandChar (lCabinClassPath,
00065                                                                iYieldDemandPair);
00066         _classPathDemandMap.insert (lPairStringDemandChar);
00067         const StringCabinClassPair_T lStringCabinClassPair (lCabinClassPath,
00068                                                            iCabinClassPairList);
00069         _stringCabinClassPairListMap.insert (lStringCabinClassPair);
00070     } else {
00071         it->second = iYieldDemandPair;
00072     }
00073 }
00074
00075 ///////////////////////////////////////////////////////////////////
00076 void OnDDate::setTotalForecast (const CabinCode_T& iCabinCode,
00077                                const WTPDemandPair_T& iWTPDemandPair) {
00078
00079     CabinForecastMap_T::iterator it =
00080     _cabinForecastMap.find (iCabinCode);
00081     if (it == _cabinForecastMap.end()) {
00082         const CabinForecastPair_T lPairCabinForecastChar (iCabinCode,
00083                                                            iWTPDemandPair);
00084         _cabinForecastMap.insert (lPairCabinForecastChar);
00085     } else {
00086         assert (false);
00087     }
00088 }
00089
00090 }

```

### 33.391 stdair/bom/OnDDate.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/stdair_rm_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/OnDDateKey.hpp>
#include <stdair/bom/OnDDateTypes.hpp>

```

#### Classes

- class [stdair::OnDDate](#)  
Class representing the actual attributes for an airline flight-date.

#### Namespaces

- [boost](#)  
Forward declarations.
- [boost::serialization](#)
- [stdair](#)  
Handle on the StdAir library context.

## 33.392 OnDDate.hpp

```

00001 #ifndef __STDAIR_BOM_ONDDATE_HPP
00002 #define __STDAIR_BOM_ONDDATE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/stdair_maths_types.hpp>
00013 #include <stdair/stdair_basic_types.hpp>
00014 #include <stdair/stdair_demand_types.hpp>
00015 #include <stdair/stdair_rm_types.hpp>
00016 #include <stdair/bom/BomAbstract.hpp>
00017 #include <stdair/bom/OnDDateKey.hpp>
00018 #include <stdair/bom/OnDDateTypes.hpp>
00019
00021 namespace boost {
00022     namespace serialization {
00023         class access;
00024     }
00025 }
00026
00027 namespace stdair {
00028
00033     class OnDDate : public BomAbstract {
00034     public:
00035         template <typename BOM> friend class FacBom;
00036         template <typename BOM> friend class FacCloneBom;
00037         friend class FacBomManager;
00038         friend class boost::serialization::access;
00039
00040         // ////////////////////////////////// Type definitions //////////////////////////////////
00041         typedef OnDDateKey Key_T;
00042
00043     public:
00044         // ////////////////////////////////// Getters //////////////////////////////////
00045         const Key_T& getKey() const {
00046             return _key;
00047         }
00048
00049         BomAbstract* const getParent() const {
00050             return _parent;
00051         }
00052
00053         const AirlineCode_T& getAirlineCode() const;
00054
00055         const stdair::Date_T getDate() const {
00056             return _key.getDate();
00057         }
00058
00059         const stdair::AirportCode_T getOrigin() const {
00060             return _key.getOrigin();
00061         }
00062
00063         const stdair::AirportCode_T getDestination() const {
00064             return _key.getDestination();
00065         }
00066
00067         const HolderMap_T& getHolderMap() const {
00068             return _holderMap;
00069         }
00070
00071         const StringDemandStructMap_T& getDemandInfoMap () const {
00072             return _classPathDemandMap;
00073         }
00074
00075         const CabinForecastMap_T& getTotalForecastMap () const {
00076             return _cabinForecastMap;
00077         }
00078
00079         const WTPDemandPair_T& getTotalForecast (const
00080 CabinCode_T& iCC) const {
00081             assert (_cabinForecastMap.find(iCC) != _cabinForecastMap.end());
00082             return _cabinForecastMap.find(iCC)->second;
00083         }
00084
00085         const CabinClassPairList_T& getCabinClassPairList (const
00086 std::string& iStr) const {
00087             assert (_stringCabinClassPairListMap.find(iStr) !=
00088 _stringCabinClassPairListMap.end());

```

```

00118         return _stringCabinClassPairListMap.find(iStr)->second;
00119     }
00120
00124     const short getNbOfSegments () const {
00125         return _key.getNbOfSegments();
00126     }
00127
00128 public:
00129     // //////////// Setters ////////////
00131     void setDemandInformation (const CabinClassPairList_T&,
00132                             const YieldDemandPair_T&);
00133
00134
00136     void setTotalForecast (const CabinCode_T&,
00137                         const WTPDemandPair_T&);
00138
00139
00140 public:
00141     // //////////// Display support methods ////////////
00147     void toStream (std::ostream& ioOut) const {
00148         ioOut << toString();
00149     }
00150
00156     void fromStream (std::istream& ioIn) {
00157     }
00158
00162     std::string toString() const;
00163
00167     const std::string describeKey() const {
00168         return _key.toString();
00169     }
00170
00171
00172 public:
00173     // //////////// (Boost) Serialisation support methods ////////////
00177     template<class Archive>
00178     void serialize (Archive& ar, const unsigned int iFileVersion);
00179
00180 private:
00185     void serialisationImplementation();
00186
00187
00188 protected:
00189     // //////////// Constructors and destructors ////////////
00193     OnDDate (const Key_T&);
00194
00198     virtual ~OnDDate();
00199
00200 private:
00204     OnDDate();
00205
00209     OnDDate (const OnDDate&);
00210
00211
00212 protected:
00213     // //////////// Attributes ////////////
00217     Key_T _key;
00218
00222     BomAbstract* _parent;
00223
00227     HolderMap_T _holderMap;
00228
00232     StringDemandStructMap_T _classPathDemandMap;
00233
00237     StringCabinClassPairListMap_T
    _stringCabinClassPairListMap;
00238
00242     CabinForecastMap_T _cabinForecastMap;
00243 };
00244
00245 }
00246 #endif // __STDAIR_BOM_ONDDATE_HPP

```

### 33.393 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

- [stdair](#)

*Handle on the StdAir library context.*

## Functions

- template void [stdair::OnDDateKey::serialize< ba::text\\_oarchive >](#) (ba::text\_oarchive &, unsigned int)
- template void [stdair::OnDDateKey::serialize< ba::text\\_iarchive >](#) (ba::text\_iarchive &, unsigned int)

## 33.394 OnDDateKey.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost Date-Time
00008 #include <boost/date_time/gregorian/formatters.hpp>
00009 // Boost.Serialization
00010 #include <boost/archive/text_iarchive.hpp>
00011 #include <boost/archive/text_oarchive.hpp>
00012 #include <boost/serialization/access.hpp>
00013 // StdAir
00014 #include <stdair/basic/BasConst_Inventory.hpp>
00015 #include <stdair/basic/BasConst_BomDisplay.hpp>
00016 #include <stdair/basic/BasConst_General.hpp>
00017 #include <stdair/bom/OnDDateKey.hpp>
00018 #include <stdair/bom/BomKeyManager.hpp>
00019 #include <stdair/bom/Inventory.hpp>
00020 #include <stdair/bom/FlightDate.hpp>
00021 #include <stdair/bom/SegmentDate.hpp>
00022 #include <stdair/bom/BomDisplay.hpp>
00023
00024 namespace stdair {
00025
00026 // //////////////////////////////////////
00027 OnDDateKey::OnDDateKey()
00028 : _OnDStringList (DEFAULT_OND_STRING_LIST) {
00029     assert (false);
00030 }
00031
00032 // //////////////////////////////////////
00033 OnDDateKey::OnDDateKey (const OnDStringList_T& iOnDStringList)
00034 : _OnDStringList (iOnDStringList) {
00035 }
00036
00037 // //////////////////////////////////////
00038 OnDDateKey::OnDDateKey (const OnDDateKey& iKey)
00039 : _OnDStringList (iKey._OnDStringList) {
00040 }
00041
00042 // //////////////////////////////////////

```

```

00043 OnDDateKey::~OnDDateKey() {
00044 }
00045
00046 // //////////////////////////////////////
00047 const Date_T OnDDateKey::getDate() const {
00048     assert(_OnDStringList.empty() == false);
00049     const OnDString_T& lFrontOnDString = _OnDStringList.front();
00050     return BomKeyManager::extractFlightDateKey (lFrontOnDString).
getDepartureDate();
00051 }
00052
00053 // //////////////////////////////////////
00054 const AirportCode_T OnDDateKey::getOrigin() const {
00055     assert(_OnDStringList.empty() == false);
00056     const OnDString_T& lFrontOnDString = _OnDStringList.front();
00057     return BomKeyManager::extractSegmentDateKey (lFrontOnDString).
getBoardingPoint();
00058 }
00059
00060 // //////////////////////////////////////
00061 const AirportCode_T OnDDateKey::getDestination() const {
00062     assert(_OnDStringList.empty() == false);
00063     const OnDString_T& lLastOnDString = _OnDStringList.back();
00064     return BomKeyManager::extractSegmentDateKey (lLastOnDString).
getOffPoint();
00065 }
00066
00067 // //////////////////////////////////////
00068 void OnDDateKey::toStream (std::ostream& ioOut) const {
00069     ioOut << "OnDDateKey: " << toString();
00070 }
00071
00072 // //////////////////////////////////////
00073 void OnDDateKey::fromStream (std::istream& ioIn) {
00074 }
00075
00076 // //////////////////////////////////////
00077 const std::string OnDDateKey::toString() const {
00078     std::ostringstream oStr;
00079     for (OnDStringList_T::const_iterator itOnDString = _OnDStringList.begin();
00080          itOnDString != _OnDStringList.end(); ++itOnDString) {
00081         oStr << *itOnDString << " ";
00082     }
00083     return oStr.str();
00084 }
00085
00086 // //////////////////////////////////////
00087 void OnDDateKey::serialisationImplementationExport() const {
00088     std::ostringstream oStr;
00089     boost::archive::text_oarchive oa (oStr);
00090     oa << *this;
00091 }
00092
00093 // //////////////////////////////////////
00094 void OnDDateKey::serialisationImplementationImport() {
00095     std::istringstream iStr;
00096     boost::archive::text_iarchive ia (iStr);
00097     ia >> *this;
00098 }
00099
00100 // //////////////////////////////////////
00101 template<class Archive>
00102 void OnDDateKey::serialize (Archive& ioArchive,
00103                             const unsigned int iFileVersion) {
00104 }
00105
00106 // //////////////////////////////////////
00107 // Explicit template instantiation
00108 namespace ba = boost::archive;
00109 template void OnDDateKey::serialize<ba::text_oarchive> (ba::text_oarchive&,
00110                                                         unsigned int);
00111 template void OnDDateKey::serialize<ba::text_iarchive> (ba::text_iarchive&,
00112                                                         unsigned int);
00113 // //////////////////////////////////////
00114 }
00115 }

```

### 33.395 stdair/bom/OnDDateKey.hpp File Reference

```
#include <iosfwd>
```

```
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

## Classes

- struct [stdair::OnDDateKey](#)

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

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.396 OnDDateKey.hpp

```
00001 #ifndef __STDAIR_BOM_ONDDATEKEY_HPP
00002 #define __STDAIR_BOM_ONDDATEKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_basic_types.hpp>
00012 #include <stdair/stdair_demand_types.hpp>
00013 #include <stdair/stdair_date_time_types.hpp>
00014 #include <stdair/bom/KeyAbstract.hpp>
00015
00016 namespace stdair {
00017
00023     struct OnDDateKey : public KeyAbstract {
00024         friend class boost::serialization::access;
00025
00026         // ////////////////////////////////// Constructors and destructors //////////////////////////////////
00027     private:
00031         OnDDateKey();
00032
00033     public:
00037         OnDDateKey (const OnDStringList_T&);
00038
00042         OnDDateKey (const OnDDateKey&);
00043
00047         ~OnDDateKey();
00048
00049
00050     public:
00051         // ////////////////////////////////// Getters //////////////////////////////////
00055         const Date_T getDate() const;
00056
00060         const AirportCode_T getOrigin() const;
00061
00065         const AirportCode_T getDestination() const;
00066
00070         const short getNbOfSegments () const {
00071             return _OnDStringList.size();
00072         }
00073
00074     public:
00075         // ////////////////////////////////// Display support methods //////////////////////////////////
00081         void toStream (std::ostream& ioOut) const;
00082
00088         void fromStream (std::istream& ioIn);
00089
00099         const std::string toString() const;
00100
00101
00102     public:
```

```

00103 // ////////// (Boost) Serialisation support methods //////////
00107 template<class Archive>
00108 void serialize (Archive& ar, const unsigned int iFileVersion);
00109
00110 private:
00115 void serialisationImplementationExport() const;
00116 void serialisationImplementationImport();
00117
00118
00119 private:
00120 // ////////// Attributes //////////
00121 OnDStringList_T _OnDStringList;
00122
00123 };
00124
00125 }
00126 #endif // __STDAIR_BOM_ONDDATEKEY_HPP

```

### 33.397 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

- [stdair](#)  
Handle on the StdAir library context.

#### Typedefs

- typedef std::list< OnDDate \* > [stdair::OnDDateList\\_T](#)
- typedef std::map< const MapKey\_T, OnDDate \* > [stdair::OnDDateMap\\_T](#)
- typedef std::pair< std::string, YieldDemandPair\_T > [stdair::StringDemandStructPair\\_T](#)
- typedef std::map< std::string, YieldDemandPair\_T > [stdair::StringDemandStructMap\\_T](#)
- typedef std::map< std::string, CabinClassPairList\_T > [stdair::StringCabinClassPairListMap\\_T](#)
- typedef std::pair< std::string, CabinClassPairList\_T > [stdair::StringCabinClassPair\\_T](#)
- typedef std::map< CabinCode\_T, WTPDemandPair\_T > [stdair::CabinForecastMap\\_T](#)
- typedef std::pair< CabinCode\_T, WTPDemandPair\_T > [stdair::CabinForecastPair\\_T](#)

### 33.398 OnDDateTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_ONDDATETYPES_HPP
00003 #define __STDAIR_BOM_ONDDATETYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // Stdair
00012 #include <stdair/bom/key_types.hpp>
00013 #include <stdair/stdair_maths_types.hpp>
00014 #include <stdair/stdair_demand_types.hpp>
00015
00016 namespace stdair {
00017
00018 // Forward declarations.
00019 class OnDDate;
00020
00022 typedef std::list<OnDDate*> OnDDateList_T;
00023

```

```

00025     typedef std::map<const MapKey_T, OnDDate*> OnDDateMap_T;
00026
00032     typedef std::pair<std::string, YieldDemandPair_T> StringDemandStructPair_T;
00033     typedef std::map<std::string, YieldDemandPair_T> StringDemandStructMap_T;
00034
00041     typedef std::map<std::string, CabinClassPairList_T>
StringCabinClassPairListMap_T;
00042     typedef std::pair<std::string, CabinClassPairList_T> StringCabinClassPair_T;
00043
00048     typedef std::map<CabinCode_T, WTPDemandPair_T> CabinForecastMap_T;
00049     typedef std::pair<CabinCode_T, WTPDemandPair_T> CabinForecastPair_T;
00050
00051 }
00052 #endif // __STDAIR_BOM_ONDDATETYPES_HPP

```

### 33.399 stdair/bom/OptimisationNotificationStruct.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/bom/OptimisationNotificationStruct.hpp>

```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.400 OptimisationNotificationStruct.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/bom/OptimisationNotificationStruct.hpp>
00009
00010 namespace stdair {
00011
00012 // //////////////////////////////////////
00013 OptimisationNotificationStruct::OptimisationNotificationStruct()
00014 : _partySize (0), _stayDuration (0), _wtp (0.0), _valueOfTime (0.0) {
00015     assert (false);
00016 }
00017
00018 // //////////////////////////////////////
00019 OptimisationNotificationStruct::
00020 OptimisationNotificationStruct (const OptimisationNotificationStruct&
iOptimisationNotification)
00021 : _origin (iOptimisationNotification._origin),
00022   _destination (iOptimisationNotification._destination),
00023   _pos (iOptimisationNotification._pos),
00024   _preferredDepartureDate (iOptimisationNotification._preferredDepartureDate),
00025   _notificationDateTime (iOptimisationNotification._notificationDateTime),
00026   _preferredCabin (iOptimisationNotification._preferredCabin),
00027   _partySize (iOptimisationNotification._partySize),
00028   _channel (iOptimisationNotification._channel),
00029   _tripType (iOptimisationNotification._tripType),
00030   _stayDuration (iOptimisationNotification._stayDuration),
00031   _frequentFlyerType (iOptimisationNotification._frequentFlyerType),
00032   _preferredDepartureTime (iOptimisationNotification._preferredDepartureTime),
00033   _wtp (iOptimisationNotification._wtp),
00034   _valueOfTime (iOptimisationNotification._valueOfTime) {
00035 }
00036
00037 // //////////////////////////////////////
00038 OptimisationNotificationStruct::
00039 OptimisationNotificationStruct (const AirportCode_T& iOrigin,
00040                                const AirportCode_T& iDestination,
00041                                const CityCode_T& iPOS,
00042                                const Date_T& iDepartureDate,
00043                                const DateTime_T& iNotificationDateTime,
00044                                const CabinCode_T& iPreferredCabin,
00045                                const NbOfSeats_T& iPartySize,

```

```

00046         const ChannelLabel_T& iChannel,
00047         const TripType_T& iTripType,
00048         const DayDuration_T& iStayDuration,
00049         const FrequentFlyer_T& iFrequentFlyerType,
00050         const Duration_T& iPreferredDepartureTime,
00051         const WTP_T& iWTP,
00052         const PriceValue_T& iValueOfTime)
00053     : _origin (iOrigin), _destination (iDestination),
00054       _pos (iPOS), _preferredDepartureDate (iDepartureDate),
00055       _notificationDateTime (iNotificationDateTime),
00056       _preferredCabin (iPreferredCabin), _partySize (iPartySize),
00057       _channel (iChannel), _tripType (iTripType),
00058       _stayDuration (iStayDuration), _frequentFlyerType (iFrequentFlyerType),
00059       _preferredDepartureTime (iPreferredDepartureTime), _wtp (iWTP),
00060       _valueOfTime (iValueOfTime) {
00061     }
00062
00063     // //////////////////////////////////////
00064     OptimisationNotificationStruct::~OptimisationNotificationStruct
00065     () {
00066     }
00067
00068     // //////////////////////////////////////
00069     void OptimisationNotificationStruct::toStream (std::ostream&
00070     ioOut) const {
00071         ioOut << describe();
00072     }
00073
00074     // //////////////////////////////////////
00075     void OptimisationNotificationStruct::fromStream (std::istream&
00076     ioIn) {
00077     }
00078
00079     // //////////////////////////////////////
00080     const std::string OptimisationNotificationStruct::describe()
00081     const {
00082         std::ostringstream oStr;
00083         oStr << "At " << _notificationDateTime
00084         << ", for (" << _pos << ") " << _origin << "-" << _destination
00085         << " " << _preferredDepartureDate << " " << _preferredCabin
00086         << " " << _partySize << " " << _channel << " " << _tripType
00087         << " " << _stayDuration << " " << _frequentFlyerType
00088         << " " << _preferredDepartureTime << " " << _wtp
00089         << " " << _valueOfTime;
00090         return oStr.str();
00091     }
00092 }

```

### 33.401 stdair/bom/OptimisationNotificationStruct.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/OptimisationNotificationTypes.hpp>

```

#### Classes

- struct [stdair::OptimisationNotificationStruct](#)

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.402 OptimisationNotificationStruct.hpp

```
00001 #ifndef __STDAIR_BOM_OPTIMISATIONNOTIFICATIONSTRUCT_HPP
```

```

00002 #define __STDAIR_BOM_OPTIMISATIONNOTIFICATIONSTRUCT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/stdair_demand_types.hpp>
00013 #include <stdair/basic/StructAbstract.hpp>
00014 #include <stdair/bom/OptimisationNotificationTypes.hpp>
00015
00016 namespace stdair {
00017
00019     struct OptimisationNotificationStruct : public
StructAbstract {
00020     public:
00021         // ////////////////////////////////////// Getters //////////////////////////////////////
00023         const AirportCode_T& getOrigin() const {
00024             return _origin;
00025         }
00026
00028         const AirportCode_T& getDestination() const {
00029             return _destination;
00030         }
00031
00033         const CityCode_T& getPOS() const {
00034             return _pos;
00035         }
00036
00038         const Date_T& getPreferedDepartureDate() const {
00039             return _preferredDepartureDate;
00040         }
00041
00043         const DateTime_T& getNotificationDateTime() const {
00044             return _notificationDateTime;
00045         }
00046
00048         const CabinCode_T& getPreferredCabin() const {
00049             return _preferredCabin;
00050         }
00051
00053         const NbOfSeats_T& getPartySize() const {
00054             return _partySize;
00055         }
00056
00058         const ChannelLabel_T& getOptimisationChannel() const {
00059             return _channel;
00060         }
00061
00063         const TripType_T& getTripType() const {
00064             return _tripType;
00065         }
00066
00068         const DayDuration_T& getStayDuration() const {
00069             return _stayDuration;
00070         }
00071
00073         const FrequentFlyer_T& getFrequentFlyerType() const {
00074             return _frequentFlyerType;
00075         }
00076
00078         const Duration_T& getPreferredDepartureTime() const {
00079             return _preferredDepartureTime;
00080         }
00081
00083         const WTP_T& getWTP() const {
00084             return _wtp;
00085         }
00086
00088         const PriceValue_T& getValueOfTime () const {
00089             return _valueOfTime;
00090         }
00091
00092         // ////////////////////////////////////// Display support method //////////////////////////////////////
00095         void toStream (std::ostream& ioOut) const;
00096
00099         void fromStream (std::istream& ioIn);
00100
00102         const std::string describe() const;
00103
00104         // ////////////////////////////////////// Constructors and Destructors //////////////////////////////////////
00105     public:
00106         OptimisationNotificationStruct (const
00108

```

```

00109     AirportCode_T& iOrigin,
00110         const AirportCode_T& iDestination,
00111         const CityCode_T& iPOS,
00112         const Date_T& iDepartureDate,
00113         const DateTime_T& iNotificationDateTime,
00114         const CabinCode_T& iPreferredCabin,
00115         const NbOfSeats_T& iPartySize,
00116         const ChannelLabel_T& iChannel,
00117         const TripType_T& iTripType,
00118         const DayDuration_T& iStayDuration,
00119         const FrequentFlyer_T& iFrequentFlyerType,
00120         const Duration_T& iPreferredDepartureTime,
00121         const WTP_T& iWTP,
00122         const PriceValue_T& iValueOfTime);
00123
00124     OptimisationNotificationStruct (const
OptimisationNotificationStruct&);
00125
00126     private:
00127         OptimisationNotificationStruct ();
00128
00129     public:
00130         ~OptimisationNotificationStruct ();
00131
00132     private:
00133         // ////////////////////////////////// Attributes //////////////////////////////////
00134         const AirportCode_T _origin;
00135
00136         const AirportCode_T _destination;
00137
00138         const CityCode_T _pos;
00139
00140         const Date_T _preferredDepartureDate;
00141
00142         const DateTime_T _notificationDateTime;
00143
00144         const CabinCode_T _preferredCabin;
00145
00146         const NbOfSeats_T _partySize;
00147
00148         const ChannelLabel_T _channel;
00149
00150         const TripType_T _tripType;
00151
00152         const DayDuration_T _stayDuration;
00153
00154         const FrequentFlyer_T _frequentFlyerType;
00155
00156         const Duration_T _preferredDepartureTime;
00157
00158         const WTP_T _wtp;
00159
00160         const PriceValue_T _valueOfTime;
00161     };
00162 }
00163 #endif // __STDAIR_BOM_OPTIMISATIONNOTIFICATIONSTRUCT_HPP

```

### 33.403 stdair/bom/OptimisationNotificationTypes.hpp File Reference

```
#include <boost/shared_ptr.hpp>
```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

#### Typedefs

- typedef boost::shared\_ptr< OptimisationNotificationStruct > [stdair::OptimisationNotificationPtr\\_T](#)



### 33.404 OptimisationNotificationTypes.hpp

```

00001 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00002 #ifndef __STDAIR_BOM_OPTIMISATIONNOTIFICATIONTYPES_HPP
00003 #define __STDAIR_BOM_OPTIMISATIONNOTIFICATIONTYPES_HPP
00004
00005 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00006 // Import section
00007 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00008 // Boost
00009 #include <boost/shared_ptr.hpp>
00010
00011 namespace stdair {
00012
00013     // Forward declarations
00014     struct OptimisationNotificationStruct;
00015
00016     ////////////////////////////////////////////////////////////////// Type definitions //////////////////////////////////////////////////////////////////
00017     typedef boost::
00018         shared_ptr<OptimisationNotificationStruct> OptimisationNotificationPtr_T;
00019
00020 }
00021
00022 #endif // __STDAIR_BOM_OPTIMISATIONNOTIFICATIONTYPES_HPP
00023

```

### 33.405 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

- [stdair](#)

*Handle on the StdAir library context.*

#### Functions

- const boost::char\_separator< char > [stdair::TokeniserDashSeparator](#) ("-")
- const boost::char\_separator< char > [stdair::TokeniserTimeSeparator](#) (":")

### 33.406 ParsedKey.cpp

```

00001 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00002 // Import section
00003 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost
00008 #include <boost/tokenizer.hpp>
00009 #include <boost/lexical_cast.hpp>
00010 #include <boost/date_time/gregorian/parsers.hpp>
00011 // StdAir

```

```

00012 #include <stdair/stdair_exceptions.hpp>
00013 #include <stdair/basic/BasConst_Inventory.hpp>
00014 #include <stdair/basic/BasConst_BomDisplay.hpp>
00015 #include <stdair/bom/InventoryKey.hpp>
00016 #include <stdair/bom/FlightDateKey.hpp>
00017 #include <stdair/bom/SegmentDateKey.hpp>
00018 #include <stdair/bom/LegDateKey.hpp>
00019 #include <stdair/bom/ParsedKey.hpp>
00020 #include <stdair/service/Logger.hpp>
00021
00022 namespace stdair {
00023
00024 // ////////////////// Tokenising support //////////////////
00025 typedef boost::tokenizer<boost::char_separator<char> > Tokeniser_T;
00026
00027 const boost::char_separator<char> TokeniserDashSeparator ("-");
00028
00029 const boost::char_separator<char> TokeniserTimeSeparator (":");
00030
00031 // //////////////////////////////////////
00032 ParsedKey::ParsedKey() : _fullKey (""), _airlineCode (""), _flightNumber (""),
00033                         _departureDate (""), _boardingPoint (""),
00034                         _offPoint (""), _boardingTime ("") {
00035 }
00036
00037 // //////////////////////////////////////
00038 ParsedKey::~ParsedKey() {
00039 }
00040
00041 // //////////////////////////////////////
00042 InventoryKey ParsedKey::getInventoryKey() const {
00043     if (_airlineCode.size() < 2 || _airlineCode.size() > 3) {
00044         STDAIR_LOG_ERROR ("No airline code can be found in '" <<
00045 _fullKey << "'");
00046         STDAIR_LOG_DEBUG ("Parsed key: " << toString());
00047         throw KeyNotFoundException ("No airline code can be found in '"
00048 + _fullKey + "'");
00049     }
00050     return _airlineCode;
00051 }
00052
00053 // //////////////////////////////////////
00054 FlightDateKey ParsedKey::getFlightDateKey() const {
00055     // Check whether the departure date has been parsed correctly.
00056     Tokeniser_T lDateTokens (_departureDate,
00057 TokeniserDashSeparator);
00058
00059     if (lDateTokens.begin() == lDateTokens.end()) {
00060         STDAIR_LOG_ERROR ("No date can be found in '" << _fullKey << "'");
00061         STDAIR_LOG_DEBUG ("Parsed key: " << toString());
00062         throw KeyNotFoundException ("No date can be found in '" +
00063 _fullKey + "'");
00064     }
00065
00066     const FlightNumber_T lFlightNumber =
00067         boost::lexical_cast<FlightNumber_T> (_flightNumber);
00068
00069     const Date_T lDepartureDate =
00070         boost::gregorian::from_simple_string (_departureDate);
00071
00072     const FlightDateKey oFlightDateKey (lFlightNumber, lDepartureDate);
00073
00074     return oFlightDateKey;
00075 }
00076
00077 // //////////////////////////////////////
00078 LegDateKey ParsedKey::getLegKey() const {
00079     if (_boardingPoint.size() != 3) {
00080         STDAIR_LOG_ERROR ("No airport code can be found in '" <<
00081 _fullKey << "'");
00082         STDAIR_LOG_DEBUG ("Parsed key: " << toString());
00083         throw KeyNotFoundException ("No airport code can be found in '"
00084 + _fullKey + "'");
00085     }
00086
00087     const LegDateKey oLegDateKey (_boardingPoint);
00088
00089     return oLegDateKey;
00090 }
00091
00092 // //////////////////////////////////////
00093 SegmentDateKey ParsedKey::getSegmentKey() const {
00094     if (_boardingPoint.size() != 3 || _offPoint.size() != 3) {
00095         STDAIR_LOG_ERROR ("No airport code can be found in '" <<
00096 _fullKey << "'");
00097         STDAIR_LOG_DEBUG ("Parsed key: " << toString());
00098         throw KeyNotFoundException ("No airport code can be found in '"
00099

```

```

00103                                     + _fullKey + "'");
00104     }
00105
00106     const SegmentDateKey oSegmentDateKey (_boardingPoint,
00107     _offPoint);
00108
00109     return oSegmentDateKey;
00110 }
00111
00112 // //////////////////////////////////////
00113 const Duration_T ParsedKey::getBoardingTime() const {
00114     // Check whether the boarding time has been parsed correctly.
00115     Tokeniser_T lTimeTokens (_boardingTime, TokeniserTimeSeparator);
00116
00117     if (lTimeTokens.begin() == lTimeTokens.end()) {
00118         STDAIR_LOG_ERROR ("No boarding time can be found in '" <<
00119         _fullKey << "'");
00120
00121         STDAIR_LOG_DEBUG ("Parsed key: " << toString());
00122         throw KeyNotFoundException ("No boarding time can be found in '"
00123         + _fullKey + "'");
00124     }
00125
00126     const Duration_T oBoardingTime (boost::posix_time::
00127     duration_from_string (_boardingTime));
00128
00129     return oBoardingTime;
00130 }
00131
00132 // //////////////////////////////////////
00133 void ParsedKey::toStream (std::ostream& ioOut) const {
00134     ioOut << "ParsedKey: " << toString();
00135 }
00136
00137 // //////////////////////////////////////
00138 void ParsedKey::fromStream (std::istream& ioIn) {
00139 }
00140
00141 // //////////////////////////////////////
00142 const std::string ParsedKey::toString() const {
00143     std::ostringstream oStr;
00144
00145     oStr << _airlineCode
00146     << DEFAULT_KEY_FLD_DELIMITER << " "
00147     << _flightNumber
00148     << DEFAULT_KEY_SUB_FLD_DELIMITER << " "
00149     << _departureDate
00150     << DEFAULT_KEY_FLD_DELIMITER << " "
00151     << _boardingPoint
00152     << DEFAULT_KEY_SUB_FLD_DELIMITER << " "
00153     << _offPoint
00154     << DEFAULT_KEY_FLD_DELIMITER << " "
00155     << _boardingTime;
00156
00157     return oStr.str();
00158 }
00159 }

```

### 33.407 stdair/bom/ParsedKey.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>

```

#### Classes

- struct [stdair::ParsedKey](#)

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.408 ParsedKey.hpp

```

00001 #ifndef __STDAIR_BOM_PARSEDKEY_HPP
00002 #define __STDAIR_BOM_PARSEDKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_date_time_types.hpp>
00012 #include <stdair/bom/KeyAbstract.hpp>
00013
00014 namespace stdair {
00015     struct InventoryKey;
00016     struct FlightDateKey;
00017     struct SegmentDateKey;
00018     struct LegDateKey;
00019
00020     struct ParsedKey : public KeyAbstract{
00021     public:
00022         // /////////////////////////////////// Getter ///////////////////////////////////
00023         InventoryKey getInventoryKey () const;
00024         FlightDateKey getFlightDateKey () const;
00025         SegmentDateKey getSegmentKey () const;
00026         LegDateKey getLegKey () const;
00027         const Duration_T getBoardingTime () const;
00028     public:
00029         // /////////////////////////////////// Display support methods ///////////////////////////////////
00030         void toStream (std::ostream& ioOut) const;
00031         void fromStream (std::istream& ioIn);
00032         const std::string toString() const;
00033     public:
00034         // /////////////////////////////////// Constructor and destructor. ///////////////////////////////////
00035         // Default constructor
00036         ParsedKey ();
00037         // Default destructor
00038         ~ParsedKey ();
00039     public:
00040         // /////////////////////////////////// Attributes ///////////////////////////////////
00041         std::string _fullKey;
00042         std::string _airlineCode;
00043         std::string _flightNumber;
00044         std::string _departureDate;
00045         std::string _boardingPoint;
00046         std::string _offPoint;
00047         std::string _boardingTime;
00048     };
00049 }
00050 #endif // __STDAIR_BOM_PARSEDKEY_HPP

```

## 33.409 stdair/bom/PeriodStruct.cpp File Reference

```

#include <sstream>
#include <cassert>
#include <stdair/basic/BasConst_Period_BOM.hpp>
#include <stdair/bom/PeriodStruct.hpp>

```

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.410 PeriodStruct.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <sstream>
00006 #include <cassert>
00007 // StdAir
00008 #include <stdair/basic/BasConst_Period_BOM.hpp>
00009 #include <stdair/bom/PeriodStruct.hpp>
00010
00011 namespace stdair {
00012
00013     // //////////////////////////////////////
00014     PeriodStruct::PeriodStruct ()
00015     : _dateRange (BOOST_DEFAULT_DATE_PERIOD), _dow () {
00016     }
00017
00018     // //////////////////////////////////////
00019     PeriodStruct::PeriodStruct (const DatePeriod_T& iDateRange,
00020                                 const DoWStruct& iDoW)
00021     : _dateRange (iDateRange), _dow (iDoW) {
00022     }
00023
00024     // //////////////////////////////////////
00025     PeriodStruct::PeriodStruct (const PeriodStruct& iPeriodStruct)
00026     : _dateRange (iPeriodStruct._dateRange), _dow (iPeriodStruct._dow) {
00027     }
00028
00029     // //////////////////////////////////////
00030     const std::string PeriodStruct::describeShort() const {
00031         std::ostringstream ostr;
00032         ostr << _dateRange << ", " << _dow.describeShort ();
00033         return ostr.str();
00034     }
00035
00036     // //////////////////////////////////////
00037     const std::string PeriodStruct::describe() const {
00038         std::ostringstream ostr;
00039         ostr << _dateRange << ", " << _dow.describe ();
00040         return ostr.str();
00041     }
00042
00043     // //////////////////////////////////////
00044     PeriodStruct PeriodStruct::
00045     addDateOffset (const DateOffset_T& iDateOffset) const {
00046         // Create a new date range by shifting the date range of this object with
00047         // iDateOffset.
00048         DatePeriod_T lNewDateRange = getDateRange();
00049         lNewDateRange.shift (iDateOffset);
00050
00051         // Create a new DoWStruct by shifting the DoWStruct of this object with
00052         // iDateOffset.
00053         const long lNbOfDaysOffset = iDateOffset.days();
00054         const DoWStruct& lDoW = getDoW();
00055         const DoWStruct lNewDoW = lDoW.shift (lNbOfDaysOffset);
00056
00057         return PeriodStruct (lNewDateRange, lNewDoW);
00058     }
00059
00060     // //////////////////////////////////////
00061     PeriodStruct PeriodStruct::
00062     intersection (const PeriodStruct& iPeriodStruct) const {
00063         const DatePeriod_T lNewDateRange =
00064             _dateRange.intersection (iPeriodStruct._dateRange);
00065         const DoWStruct lNewDoW = _dow.intersection (iPeriodStruct._dow);
00066
00067         return PeriodStruct (lNewDateRange, lNewDoW);
00068     }
00069
00070     // //////////////////////////////////////
00071     const bool PeriodStruct::isValid () const {
00072         if (_dateRange.is_null() == false && _dow.isValid()) {
00073             return true;
00074         }
00075         return false;
00076     }
00077 }
00078
00079 }

```

## 33.411 stdair/bom/PeriodStruct.hpp File Reference

```
#include <string>
#include <vector>
#include <stdair/stdair_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/DoWStruct.hpp>
```

## Classes

- struct [stdair::PeriodStruct](#)

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.412 PeriodStruct.hpp

```
00001 #ifndef __STDAIR_BOM_PERIODSTRUCT_HPP
00002 #define __STDAIR_BOM_PERIODSTRUCT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010 // StdAir
00011 #include <stdair/stdair_types.hpp>
00012 #include <stdair/basic/StructAbstract.hpp>
00013 #include <stdair/bom/DoWStruct.hpp>
00014
00015 namespace stdair {
00016
00019     struct PeriodStruct : public StructAbstract {
00020     public:
00021         // //////////////////////////////////////
00023         const DatePeriod_T& getDateRange () const {
00024             return _dateRange;
00025         }
00026         const DoWStruct& getDoW () const {
00027             return _dow;
00028         }
00029
00030     public:
00031         // //////////////////////////////////////
00033         void setDateRange (const DatePeriod_T& iDateRange) {
00034             _dateRange = iDateRange;
00035         }
00036         void setDoW (const DoWStruct& iDoW) { _dow = iDoW; }
00037
00038     public:
00040         const std::string describe() const;
00041
00043         const std::string describeShort() const;
00044
00045     public:
00046         // //////////////////////////////////////
00048         PeriodStruct addDateOffset (const DateOffset_T&) const;
00049
00052         PeriodStruct intersection (const PeriodStruct&) const;
00053
00055         const bool isValid () const;
00056
00057     public:
00059         PeriodStruct (const DatePeriod_T&, const DoWStruct&);
00061         PeriodStruct ();
00062         PeriodStruct (const PeriodStruct&);
00064         ~PeriodStruct () { }
00065
00066     private:
00067         // Attributes
```

```

00068     DatePeriod_T _dateRange;
00069     DoWStruct _dow;
00070 };
00071
00072 }
00073 #endif // __STDAIR_BOM_PERIODSTRUCT_HPP

```

### 33.413 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

- [stdair](#)

*Handle on the StdAir library context.*

### 33.414 Policy.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <sstream>
00006 #include <cassert>
00007 #include <iomanip>
00008 #include <iostream>
00009 //STDAIR
00010 #include <stdair/basic/BasConst_Inventory.hpp>
00011 #include <stdair/bom/BomManager.hpp>
00012 #include <stdair/bom/BookingClass.hpp>
00013 #include <stdair/bom/BookingClassTypes.hpp>
00014 #include <stdair/bom/Policy.hpp>
00015
00016 namespace stdair {
00017
00018     // //////////////////////////////////////
00019     Policy::Policy () :
00020         _key (DEFAULT_POLICY_CODE), _parent (NULL) {
00021         assert (false);
00022     }
00023
00024     // //////////////////////////////////////
00025     Policy::Policy (const Policy& iPolicy)
00026     : _key (DEFAULT_POLICY_CODE), _parent (NULL) {
00027         assert (false);
00028     }
00029
00030     // //////////////////////////////////////
00031     Policy::Policy (const Key_T& iKey) : _key (iKey), _parent (NULL) {
00032     }
00033
00034     // //////////////////////////////////////
00035     Policy::~Policy() {
00036     }
00037
00038     // //////////////////////////////////////
00039     std::string Policy::toString () const {
00040         std::ostringstream oStr;
00041         oStr << describeKey();
00042
00043         oStr << std::fixed << std::setprecision (2)
00044             << " "; " << _demand
00045             << " "; " << _stdDev

```

```

00046         << " "; " << _yield << std::endl;
00047
00048         return oStr.str();
00049     }
00050
00051     // //////////////////////////////////////
00052     const BookingClassList_T& Policy::getBookingClassList()
00053     const {
00054         return BomManager::getList<BookingClass> (*this);
00055     }
00056
00057     // //////////////////////////////////////
00058     const Revenue_T Policy::getTotalRevenue () const {
00059         Revenue_T oTotalRevenue = 0.0;
00060         for (YieldDemandMap_T::const_iterator itYD = _yieldDemandMap.begin();
00061              itYD != _yieldDemandMap.end(); ++itYD) {
00062             const Yield_T& lYield = itYD->first;
00063             const double& lDemand = itYD->second;
00064             oTotalRevenue += lYield*lDemand;
00065         }
00066         return oTotalRevenue;
00067     }
00068
00069     // //////////////////////////////////////
00070     void Policy::addYieldDemand (const Yield_T& iYield,
00071                                 const NbOfBookings_T& iDemand) {
00072         YieldDemandMap_T::iterator itYD = _yieldDemandMap.find (iYield);
00073         if (itYD == _yieldDemandMap.end()) {
00074             bool insert = _yieldDemandMap.insert (YieldDemandMap_T::value_type
00075                                                    (iYield, iDemand)).second;
00076             assert (insert == true);
00077         } else {
00078             NbOfBookings_T& lDemand = itYD->second;
00079             lDemand += iDemand;
00080         }
00081     }
00082
00083 }

```

### 33.415 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>

```

#### Classes

- class [stdair::Policy](#)

#### Namespaces

- [boost](#)

*Forward declarations.*

- [boost::serialization](#)
- [stdair](#)

*Handle on the StdAir library context.*

### 33.416 Policy.hpp

```

00001 #ifndef __STDAIR_BOM_POLICY_HPP
00002 #define __STDAIR_BOM_POLICY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section

```



```

00006 // //////////////////////////////////////
00007 // STL
00008 #include <cmath>
00009 // StdAir
00010 #include <stdair/stdair_basic_types.hpp>
00011 #include <stdair/stdair_rm_types.hpp>
00012 #include <stdair/bom/BomAbstract.hpp>
00013 #include <stdair/bom/BookingClassTypes.hpp>
00014 #include <stdair/bom/PolicyKey.hpp>
00015
00016 namespace boost {
00017     namespace serialization {
00018         class access;
00019     }
00020 }
00021
00022 namespace stdair {
00023
00024     class Policy : public BomAbstract {
00025     public:
00026         template <typename BOM> friend class FacBom;
00027         friend class FacBomManager;
00028         friend class boost::serialization::access;
00029
00030         // ////////////////////////////////// Type definitions //////////////////////////////////
00031         typedef PolicyKey Key_T;
00032
00033     public:
00034         // ////////////////////////////////// Getters //////////////////////////////////
00035         const Key_T& getKey() const {
00036             return _key;
00037         }
00038
00039         BomAbstract* const getParent() const {
00040             return _parent;
00041         }
00042
00043         const HolderMap_T& getHolderMap() const {
00044             return _holderMap;
00045         }
00046
00047         const BookingClassList_T& getBookingClassList() const;
00048
00049         const NbOfBookings_T& getDemand() const {
00050             return _demand;
00051         }
00052
00053         const StdDevValue_T& getStdDev() const {
00054             return _stdDev;
00055         }
00056
00057         const Yield_T& getYield() const {
00058             return _yield;
00059         }
00060
00061         const Revenue_T getTotalRevenue () const;
00062
00063     public:
00064         // ////////////////////////////////// Setters //////////////////////////////////
00065         void setDemand (const NbOfBookings_T& iDemand) {
00066             _demand = iDemand;
00067         }
00068
00069         void setStdDev (const StdDevValue_T& iStdDev) {
00070             _stdDev = iStdDev;
00071         }
00072
00073         void setYield (const Yield_T& iYield) {
00074             _yield = iYield;
00075         }
00076
00077         void resetDemandForecast () {
00078             _demand = 0.0;
00079             _stdDev = 0.0;
00080             _yieldDemandMap.clear();
00081         }
00082
00083         void addYieldDemand (const Yield_T&, const
00084             NbOfBookings_T&);
00085
00086     public:
00087         // ////////////////////////////////// Display support methods //////////////////////////////////
00088         void toStream (std::ostream& ioOut) const {
00089             ioOut << toString();
00090         }
00091
00092         void fromStream (std::istream& ioIn) {

```

```

00126     }
00127
00131     std::string toString() const;
00132
00136     const std::string describeKey() const {
00137         return _key.toString();
00138     }
00139
00140
00141 public:
00142     // ////////// (Boost) Serialisation support methods //////////
00146     template<class Archive>
00147     void serialize (Archive& ar, const unsigned int iFileVersion);
00148
00149 private:
00157     void serialisationImplementationExport() const;
00158     void serialisationImplementationImport();
00159
00160
00161 protected:
00162     // ////////// Constructors and destructor. //////////
00166     Policy (const Key_T&);
00167
00171     virtual ~Policy();
00172
00173 private:
00177     Policy();
00178
00182     Policy (const Policy&);
00183
00184
00185 private:
00186     // ////////// Attributes //////////
00190     Key_T _key;
00191
00195     BomAbstract* _parent;
00196
00200     HolderMap_T _holderMap;
00201
00205     NbOfBookings_T _demand;
00206
00210     StdDevValue_T _stdDev;
00211
00215     Yield_T _yield;
00216
00220     YieldDemandMap_T _yieldDemandMap;
00221
00222 };
00223 }
00224 #endif // __STDAIR_BOM_POLICY_HPP

```

### 33.417 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

- [stdair](#)

*Handle on the StdAir library context.*

#### 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)

## 33.418 PolicyKey.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/access.hpp>
00011 // StdAir
00012 #include <stdair/basic/BasConst_Inventory.hpp>
00013 #include <stdair/bom/PolicyKey.hpp>
00014
00015 namespace stdair {
00016
00017 // //////////////////////////////////////
00018 PolicyKey::PolicyKey() : _policyCode (DEFAULT_POLICY_CODE) {
00019     assert (false);
00020 }
00021
00022 // //////////////////////////////////////
00023 PolicyKey::PolicyKey (const PolicyKey& iPolicyKey)
00024 : _policyCode (iPolicyKey._policyCode) {
00025 }
00026
00027 // //////////////////////////////////////
00028 PolicyKey::PolicyKey (const PolicyCode_T& iPolicyCode)
00029 : _policyCode (iPolicyCode) {
00030 }
00031
00032 // //////////////////////////////////////
00033 PolicyKey::~PolicyKey() {
00034 }
00035
00036 // //////////////////////////////////////
00037 void PolicyKey::toStream (std::ostream& ioOut) const {
00038     ioOut << "PolicyKey: " << toString();
00039 }
00040
00041 // //////////////////////////////////////
00042 void PolicyKey::fromStream (std::istream& ioIn) {
00043 }
00044
00045 // //////////////////////////////////////
00046 const std::string PolicyKey::toString() const {
00047     std::ostringstream oStr;
00048     oStr << _policyCode;
00049     return oStr.str();
00050 }
00051
00052 // //////////////////////////////////////
00053 void PolicyKey::serialisationImplementationExport() const {
00054     std::ostringstream oStr;
00055     boost::archive::text_oarchive oa (oStr);
00056     oa << *this;
00057 }
00058
00059 // //////////////////////////////////////
00060 void PolicyKey::serialisationImplementationImport() {
00061     std::istringstream iStr;
00062     boost::archive::text_iarchive ia (iStr);
00063     ia >> *this;
00064 }
00065
00066 // //////////////////////////////////////
00067 template<class Archive>
00068 void PolicyKey::serialize (Archive& ioArchive,
00069                             const unsigned int iFileVersion) {
00070     ioArchive & _policyCode;
00071 }
00072
00073 // //////////////////////////////////////
00074 // Explicit template instantiation
00075 namespace ba = boost::archive;
00076 template void PolicyKey::serialize<ba::text_oarchive> (ba::text_oarchive&,
00077                                                         unsigned int);
00078 template void PolicyKey::serialize<ba::text_iarchive> (ba::text_iarchive&,
00079                                                         unsigned int);
00080
00081 // //////////////////////////////////////
00082 }
00083
00084
00085
00086 }

```

## 33.419 stdair/bom/PolicyKey.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

## Classes

- struct [stdair::PolicyKey](#)  
*Key of a given policy, made of a policy code.*

## Namespaces

- [boost](#)  
*Forward declarations.*
- [boost::serialization](#)
- [stdair](#)  
*Handle on the StdAir library context.*

## 33.420 PolicyKey.hpp

```
00001 #ifndef __STDAIR_BOM_POLICYKEY_HPP
00002 #define __STDAIR_BOM_POLICYKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/bom/KeyAbstract.hpp>
00013
00015 namespace boost {
00016     namespace serialization {
00017         class access;
00018     }
00019 }
00020
00021 namespace stdair {
00022
00026     struct PolicyKey : public KeyAbstract {
00027         friend class boost::serialization::access;
00028
00029         // ////////////////////////////////// Constructors and destructors //////////////////////////////////
00030     private:
00031         PolicyKey();
00032
00033     public:
00034         PolicyKey (const PolicyCode_T& iPolicyCode);
00035
00036         PolicyKey (const PolicyKey&);
00037
00038         ~PolicyKey();
00039
00040     public:
00041         // ////////////////////////////////// Getters //////////////////////////////////
00042         const PolicyCode_T& getPolicyCode () const {
00043             return _policyCode;
00044         }
00045
00046     public:
00047         // ////////////////////////////////// Display support methods //////////////////////////////////
00048         void toStream (std::ostream& ioOut) const;
00049
00050         void fromStream (std::istream& ioIn);
00051
00052     };
00053
00054 }
```

```

00076
00086     const std::string toString() const;
00087
00088
00089 public:
00090     // //////////// (Boost) Serialisation support methods ////////////
00094     template<class Archive>
00095     void serialize (Archive& ar, const unsigned int iFileVersion);
00096
00097 private:
00102     void serialisationImplementationExport() const;
00103     void serialisationImplementationImport();
00104
00105
00106 private:
00107     // //////////// Attributes ////////////
00111     PolicyCode_T _policyCode;
00112 };
00113
00114 }
00115 #endif // __STDAIR_BOM_POLICYKEY_HPP

```

### 33.421 stdair/bom/PolicyTypes.hpp File Reference

```

#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>

```

#### Namespaces

- [stdair](#)  
Handle on the StdAir library context.

#### Typedefs

- typedef std::list< Policy \* > [stdair::PolicyList\\_T](#)
- typedef std::map< const MapKey\_T, Policy \* > [stdair::PolicyMap\\_T](#)

### 33.422 PolicyTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_POLICYTYPES_HPP
00003 #define __STDAIR_BOM_POLICYTYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // StdAir
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016     // Forward declarations.
00017     class Policy;
00018
00020     typedef std::list<Policy*> PolicyList_T;
00021
00023     typedef std::map<const MapKey_T, Policy*> PolicyMap_T;
00024
00025 }
00026 #endif // __STDAIR_BOM_POLICYTYPES_HPP

```

### 33.423 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

- [stdair](#)

*Handle on the StdAir library context.*

## 33.424 PosChannel.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_Request.hpp>
00009 #include <stdair/service/Logger.hpp>
00010 #include <stdair/bom/PosChannel.hpp>
00011
00012 namespace stdair {
00013
00014 // //////////////////////////////////////
00015 PosChannel::PosChannel()
00016 : _key (DEFAULT_POS,
00017        DEFAULT_CHANNEL),
00018   _parent (NULL) {
00019     // That constructor is used by the serialisation process
00020 }
00021
00022 // //////////////////////////////////////
00023 PosChannel::PosChannel (const PosChannel& iPosChannel)
00024 : _key (iPosChannel.getKey()), _parent (NULL) {
00025 }
00026
00027 // //////////////////////////////////////
00028 PosChannel::PosChannel (const Key_T& iKey)
00029 : _key (iKey), _parent (NULL) {
00030 }
00031
00032 // //////////////////////////////////////
00033 PosChannel::~PosChannel () {
00034 }
00035
00036 // //////////////////////////////////////
00037 std::string PosChannel::toString() const {
00038     std::ostringstream oStr;
00039     oStr << describeKey();
00040     return oStr.str();
00041 }
00042 }
```

## 33.425 stdair/bom/PosChannel.hpp File Reference

```
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/PosChannelKey.hpp>
#include <stdair/bom/PosChannelTypes.hpp>
```

## Classes

- class [stdair::PosChannel](#)

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

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.426 PosChannel.hpp

```

00001 #ifndef __STDAIR_BOM_POSCHANNEL_HPP
00002 #define __STDAIR_BOM_POSCHANNEL_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STDAIR
00008 #include <stdair/bom/BomAbstract.hpp>
00009 #include <stdair/bom/PosChannelKey.hpp>
00010 #include <stdair/bom/PosChannelTypes.hpp>
00011
00012 // Forward declaration
00013 namespace stdair {
00014
00015     class PosChannel : public BomAbstract {
00016     public:
00017         template <typename BOM> friend class FacBom;
00018         template <typename BOM> friend class FacCloneBom;
00019         friend class FacBomManager;
00020
00021         // ////////// Type definitions
00022         typedef PosChannelKey Key_T;
00023
00024     public:
00025         // ////////// Display support methods //////////
00026         void toStream (std::ostream& ioOut) const {
00027             ioOut << toString();
00028         }
00029
00030         void fromStream (std::istream& ioIn) {
00031         }
00032
00033         std::string toString() const;
00034
00035         const std::string describeKey() const {
00036             return _key.toString();
00037         }
00038
00039     public:
00040         // ////////// Getters //////////
00041         const Key_T& getKey() const {
00042             return _key;
00043         }
00044
00045         BomAbstract* const getParent() const {
00046             return _parent;
00047         }
00048
00049         const stdair::HolderMap_T& getHolderMap() const {
00050             return _holderMap;
00051         }
00052
00053         const CityCode_T& getPos() const {
00054             return _key.getPos();
00055         }
00056
00057         const ChannelLabel_T& getChannel() const {
00058             return _key.getChannel();
00059         }
00060
00061     protected:
00062         // ////////// Constructors and destructors //////////
00063         PosChannel (const Key_T&);
00064
00065         virtual ~PosChannel();
00066
00067     private:
00068         PosChannel ();
00069
00070         PosChannel (const PosChannel&);
00071
00072     protected:
00073         // ////////// Attributes //////////
00074         Key_T _key;
00075
00076         BomAbstract* _parent;
00077

```

```

00133
00137     HolderMap_T _holderMap;
00138
00139 };
00140
00141 }
00142 #endif // __STDAIR_BOM_POSCHANNEL_HPP
00143

```

### 33.427 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

- [stdair](#)

*Handle on the StdAir library context.*

### 33.428 PosChannelKey.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <ostream>
00006 #include <sstream>
00007 // STDAIR
00008 #include <stdair/basic/BasConst_BomDisplay.hpp>
00009 #include <stdair/basic/BasConst_Request.hpp>
00010 #include <stdair/bom/PosChannelKey.hpp>
00011
00012 namespace stdair {
00013
00014     // //////////////////////////////////////
00015     PosChannelKey::PosChannelKey()
00016         : _pos (DEFAULT_POS),
00017           _channel (DEFAULT_CHANNEL) {
00018         assert (false);
00019     }
00020
00021     // //////////////////////////////////////
00022     PosChannelKey::PosChannelKey (const CityCode_T& iPos,
00023                                   const ChannelLabel_T& iChannel)
00024         : _pos (iPos), _channel (iChannel) {
00025     }
00026
00027     // //////////////////////////////////////
00028     PosChannelKey::PosChannelKey (const PosChannelKey& iKey)
00029         : _pos (iKey._pos), _channel (iKey._channel) {
00030     }
00031
00032     // //////////////////////////////////////
00033     PosChannelKey::~PosChannelKey () {
00034     }
00035
00036     // //////////////////////////////////////
00037     void PosChannelKey::toStream (std::ostream& ioOut) const {
00038         ioOut << "PosChannelKey: " << toString() << std::endl;
00039     }
00040
00041     // //////////////////////////////////////
00042     void PosChannelKey::fromStream (std::istream& ioIn) {
00043     }
00044
00045     // //////////////////////////////////////
00046     const std::string PosChannelKey::toString() const {
00047         std::ostringstream oStr;
00048         oStr << _pos << DEFAULT_KEY_SUB_FLD_DELIMITER
00049             << " " << _channel;

```



```

00050     return oStr.str();
00051 }
00052
00053 }

```

### 33.429 stdair/bom/PosChannelKey.hpp File Reference

```

#include <stdair/bom/KeyAbstract.hpp>
#include <stdair/stdair_types.hpp>

```

#### Classes

- struct [stdair::PosChannelKey](#)  
*Key of point of sale and channel.*

#### Namespaces

- [stdair](#)  
*Handle on the StdAir library context.*

### 33.430 PosChannelKey.hpp

```

00001 #ifndef __STDAIR_BOM_POSCHANNELKEY_HPP
00002 #define __STDAIR_BOM_POSCHANNELKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // SIMFQT
00008 #include <stdair/bom/KeyAbstract.hpp>
00009 #include <stdair/stdair_types.hpp>
00010
00011 namespace stdair {
00015     struct PosChannelKey : public KeyAbstract {
00016
00017     public:
00018         // ////////////////////////////////// Construction //////////////////////////////////
00022         PosChannelKey (const stdair::CityCode_T&, const
stdair::ChannelLabel_T&);
00026         PosChannelKey (const PosChannelKey&);
00030         ~PosChannelKey ();
00031     private:
00035         PosChannelKey ();
00036
00037     public:
00038         // ////////////////////////////////// Getters //////////////////////////////////
00039
00043         const stdair::CityCode_T& getPos() const {
00044             return _pos;
00045         }
00046
00050         const stdair::ChannelLabel_T& getChannel() const {
00051             return _channel;
00052         }
00053
00054     public:
00055         // ////////////////////////////////// Display support methods //////////////////////////////////
00060         void toStream (std::ostream& ioOut) const;
00061
00066         void fromStream (std::istream& ioIn);
00067
00072         const std::string toString() const;
00073
00074     private:
00075         // ////////////////////////////////// Attributes //////////////////////////////////
00079         CityCode_T _pos;
00080
00085         ChannelLabel_T _channel;
00086
00087     };
00088

```

```
00089 }
00090 #endif // __STDAIR_BOM_POSCHANNELKEY_HPP
```

### 33.431 stdair/bom/PosChannelTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

#### Namespaces

- [stdair](#)  
*Handle on the StdAir library context.*

#### Typedefs

- typedef std::list< PosChannel \* > [stdair::PosChannelList\\_T](#)
- typedef std::map< const MapKey\_T, PosChannel \* > [stdair::PosChannelMap\\_T](#)
- typedef std::pair< MapKey\_T, PosChannel \* > [stdair::PosChannelWithKey\\_T](#)
- typedef std::list< PosChannelWithKey\_T > [stdair::PosChannelDetailedList\\_T](#)

### 33.432 PosChannelTypes.hpp

```
00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_POSCHANNELTYPES_HPP
00003 #define __STDAIR_BOM_POSCHANNELTYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // STDAIR
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016     // Forward declarations.
00017     class PosChannel;
00018
00020     typedef std::list<PosChannel*> PosChannelList_T;
00021
00023     typedef std::map<const MapKey_T, PosChannel*> PosChannelMap_T;
00024
00026     typedef std::pair<MapKey_T, PosChannel*> PosChannelWithKey_T;
00027     typedef std::list<PosChannelWithKey_T> PosChannelDetailedList_T;
00028 }
00029 #endif // __STDAIR_BOM_POSCHANNELTYPES_HPP
```

### 33.433 stdair/bom/RMEventStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/bom/RMEventStruct.hpp>
```

#### Namespaces

- [stdair](#)  
*Handle on the StdAir library context.*

### 33.434 RMEventStruct.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/bom/RMEventStruct.hpp>
00009
00010 namespace stdair {
00011
00012 // //////////////////////////////////////
00013 RMEventStruct::RMEventStruct() {
00014     assert (false);
00015 }
00016
00017 // //////////////////////////////////////
00018 RMEventStruct::
00019 RMEventStruct (const RMEventStruct& iRMEvent)
00020     : _airlineCode (iRMEvent._airlineCode),
00021       _flightDateDescription (iRMEvent._flightDateDescription),
00022       _RMEventTime (iRMEvent._RMEventTime) {
00023 }
00024
00025 // //////////////////////////////////////
00026 RMEventStruct::
00027 RMEventStruct (const AirlineCode_T& iAirlineCode,
00028               const KeyDescription_T& iFlightDateDescription,
00029               const DateTime_T& iRMEventTime)
00030     : _airlineCode (iAirlineCode),
00031       _flightDateDescription (iFlightDateDescription),
00032       _RMEventTime (iRMEventTime) {
00033 }
00034
00035 // //////////////////////////////////////
00036 RMEventStruct::~RMEventStruct () {
00037 }
00038
00039 // //////////////////////////////////////
00040 void RMEventStruct::toStream (std::ostream& ioOut) const {
00041     ioOut << describe();
00042 }
00043
00044 // //////////////////////////////////////
00045 void RMEventStruct::fromStream (std::istream& ioIn) {
00046 }
00047
00048 // //////////////////////////////////////
00049 const std::string RMEventStruct::describe() const {
00050     std::ostringstream oStr;
00051     oStr << _airlineCode << ", " << _flightDateDescription << ", "
00052         << _RMEventTime;
00053     return oStr.str();
00054 }
00055
00056 }

```

### 33.435 stdair/bom/RMEventStruct.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/RMEventTypes.hpp>

```

#### Classes

- struct [stdair::RMEventStruct](#)

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.436 RMEventStruct.hpp

```

00001 #ifndef __STDAIR_BOM_RMEVENTSTRUCT_HPP
00002 #define __STDAIR_BOM_RMEVENTSTRUCT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/stdair_demand_types.hpp>
00013 #include <stdair/basic/StructAbstract.hpp>
00014 #include <stdair/bom/RMEventTypes.hpp>
00015
00016 namespace stdair {
00017
00018     struct RMEventStruct : public StructAbstract {
00019     public:
00020         // ////////////////////////////////////// Getters //////////////////////////////////////
00021         const AirlineCode_T& getAirlineCode() const {
00022             return _airlineCode;
00023         }
00024
00025         const KeyDescription_T& getFlightDateDescription() const {
00026             return _flightDateDescription;
00027         }
00028
00029         const DateTime_T& getRMEventTime() const {
00030             return _RMEventTime;
00031         }
00032
00033         // ////////////////////////////////////// Display support method //////////////////////////////////////
00034         void toStream (std::ostream& ioOut) const;
00035
00036         void fromStream (std::istream& ioIn);
00037
00038         const std::string describe() const;
00039
00040         // ////////////////////////////////////// Constructors and Destructors //////////////////////////////////////
00041     public:
00042         RMEventStruct (const AirlineCode_T&, const
00043             KeyDescription_T&, const DateTime_T&);
00044
00045         RMEventStruct (const RMEventStruct&);
00046
00047         RMEventStruct ();
00048
00049     public:
00050         ~RMEventStruct();
00051
00052     private:
00053         // ////////////////////////////////////// Attributes //////////////////////////////////////
00054         const AirlineCode_T _airlineCode;
00055
00056         const KeyDescription_T _flightDateDescription;
00057
00058         const DateTime_T _RMEventTime;
00059     };
00060 }
00061 #endif // __STDAIR_BOM_RMEVENTSTRUCT_HPP

```

## 33.437 stdair/bom/RMEventTypes.hpp File Reference

```

#include <list>
#include <boost/shared_ptr.hpp>

```

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## Typedefs

- typedef boost::shared\_ptr< RMEventStruct > [stdair::RMEventPtr\\_T](#)
- typedef std::list< RMEventStruct > [stdair::RMEventList\\_T](#)

### 33.438 RMEventTypes.hpp

```
00001 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00002 #ifndef __STDAIR_BOM_RMEVENTTYPES_HPP
00003 #define __STDAIR_BOM_RMEVENTTYPES_HPP
00004
00005 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00006 // Import section
00007 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00008 // STL
00009 #include <list>
00010 // Boost
00011 #include <boost/shared_ptr.hpp>
00012
00013 namespace stdair {
00014
00015     // Forward declarations
00016     struct RMEventStruct;
00017
00018     ////////////////////////////////////////////////////////////////// Type definitions //////////////////////////////////////////////////////////////////
00019     typedef boost::shared_ptr<RMEventStruct> RMEventPtr_T;
00020
00021     typedef std::list<RMEventStruct> RMEventList_T;
00022
00023 }
00024
00025 #endif // __STDAIR_BOM_RMEVENTTYPES_HPP
00026
00027
```

### 33.439 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

- [stdair](#)

*Handle on the StdAir library context.*

### 33.440 SegmentCabin.cpp

```
00001 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00002 // Import section
```

```

00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_BookingClass.hpp>
00009 #include <stdair/basic/BasConst_Inventory.hpp>
00010 #include <stdair/basic/BasConst_Yield.hpp>
00011 #include <stdair/basic/BasConst_BomDisplay.hpp>
00012 #include <stdair/bom/BomManager.hpp>
00013 #include <stdair/bom/SegmentDate.hpp>
00014 #include <stdair/bom/SegmentCabin.hpp>
00015 #include <stdair/bom/BookingClass.hpp>
00016 #include <stdair/bom/BookingClassTypes.hpp>
00017 #include <stdair/bom/Policy.hpp>
00018
00019 namespace stdair {
00020
00021 // //////////////////////////////////////
00022 SegmentCabin::SegmentCabin() : _key (DEFAULT_CABIN_CODE), _parent (NULL) {
00023     assert (false);
00024 }
00025
00026 // //////////////////////////////////////
00027 SegmentCabin::SegmentCabin (const SegmentCabin& iSegmentCabin)
00028 : _key (iSegmentCabin._key), _parent (NULL),
00029   _capacity (iSegmentCabin._capacity),
00030   _blockSpace (iSegmentCabin._blockSpace),
00031   _bookingCounter (iSegmentCabin._bookingCounter),
00032   _committedSpace (iSegmentCabin._committedSpace),
00033   _availabilityPool (iSegmentCabin._availabilityPool),
00034   _currentBidPrice (iSegmentCabin._currentBidPrice),
00035   _fareFamilyActivation (iSegmentCabin._fareFamilyActivation) {
00036 }
00037
00038 // //////////////////////////////////////
00039 SegmentCabin::SegmentCabin (const Key_T& iKey)
00040 : _key (iKey), _parent (NULL),
00041   _capacity (DEFAULT_CABIN_CAPACITY),
00042   _blockSpace (DEFAULT_BLOCK_SPACE),
00043   _bookingCounter (DEFAULT_CLASS_NB_OF_BOOKINGS),
00044   _committedSpace (DEFAULT_COMMITTED_SPACE),
00045   _availabilityPool (DEFAULT_AVAILABILITY),
00046   _bidPriceVector (DEFAULT_BID_PRICE_VECTOR),
00047   _currentBidPrice (DEFAULT_BID_PRICE),
00048   _fareFamilyActivation (false) {
00049 }
00050
00051 // //////////////////////////////////////
00052 SegmentCabin::~SegmentCabin() {
00053 }
00054
00055 // //////////////////////////////////////
00056 const MapKey_T SegmentCabin::getFullerKey() const {
00057     const SegmentDate& lSegmentDate = BomManager::getParent<SegmentDate>(*this);
00058     const MapKey_T oFullKey =
00059         lSegmentDate.describeKey() + DEFAULT_KEY_FLD_DELIMITER +
00060         getCabinCode();
00061     return oFullKey;
00062 }
00063
00064 // //////////////////////////////////////
00065 std::string SegmentCabin::toString() const {
00066     std::ostringstream oStr;
00067     oStr << describeKey();
00068     return oStr.str();
00069 }
00070
00071 // //////////////////////////////////////
00072 const std::string SegmentCabin::describeConvexHull() const {
00073     std::ostringstream oStr;
00074     for (PolicyList_T::const_iterator itP = _convexHull.begin();
00075          itP != _convexHull.end(); ++itP) {
00076         const Policy* lPolicy = *itP;
00077         assert (lPolicy != NULL);
00078         oStr << lPolicy->toString();
00079     }
00080     return oStr.str();
00081 }
00082
00083 // //////////////////////////////////////
00084 void SegmentCabin::
00085 updateFromReservation (const NbOfBookings_T& iNbOfBookings) {
00086     _committedSpace += iNbOfBookings;
00087 }
00088

```

```

00089 // //////////////////////////////////////
00090 void SegmentCabin::addPolicy (Policy& ioPolicy) {
00091     _convexHull.push_back (&ioPolicy);
00092 }
00093 }
00094

```

### 33.441 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>

```

#### Classes

- class [stdair::SegmentCabin](#)

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

#### Namespaces

- [boost](#)

*Forward declarations.*

- [boost::serialization](#)
- [stdair](#)

*Handle on the StdAir library context.*

### 33.442 SegmentCabin.hpp

```

00001 #ifndef __STDAIR_BOM_SEGMENTCABIN_HPP
00002 #define __STDAIR_BOM_SEGMENTCABIN_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/bom/BomAbstract.hpp>
00013 #include <stdair/bom/SegmentCabinKey.hpp>
00014 #include <stdair/bom/SegmentCabinTypes.hpp>
00015 #include <stdair/bom/PolicyTypes.hpp>
00016
00017 namespace boost {
00018     namespace serialization {
00019         class access;
00020     }
00021 }
00022
00023 namespace stdair {
00024     // Forward declarations
00025     class SegmentSnapshotTable;
00026     class Policy;
00027
00028     class SegmentCabin : public BomAbstract {
00029     public:
00030         template <typename BOM> friend class FacBom;
00031         template <typename BOM> friend class FacCloneBom;
00032         friend class FacBomManager;
00033         friend class boost::serialization::access;
00034     public:
00035         // ////////////////////////////////// Type definitions //////////////////////////////////

```

```

00044     typedef SegmentCabinKey Key_T;
00045
00046
00047 public:
00048     // //////////// Getters ////////////
00052     const Key_T& getKey() const {
00053         return _key;
00054     }
00055
00059     BomAbstract* const getParent() const {
00060         return _parent;
00061     }
00062
00066     const HolderMap_T& getHolderMap() const {
00067         return _holderMap;
00068     }
00069
00073     const CabinCode_T& getCabinCode() const {
00074         return _key.getCabinCode();
00075     }
00076
00085     const MapKey_T getFullerKey() const;
00086
00088     const SegmentSnapshotTable& getSegmentSnapshotTable() const
00089 {
00089     assert (_segmentSnapshotTable != NULL);
00090     return *_segmentSnapshotTable;
00091 }
00092
00094     const CabinCapacity_T& getCapacity() const {
00095         return _capacity;
00096     }
00097
00099     const BlockSpace_T& getBlockSpace() const {
00100         return _blockSpace;
00101     }
00102
00104     const BlockSpace_T& getMIN() const {
00105         return _min;
00106     }
00107
00109     const UPR_T& getUPR() const {
00110         return _upr;
00111     }
00112
00114     const NbOfBookings_T& getBookingCounter() const {
00115         return _bookingCounter;
00116     }
00117
00119     const CommittedSpace_T& getCommittedSpace() const {
00120         return _committedSpace;
00121     }
00122
00124     const Availability_T& getAvailabilityPool() const {
00125         return _availabilityPool;
00126     }
00127
00129     const BidPrice_T& getCurrentBidPrice() const {
00130         return _currentBidPrice;
00131     }
00132
00134     const BidPriceVector_T& getBidPriceVector() const {
00135         return _bidPriceVector;
00136     }
00137
00139     const bool getFareFamilyStatus() const {
00140         return _fareFamilyActivation;
00141     }
00142
00144     const PolicyList_T& getConvexHull() const {
00145         return _convexHull;
00146     }
00147
00148 public:
00149     // //////////// Setters ////////////
00151     void setSegmentSnapshotTable (SegmentSnapshotTable& ioTable)
00152 {
00152     _segmentSnapshotTable = &ioTable;
00153 }
00154
00156     void setCapacity (const CabinCapacity_T& iCapacity) {
00157         _capacity = iCapacity;
00158     }
00159
00161     void setBlockSpace (const BlockSpace_T& iBlockSpace) {
00162         _blockSpace = iBlockSpace;
00163     }

```



```

00164
00166 void setMIN (const BlockSpace_T& iMIN) {
00167     _min = iMIN;
00168 }
00169
00171 void setUPR (const UPR_T& iUPR) {
00172     _upr = iUPR;
00173 }
00174
00176 void setBookingCounter (const NbOfBookings_T& iBookingCounter) {
00177     _bookingCounter = iBookingCounter;
00178 }
00179
00181 void setCommittedSpace (const CommittedSpace_T& iCommittedSpace) {
00182     _committedSpace = iCommittedSpace;
00183 }
00184
00186 void setAvailabilityPool (const Availability_T& iAvailabilityPool) {
00187     _availabilityPool = iAvailabilityPool;
00188 }
00189
00191 void setBidPriceVector (const BidPriceVector_T& iBPV) {
00192     _bidPriceVector = iBPV;
00193 }
00194
00196 void activateFareFamily () {
00197     _fareFamilyActivation = true;
00198 }
00199
00200 public:
00201     // //////////// Business methods ////////////
00203     void updateFromReservation (const NbOfBookings_T&);
00204
00206     void resetConvexHull () { _convexHull.clear(); }
00207
00212     void addPolicy (Policy&);
00213
00214 public:
00215     // //////////// Display support methods ////////////
00221     void toStream (std::ostream& ioOut) const {
00222         ioOut << toString();
00223     }
00224
00230     void fromStream (std::istream& ioIn) {
00231     }
00232
00236     std::string toString() const;
00237
00241     const std::string describeKey() const {
00242         return _key.toString();
00243     }
00244
00248     const std::string describeConvexHull() const;
00249
00250
00251 public:
00252     // //////////// (Boost) Serialisation support methods ////////////
00256     template<class Archive>
00257     void serialize (Archive& ar, const unsigned int iFileVersion);
00258
00259 private:
00267     void serialisationImplementationExport() const;
00268     void serialisationImplementationImport();
00269
00270
00271 protected:
00272     // //////////// Constructors and destructors ////////////
00276     SegmentCabin (const Key_T&);
00277
00281     virtual ~SegmentCabin();
00282
00283 private:
00287     SegmentCabin();
00288
00292     SegmentCabin (const SegmentCabin&);
00293
00294
00295 protected:
00296     // //////////// Attributes ////////////
00300     Key_T _key;
00301
00305     BomAbstract* _parent;
00306
00310     HolderMap_T _holderMap;
00311
00315     SegmentSnapshotTable* _segmentSnapshotTable;
00316

```

```

00318     CabinCapacity_T _capacity;
00319
00321     BlockSpace_T _blockSpace;
00322
00324     BlockSpace_T _min;
00325
00327     UPR_T _upr;
00328
00330     NbOfBookings_T _bookingCounter;
00331
00333     CommittedSpace_T _committedSpace;
00334
00336     Availability_T _availabilityPool;
00337
00339     BidPriceVector_T _bidPriceVector;
00340
00342     BidPrice_T _currentBidPrice;
00343
00345     bool _fareFamilyActivation;
00346
00348     PolicyList_T _convexHull;
00349
00350 };
00351
00352 }
00353 #endif // __STDAIR_BOM_SEGMENTCABIN_HPP
00354

```

### 33.443 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

- [stdair](#)

*Handle on the StdAir library context.*

#### Functions

- template void [stdair::SegmentCabinKey::serialize< ba::text\\_oarchive >](#) (ba::text\_oarchive &, unsigned int)
- template void [stdair::SegmentCabinKey::serialize< ba::text\\_iarchive >](#) (ba::text\_iarchive &, unsigned int)

### 33.444 SegmentCabinKey.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/access.hpp>
00011 // StdAir
00012 #include <stdair/basic/BasConst_Inventory.hpp>
00013 #include <stdair/bom/SegmentCabinKey.hpp>
00014
00015 namespace stdair {
00016
00017 // //////////////////////////////////////
00018 SegmentCabinKey::SegmentCabinKey() : _cabinCode (DEFAULT_CABIN_CODE) {
00019     assert (false);

```

```

00020     }
00021
00022     // //////////////////////////////////////
00023 SegmentCabinKey::SegmentCabinKey (const CabinCode_T& iCabinCode)
00024     : _cabinCode (iCabinCode) {
00025     }
00026
00027     // //////////////////////////////////////
00028 SegmentCabinKey::SegmentCabinKey (const SegmentCabinKey& iKey)
00029     : _cabinCode (iKey._cabinCode) {
00030     }
00031
00032     // //////////////////////////////////////
00033 SegmentCabinKey::~SegmentCabinKey () {
00034     }
00035
00036     // //////////////////////////////////////
00037 void SegmentCabinKey::toStream (std::ostream& ioOut) const {
00038     ioOut << "SegmentCabinKey: " << toString();
00039 }
00040
00041     // //////////////////////////////////////
00042 void SegmentCabinKey::fromStream (std::istream& ioIn) {
00043 }
00044
00045     // //////////////////////////////////////
00046 const std::string SegmentCabinKey::toString() const {
00047     std::ostringstream oStr;
00048     oStr << _cabinCode;
00049     return oStr.str();
00050 }
00051
00052     // //////////////////////////////////////
00053 void SegmentCabinKey::serialisationImplementationExport() const {
00054     std::ostringstream oStr;
00055     boost::archive::text_oarchive oa (oStr);
00056     oa << *this;
00057 }
00058
00059     // //////////////////////////////////////
00060 void SegmentCabinKey::serialisationImplementationImport() {
00061     std::istringstream iStr;
00062     boost::archive::text_iarchive ia (iStr);
00063     ia >> *this;
00064 }
00065
00066     // //////////////////////////////////////
00067 template<class Archive>
00068 void SegmentCabinKey::serialize (Archive& ioArchive,
00069                                 const unsigned int iFileVersion) {
00070     ioArchive & _cabinCode;
00071 }
00072
00073     // //////////////////////////////////////
00074 // Explicit template instantiation
00075 namespace ba = boost::archive;
00076 template void SegmentCabinKey::
00077 serialize<ba::text_oarchive> (ba::text_oarchive&, unsigned int);
00078 template void SegmentCabinKey::
00079 serialize<ba::text_iarchive> (ba::text_iarchive&, unsigned int);
00080 // //////////////////////////////////////
00081 }
00082

```

### 33.445 stdair/bom/SegmentCabinKey.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>

```

#### Classes

- struct [stdair::SegmentCabinKey](#)

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

## Namespaces

- [boost](#)

*Forward declarations.*

- [boost::serialization](#)
- [stdair](#)

*Handle on the StdAir library context.*

## 33.446 SegmentCabinKey.hpp

```

00001 #ifndef __STDAIR_BOM_SEGMENTCABINKEY_HPP
00002 #define __STDAIR_BOM_SEGMENTCABINKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_basic_types.hpp>
00012 #include <stdair/bom/KeyAbstract.hpp>
00013
00015 namespace boost {
00016     namespace serialization {
00017         class access;
00018     }
00019 }
00020
00021 namespace stdair {
00022
00026     struct SegmentCabinKey : public KeyAbstract {
00027         friend class boost::serialization::access;
00028
00029         // ////////// Constructors and destructors //////////
00030     private:
00031         SegmentCabinKey();
00032
00033     public:
00034         SegmentCabinKey (const CabinCode_T& iCabinCode);
00035
00036         SegmentCabinKey (const SegmentCabinKey&);
00037
00038         ~SegmentCabinKey();
00039
00040     public:
00041         // ////////// Getters //////////
00042         const CabinCode_T& getCabinCode() const {
00043             return _cabinCode;
00044         }
00045
00046     public:
00047         // ////////// Display support methods //////////
00048         void toStream (std::ostream& ioOut) const;
00049
00050         void fromStream (std::istream& ioIn);
00051
00052         const std::string toString() const;
00053
00054     public:
00055         // ////////// (Boost) Serialisation support methods //////////
00056         template<class Archive>
00057         void serialize (Archive& ar, const unsigned int iFileVersion);
00058
00059     private:
00060         void serialisationImplementationExport() const;
00061         void serialisationImplementationImport();
00062
00063     private:
00064         // ////////// Attributes //////////
00065         CabinCode_T _cabinCode;
00066     };
00067 }
00068
00069 #endif // __STDAIR_BOM_SEGMENTCABINKEY_HPP

```

### 33.447 stdair/bom/SegmentCabinTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

#### Typedefs

- typedef std::list< SegmentCabin \* > [stdair::SegmentCabinList\\_T](#)
- typedef std::map< const MapKey\_T, SegmentCabin \* > [stdair::SegmentCabinMap\\_T](#)

### 33.448 SegmentCabinTypes.hpp

```
00001 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00002 #ifndef __STDAIR_BOM_SEGMENTCABINTYPES_HPP
00003 #define __STDAIR_BOM_SEGMENTCABINTYPES_HPP
00004
00005 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00006 // Import section
00007 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // StdAir
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016     // Forward declarations.
00017     class SegmentCabin;
00018
00019     typedef std::list<SegmentCabin*> SegmentCabinList_T;
00020
00021     typedef std::map<const MapKey_T, SegmentCabin*> SegmentCabinMap_T;
00022
00023 }
00024
00025 #endif // __STDAIR_BOM_SEGMENTCABINTYPES_HPP
00026
00027
```

### 33.449 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

- [stdair](#)

*Handle on the StdAir library context.*

## 33.450 SegmentDate.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_BookingClass.hpp>
00009 #include <stdair/basic/BasConst_Inventory.hpp>
00010 #include <stdair/bom/BomManager.hpp>
00011 #include <stdair/bom/SegmentDate.hpp>
00012 #include <stdair/bom/SegmentCabin.hpp>
00013
00014 namespace stdair {
00015
00016 // //////////////////////////////////////
00017 SegmentDate::SegmentDate()
00018 : _key (DEFAULT_ORIGIN, DEFAULT_DESTINATION), _parent (NULL),
00019   _operatingSegmentDate (NULL) {
00020   assert (false);
00021 }
00022
00023 // //////////////////////////////////////
00024 SegmentDate::SegmentDate (const SegmentDate& iSegmentDate)
00025 : _key (iSegmentDate._key),
00026   _parent (NULL),
00027   _operatingSegmentDate (NULL),
00028   _boardingDate (iSegmentDate._boardingDate),
00029   _boardingTime (iSegmentDate._boardingTime),
00030   _offDate (iSegmentDate._offDate),
00031   _offTime (iSegmentDate._offTime),
00032   _elapsedTime (iSegmentDate._elapsedTime),
00033   _distance (iSegmentDate._distance),
00034   _routingLegKeyList (iSegmentDate._routingLegKeyList) {
00035 }
00036
00037 // //////////////////////////////////////
00038 SegmentDate::SegmentDate (const Key_T& iKey)
00039 : _key (iKey), _parent (NULL),
00040   _operatingSegmentDate (NULL) {
00041 }
00042
00043 // //////////////////////////////////////
00044 SegmentDate::~SegmentDate() {
00045 }
00046
00047 // //////////////////////////////////////
00048 std::string SegmentDate::toString() const {
00049   std::ostringstream oStr;
00050   oStr << describeKey();
00051   return oStr.str();
00052 }
00053
00054 // //////////////////////////////////////
00055 const Duration_T SegmentDate::getTimeOffset() const {
00056   // TimeOffset = (OffTime - BoardingTime) + (OffDate - BoardingDate) * 24
00057   //              - ElapsedTime
00058   Duration_T oTimeOffset = (_offTime - _boardingTime);
00059   const DateOffset_T& lDateOffset = getDateOffset();
00060   const Duration_T lDateOffsetInHours (lDateOffset.days() * 24, 0, 0);
00061   oTimeOffset += lDateOffsetInHours - _elapsedTime;
00062   return oTimeOffset;
00063 }
00064 }
00065

```

## 33.451 stdair/bom/SegmentDate.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/SegmentDateKey.hpp>
#include <stdair/bom/SegmentDateTypes.hpp>

```

## Classes

- class [stdair::SegmentDate](#)

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

## Namespaces

- [boost](#)

*Forward declarations.*

- [boost::serialization](#)
- [stdair](#)

*Handle on the StdAir library context.*

## Typedefs

- typedef std::list< std::string > [stdair::RoutingLegKeyList\\_T](#)

## 33.452 SegmentDate.hpp

```

00001 #ifndef __STDAIR_BOM_SEGMENTDATE_HPP
00002 #define __STDAIR_BOM_SEGMENTDATE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/bom/BomAbstract.hpp>
00013 #include <stdair/bom/SegmentDateKey.hpp>
00014 #include <stdair/bom/SegmentDateTypes.hpp>
00015
00017 namespace boost {
00018     namespace serialization {
00019         class access;
00020     }
00021 }
00022
00023 namespace stdair {
00024
00026     struct SegmentCabinKey;
00027     class SegmentCabin;
00028
00029     // Define the routing leg keys list type.
00030     typedef std::list<std::string> RoutingLegKeyList_T;
00031
00036     class SegmentDate : public BomAbstract {
00037     public:
00038         template <typename BOM> friend class FacBom;
00039         template <typename BOM> friend class FacCloneBom;
00040         friend class FacBomManager;
00041         friend class boost::serialization::access;
00042
00043         // ////////////////////////////////// Type definitions //////////////////////////////////
00044         typedef SegmentDateKey Key_T;
00045
00046     public:
00047         // ////////////////////////////////// Getters //////////////////////////////////
00048         const Key_T& getKey() const {
00049             return _key;
00050         }
00051
00052         BomAbstract* const getParent() const {
00053             return _parent;
00054         }
00055
00056         const AirportCode_T& getBoardingPoint() const {
00057             return _key.getBoardingPoint();
00058         }
00059
00060         const AirportCode_T& getOffPoint() const {

```

```

00077     return _key.getOffPoint();
00078 }
00079
00083 const HolderMap_T& getHolderMap() const {
00084     return _holderMap;
00085 }
00086
00090 const Date_T& getBoardingDate() const {
00091     return _boardingDate;
00092 }
00093
00097 const Duration_T& getBoardingTime() const {
00098     return _boardingTime;
00099 }
00100
00104 const Date_T& getOffDate() const {
00105     return _offDate;
00106 }
00107
00111 const Duration_T& getOffTime() const {
00112     return _offTime;
00113 }
00114
00118 const Duration_T& getElapsedTime() const {
00119     return _elapsedTime;
00120 }
00121
00125 const Distance_T& getDistance() const {
00126     return _distance;
00127 }
00128
00132 const DateOffset_T getDateOffset() const {
00133     return _offDate - _boardingDate;
00134 }
00135
00144 const Duration_T getTimeOffset() const;
00145
00149 SegmentDate* getOperatingSegmentDate () const {
00150     return _operatingSegmentDate;
00151 }
00152
00156 const SegmentDateList_T& getMarketingSegmentDateList ()
const {
00157     return _marketingSegmentDateList;
00158 }
00159
00163 const RoutingLegKeyList_T& getLegKeyList () const {
00164     return _routingLegKeyList;
00165 }
00166
00167 public:
00168     // ////////// Setters //////////
00172 void setBoardingDate (const Date_T& iBoardingDate) {
00173     _boardingDate = iBoardingDate;
00174 }
00175
00179 void setBoardingTime (const Duration_T& iBoardingTime) {
00180     _boardingTime = iBoardingTime;
00181 }
00182
00186 void setOffDate (const Date_T& iOffDate) {
00187     _offDate = iOffDate;
00188 }
00189
00193 void setOffTime (const Duration_T& iOffTime) {
00194     _offTime = iOffTime;
00195 }
00196
00200 void setElapsedTime (const Duration_T& iElapsedTime) {
00201     _elapsedTime = iElapsedTime;
00202 }
00203
00207 void setDistance (const Distance_T& iDistance) {
00208     _distance = iDistance;
00209 }
00210
00214 void addLegKey (const std::string& iLegKey) {
00215     _routingLegKeyList.push_back(iLegKey);
00216 }
00217
00218 private:
00222 void linkWithOperating (SegmentDate& iSegmentDate) {
00223     _operatingSegmentDate = &iSegmentDate;
00224 }
00225
00226 public:
00227     // ////////// Display support methods //////////

```



```

00233     void toStream (std::ostream& ioOut) const {
00234         ioOut << toString();
00235     }
00236
00242     void fromStream (std::istream& ioIn) {
00243     }
00244
00248     std::string toString() const;
00249
00253     const std::string describeKey() const {
00254         return _key.toString();
00255     }
00256
00257
00258 public:
00259     // ////////// (Boost) Serialisation support methods //////////
00263     template<class Archive>
00264     void serialize (Archive& ar, const unsigned int iFileVersion);
00265
00266 private:
00274     void serialisationImplementationExport() const;
00275     void serialisationImplementationImport();
00276
00277
00278 protected:
00279     // ////////// Constructors and destructors //////////
00283     SegmentDate (const Key_T&);
00284
00288     virtual ~SegmentDate();
00289
00290 private:
00294     SegmentDate();
00295
00299     SegmentDate (const SegmentDate&);
00300
00301
00302 protected:
00303     // ////////// Attributes //////////
00307     Key_T _key;
00308
00312     BomAbstract* _parent;
00313
00317     HolderMap_T _holderMap;
00318
00325     SegmentDate* _operatingSegmentDate;
00326
00333     SegmentDateList_T _marketingSegmentDateList;
00334
00338     Date_T _boardingDate;
00339
00343     Duration_T _boardingTime;
00344
00348     Date_T _offDate;
00349
00353     Duration_T _offTime;
00354
00358     Duration_T _elapsedTime;
00359
00363     Distance_T _distance;
00364
00368     RoutingLegKeyList_T _routingLegKeyList;
00369 };
00370
00371 }
00372 #endif // __STDAIR_BOM_SEGMENTDATE_HPP
00373

```

### 33.453 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

- [stdair](#)

*Handle on the StdAir library context.*

## Functions

- template void [stdair::SegmentDateKey::serialize< ba::text\\_oarchive >](#) (ba::text\_oarchive &, unsigned int)
- template void [stdair::SegmentDateKey::serialize< ba::text\\_iarchive >](#) (ba::text\_iarchive &, unsigned int)

## 33.454 SegmentDateKey.cpp

```

00001 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00002 // Import section
00003 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/access.hpp>
00011 // StdAir
00012 #include <stdair/basic/BasConst_Inventory.hpp>
00013 #include <stdair/basic/BasConst_BomDisplay.hpp>
00014 #include <stdair/bom/SegmentDateKey.hpp>
00015
00016 namespace stdair {
00017
00018 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00019 SegmentDateKey::SegmentDateKey()
00020 : _boardingPoint (DEFAULT_ORIGIN), _offPoint (
00021     DEFAULT_DESTINATION) {
00022     assert (false);
00023 }
00024
00025 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00026 SegmentDateKey::SegmentDateKey (const AirportCode_T& iBoardingPoint,
00027     const AirportCode_T& iOffPoint)
00028 : _boardingPoint (iBoardingPoint), _offPoint (iOffPoint) {
00029 }
00030
00031 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00032 SegmentDateKey::SegmentDateKey (const SegmentDateKey& iKey)
00033 : _boardingPoint (iKey._boardingPoint), _offPoint (iKey._offPoint) {
00034 }
00035
00036 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00037 SegmentDateKey::~SegmentDateKey() {
00038 }
00039
00040 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00041 void SegmentDateKey::toStream (std::ostream& ioOut) const {
00042     ioOut << "SegmentDateKey: " << toString() << std::endl;
00043 }
00044
00045 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00046 void SegmentDateKey::fromStream (std::istream& ioIn) {
00047 }
00048
00049 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00050 const std::string SegmentDateKey::toString() const {
00051     std::ostringstream oStr;
00052     oStr << _boardingPoint
00053         << DEFAULT_KEY_SUB_FLD_DELIMITER << " " << _offPoint;
00054     return oStr.str();
00055 }
00056
00057 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00058 void SegmentDateKey::serialisationImplementationExport() const {
00059     std::ostringstream oStr;
00060     boost::archive::text_oarchive oa (oStr);
00061     oa << *this;
00062 }
00063
00064 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00065 void SegmentDateKey::serialisationImplementationImport() {
00066     std::istringstream iStr;
00067     boost::archive::text_iarchive ia (iStr);

```

```

00067     ia >> *this;
00068 }
00069
00070 // //////////////////////////////////////
00071 template<class Archive>
00072 void SegmentDateKey::serialize (Archive& ioArchive,
00073                                const unsigned int iFileVersion) {
00074     ioArchive & _boardingPoint & _offPoint;
00075 }
00076
00077 // //////////////////////////////////////
00078 // Explicit template instantiation
00079 namespace ba = boost::archive;
00080 template void SegmentDateKey::serialize<ba::text_oarchive> (ba::text_oarchive&,
00081                                                            unsigned int);
00082 template void SegmentDateKey::serialize<ba::text_iarchive> (ba::text_iarchive&,
00083                                                            unsigned int);
00084 // //////////////////////////////////////
00085
00086 }

```

### 33.455 stdair/bom/SegmentDateKey.hpp File Reference

```

#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>

```

#### Classes

- struct [stdair::SegmentDateKey](#)  
*Key of a given segment-date, made of an origin and a destination airports.*

#### Namespaces

- [boost](#)  
*Forward declarations.*
- [boost::serialization](#)
- [stdair](#)  
*Handle on the StdAir library context.*

### 33.456 SegmentDateKey.hpp

```

00001 #ifndef __STDAIR_BOM_SEGMENTDATEKEY_HPP
00002 #define __STDAIR_BOM_SEGMENTDATEKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009 #include <stdair/bom/KeyAbstract.hpp>
00010
00012 namespace boost {
00013     namespace serialization {
00014         class access;
00015     }
00016 }
00017
00018 namespace stdair {
00019
00024     struct SegmentDateKey : public KeyAbstract {
00025         friend class boost::serialization::access;
00026
00027         // ////////////////////////////////// Constructors and destructors //////////////////////////////////
00028     private:
00032         SegmentDateKey();
00033
00034     public:
00038         SegmentDateKey (const AirportCode_T&, const
AirportCode_T&);

```

```

00042     SegmentDateKey (const SegmentDateKey&);
00046     ~SegmentDateKey();
00047
00048
00049     // //////////// Getters ////////////
00051     const AirportCode_T& getBoardingPoint() const {
00052         return _boardingPoint;
00053     }
00054
00056     const AirportCode_T& getOffPoint() const {
00057         return _offPoint;
00058     }
00059
00060
00061     // //////////// Display support methods ////////////
00067     void toStream (std::ostream& ioOut) const;
00068
00074     void fromStream (std::istream& ioIn);
00075
00085     const std::string toString() const;
00086
00087
00088 public:
00089     // //////////// (Boost) Serialisation support methods ////////////
00093     template<class Archive>
00094     void serialize (Archive& ar, const unsigned int iFileVersion);
00095
00096 private:
00101     void serialisationImplementationExport() const;
00102     void serialisationImplementationImport();
00103
00104
00105 private:
00106     // //////////// Attributes ////////////
00110     AirportCode_T _boardingPoint;
00111
00115     AirportCode_T _offPoint;
00116 };
00117
00118 }
00119 #endif // __STDAIR_BOM_SEGMENTDATEKEY_HPP

```

### 33.457 stdair/bom/SegmentDateTypes.hpp File Reference

```

#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>

```

#### Namespaces

- [stdair](#)  
Handle on the StdAir library context.

#### Typedefs

- typedef std::list< SegmentDate \* > [stdair::SegmentDateList\\_T](#)
- typedef std::map< const MapKey\_T, SegmentDate \* > [stdair::SegmentDateMap\\_T](#)

### 33.458 SegmentDateTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_SEGMENTDATETYPES_HPP
00003 #define __STDAIR_BOM_SEGMENTDATETYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // StdAir

```

```

00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015     // Forward declarations.
00016     class SegmentDate;
00017     typedef std::list<SegmentDate*> SegmentDateList_T;
00018
00019     typedef std::map<const MapKey_T, SegmentDate*> SegmentDateMap_T;
00020 }
00021
00022 #endif // __STDAIR_BOM_SEGMENTDATETYPES_HPP
00023

```

### 33.459 stdair/bom/SegmentPeriod.cpp File Reference

```

#include <cassert>
#include <stdair/basic/BasConst_BookingClass.hpp>
#include <stdair/bom/SegmentPeriod.hpp>

```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.460 SegmentPeriod.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // STDAIR
00007 #include <stdair/basic/BasConst_BookingClass.hpp>
00008 #include <stdair/bom/SegmentPeriod.hpp>
00009
00010 namespace stdair {
00011
00012     // //////////////////////////////////////
00013     SegmentPeriod::SegmentPeriod (const Key_T& iKey)
00014         : _key (iKey), _parent (NULL), _boardingDateOffset (0), _offDateOffset (0) {
00015     }
00016
00017     // //////////////////////////////////////
00018     SegmentPeriod::SegmentPeriod (const SegmentPeriod& iSegmentPeriod)
00019         : _key (iSegmentPeriod.getKey()),
00020           _parent (NULL),
00021           _boardingTime (iSegmentPeriod._boardingTime),
00022           _offTime (iSegmentPeriod._offTime),
00023           _boardingDateOffset (iSegmentPeriod._boardingDateOffset),
00024           _offDateOffset (iSegmentPeriod._offDateOffset),
00025           _elapsedTime (iSegmentPeriod._elapsedTime) {
00026     }
00027
00028     // //////////////////////////////////////
00029     SegmentPeriod::~SegmentPeriod () {
00030     }
00031
00032     // //////////////////////////////////////
00033     std::string SegmentPeriod::toString() const {
00034         std::ostringstream oStr;
00035         oStr << describeKey();
00036         return oStr.str();
00037     }
00038
00039     // //////////////////////////////////////
00040     void SegmentPeriod::
00041     addCabinBookingClassList (const CabinCode_T& iCabinCode,
00042                             const ClassList_String_T& iClassCodeList) {
00043         const bool insert = _cabinBookingClassMap.
00044             insert (CabinBookingClassMap_T::value_type (iCabinCode,
00045                                                         iClassCodeList)).second;
00046         assert (insert == true);
00047     }
00048

```

```

00047     }
00048
00049 }
```

### 33.461 stdair/bom/SegmentPeriod.hpp File Reference

```

#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/SegmentPeriodKey.hpp>
#include <stdair/bom/SegmentPeriodTypes.hpp>
```

#### Classes

- class [stdair::SegmentPeriod](#)

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.462 SegmentPeriod.hpp

```

00001 #ifndef __STDAIR_BOM_SEGMENTPERIOD_HPP
00002 #define __STDAIR_BOM_SEGMENTPERIOD_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STDAIR
00008 #include <stdair/bom/BomAbstract.hpp>
00009 #include <stdair/bom/SegmentPeriodKey.hpp>
00010 #include <stdair/bom/SegmentPeriodTypes.hpp>
00011
00012 namespace stdair {
00013
00014     class SegmentPeriod : public BomAbstract {
00015     public:
00016         template <typename BOM> friend class FacBom;
00017         template <typename BOM> friend class FacCloneBom;
00018         friend class FacBomManager;
00019
00020     public:
00021         // Type definitions.
00022         typedef SegmentPeriodKey Key_T;
00023
00024     public:
00025         // ////////////////////////////////// Getters //////////////////////////////////
00026         const Key_T& getKey() const { return _key; }
00027
00028         BomAbstract* const getParent() const { return _parent; }
00029
00030         const AirportCode_T& getBoardingPoint () const {
00031             return _key.getBoardingPoint();
00032         }
00033
00034         const AirportCode_T& getOffPoint () const { return
00035             _key.getOffPoint(); }
00036
00037         const Duration_T& getBoardingTime () const { return
00038             _boardingTime; }
00039
00040         const Duration_T& getOffTime () const { return _offTime; }
00041
00042         const DateOffset_T& getBoardingDateOffset () const {
00043             return _boardingDateOffset;
00044         }
00045
00046         const DateOffset_T& getOffDateOffset () const { return
00047             _offDateOffset; }
00048
00049         const Duration_T& getElapsedTime() const { return
00050             _elapsedTime; }
00051
00052 }
```

```

00059     const CabinBookingClassMap_T& getCabinBookingClassMap ()
00060     const {
00061         return _cabinBookingClassMap;
00062     }
00064     const HolderMap_T& getHolderMap() const { return
_holderMap; }
00065
00066 public:
00067     // ////////// Setters //////////
00069     void setBoardingTime (const Duration_T& iBoardingTime) {
00070         _boardingTime = iBoardingTime;
00071     }
00072
00074     void setOffTime (const Duration_T& iOffTime) { _offTime = iOffTime; }
00075
00077     void setBoardingDateOffset (const DateOffset_T& iDateOffset) {
00078         _boardingDateOffset = iDateOffset;
00079     }
00080
00082     void setOffDateOffset (const DateOffset_T& iDateOffset) {
00083         _offDateOffset = iDateOffset;
00084     }
00085
00087     void setElapsedTime (const Duration_T& iElapsedTime) {
00088         _elapsedTime = iElapsedTime;
00089     }
00090
00093     void addCabinBookingClassList (const CabinCode_T&,
00094                                   const ClassList_String_T&);
00095
00096 public:
00097     // ////////// Display support methods //////////
00100     void toStream (std::ostream& ioOut) const { ioOut << toString(); }
00101
00104     void fromStream (std::istream& ioIn) { }
00105
00107     std::string toString() const;
00108
00110     const std::string describeKey() const { return _key.toString(); }
00111
00112 protected:
00113     // ////////// Constructors and destructors //////////
00117     SegmentPeriod (const Key_T&);
00121     virtual ~SegmentPeriod();
00122
00123 private:
00127     SegmentPeriod();
00131     SegmentPeriod (const SegmentPeriod&);
00132
00133 protected:
00134     // Attributes
00135     Key_T _key;
00136     BomAbstract* _parent;
00137     Duration_T _boardingTime;
00138     Duration_T _offTime;
00139     DateOffset_T _boardingDateOffset;
00140     DateOffset_T _offDateOffset;
00141     Duration_T _elapsedTime;
00142     CabinBookingClassMap_T _cabinBookingClassMap;
00143     HolderMap_T _holderMap;
00144 };
00145
00146 }
00147 #endif // __STDAIR_BOM_SEGMENTPERIOD_HPP
00148

```

### 33.463 stdair/bom/SegmentPeriodKey.cpp File Reference

```

#include <sstream>
#include <stdair/bom/SegmentPeriodKey.hpp>

```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.464 SegmentPeriodKey.cpp

```

00001 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00002 // Import section
00003 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <sstream>
00006 // StdAir
00007 #include <stdair/bom/SegmentPeriodKey.hpp>
00008
00009 namespace stdair {
00010
00011 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00012 SegmentPeriodKey::SegmentPeriodKey (const AirportCode_T& iBoardingPoint,
00013                                     const AirportCode_T& iOffPoint)
00014 : _boardingPoint (iBoardingPoint), _offPoint (iOffPoint) {
00015 }
00016
00017 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00018 SegmentPeriodKey::SegmentPeriodKey (const SegmentPeriodKey& iKey)
00019 : _boardingPoint (iKey._boardingPoint), _offPoint (iKey._offPoint) {
00020 }
00021
00022 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00023 SegmentPeriodKey::~SegmentPeriodKey () {
00024 }
00025
00026 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00027 void SegmentPeriodKey::toStream (std::ostream& ioOut) const {
00028     ioOut << "SegmentPeriodKey: " << toString() << std::endl;
00029 }
00030
00031 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00032 void SegmentPeriodKey::fromStream (std::istream& ioIn) {
00033 }
00034
00035 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00036 const std::string SegmentPeriodKey::toString() const {
00037     std::ostringstream oStr;
00038     oStr << _boardingPoint << "-" << _offPoint;
00039     return oStr.str();
00040 }
00041
00042 }

```

## 33.465 stdair/bom/SegmentPeriodKey.hpp File Reference

```

#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>

```

## Classes

- struct [stdair::SegmentPeriodKey](#)

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.466 SegmentPeriodKey.hpp

```

00001 #ifndef __STDAIR_BOM_SEGMENTPERIODKEY_HPP
00002 #define __STDAIR_BOM_SEGMENTPERIODKEY_HPP
00003
00004 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00005 // Import section
00006 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009 #include <stdair/bom/KeyAbstract.hpp>
00010

```



```

00011 namespace stdair {
00012
00014     struct SegmentPeriodKey : public KeyAbstract {
00015
00016     private:
00017         // //////////// Default constructor ////////////
00018         SegmentPeriodKey () { };
00019     public:
00020         // //////////// Construction ////////////
00022         SegmentPeriodKey (const AirportCode_T&, const
AirportCode_T&);
00023         SegmentPeriodKey (const SegmentPeriodKey&);
00025         ~SegmentPeriodKey ();
00026
00027         // //////////// Getters ////////////
00029         const AirportCode_T& getBoardingPoint() const {
00030             return _boardingPoint;
00031         }
00032
00034         const AirportCode_T& getOffPoint() const {
00035             return _offPoint;
00036         }
00037
00038         // //////////// Display support methods ////////////
00041         void toStream (std::ostream& ioOut) const;
00042
00045         void fromStream (std::istream& ioIn);
00046
00052         const std::string toString() const;
00053
00054     private:
00055         // Attributes
00057         AirportCode_T _boardingPoint;
00058
00060         AirportCode_T _offPoint;
00061     };
00062
00063 }
00064 #endif // __STDAIR_BOM_SEGMENTPERIODKEY_HPP

```

### 33.467 stdair/bom/SegmentPeriodTypes.hpp File Reference

```

#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>

```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

#### Typedefs

- typedef std::list< SegmentPeriod \* > [stdair::SegmentPeriodList\\_T](#)
- typedef std::map< const MapKey\_T, SegmentPeriod \* > [stdair::SegmentPeriodMap\\_T](#)
- typedef std::pair< MapKey\_T, SegmentPeriod \* > [stdair::SegmentPeriodWithKey\\_T](#)
- typedef std::list< SegmentPeriodWithKey\_T > [stdair::SegmentPeriodDetailedList\\_T](#)

### 33.468 SegmentPeriodTypes.hpp

```

00001 // ////////////////////////////////////////////
00002 #ifndef __STDAIR_BOM_SEGMENTPERIODTYPES_HPP
00003 #define __STDAIR_BOM_SEGMENTPERIODTYPES_HPP
00004
00005 // ////////////////////////////////////////////
00006 // Import section
00007 // ////////////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>

```

```

00011 // StdAir
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016     // Forward declarations.
00017     class SegmentPeriod;
00018
00020     typedef std::list<SegmentPeriod*> SegmentPeriodList_T;
00021
00023     typedef std::map<const MapKey_T, SegmentPeriod*> SegmentPeriodMap_T;
00024
00026     typedef std::pair<MapKey_T, SegmentPeriod*> SegmentPeriodWithKey_T;
00027     typedef std::list<SegmentPeriodWithKey_T> SegmentPeriodDetailedList_T;
00028 }
00029 #endif // __STDAIR_BOM_SEGMENTPERIODTYPES_HPP
00030

```

### 33.469 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

- [stdair](#)

*Handle on the StdAir library context.*

### 33.470 SegmentSnapshotTable.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost
00008 #include <boost/multi_array.hpp>
00009 // Boost.Serialization
00010 #include <boost/archive/text_iarchive.hpp>
00011 #include <boost/archive/text_oarchive.hpp>
00012 #include <boost/serialization/access.hpp>
00013 // StdAir
00014 #include <stdair/basic/BasConst_Inventory.hpp>
00015 #include <stdair/bom/BomManager.hpp>
00016 #include <stdair/bom/SegmentSnapshotTable.hpp>
00017
00018 namespace stdair {
00019
00020     // //////////////////////////////////////
00021     SegmentSnapshotTable::SegmentSnapshotTable()
00022         : _key (DEFAULT_TABLE_ID), _parent (NULL) {
00023         assert (false);
00024     }
00025
00026     // //////////////////////////////////////
00027     SegmentSnapshotTable::SegmentSnapshotTable (const SegmentSnapshotTable&)
00028         : _key (DEFAULT_TABLE_ID), _parent (NULL) {
00029         assert (false);
00030     }
00031
00032     // //////////////////////////////////////
00033     SegmentSnapshotTable::
00034     SegmentSnapshotTable (const Key_T& iKey) : _key (iKey), _parent (NULL) {

```

```

00035     }
00036
00037     // //////////////////////////////////////
00038     SegmentSnapshotTable::~SegmentSnapshotTable() {
00039     }
00040
00041     // //////////////////////////////////////
00042     std::string SegmentSnapshotTable::toString() const {
00043         std::ostringstream ostr;
00044         ostr << describeKey();
00045         return ostr.str();
00046     }
00047
00048     // //////////////////////////////////////
00049     void SegmentSnapshotTable::
00050     initSnapshotBlocks (const SegmentCabinIndexMap_T&
00051         iSegmentCabinIndexMap,
00052         const ClassIndexMap_T& iClassIndexMap) {
00053         _segmentCabinIndexMap = iSegmentCabinIndexMap;
00054         _classIndexMap = iClassIndexMap;
00055
00056         unsigned int lNumberOfSegmentCabins = _segmentCabinIndexMap.size();
00057         unsigned int lNumberOfClasses = _classIndexMap.size();
00058
00059         // Initialise the snapshot blocks
00060         // Normally, the block includes snapshots from DTD MAX to DTD 0, thus
00061         // DEFAULT_MAX_DTD + 1 values. However, we would like to add the day
00062         // before DTD MAX (this value will be initialised to zero).
00063         _bookingSnapshotBlock.
00064             resize (boost::extents[lNumberOfSegmentCabins*lNumberOfClasses]
00065                 [DEFAULT_MAX_DTD + 2]);
00066         _cancellationSnapshotBlock.
00067             resize (boost::extents[lNumberOfSegmentCabins*lNumberOfClasses]
00068                 [DEFAULT_MAX_DTD + 2]);
00069         _productOrientedNetBookingSnapshotBlock.
00070             resize (boost::extents[lNumberOfSegmentCabins*lNumberOfClasses]
00071                 [DEFAULT_MAX_DTD + 2]);
00072         _priceOrientedNetBookingSnapshotBlock.
00073             resize (boost::extents[lNumberOfSegmentCabins*lNumberOfClasses]
00074                 [DEFAULT_MAX_DTD + 2]);
00075         _productOrientedGrossBookingSnapshotBlock.
00076             resize (boost::extents[lNumberOfSegmentCabins*lNumberOfClasses]
00077                 [DEFAULT_MAX_DTD + 2]);
00078         _priceOrientedGrossBookingSnapshotBlock.
00079             resize (boost::extents[lNumberOfSegmentCabins*lNumberOfClasses]
00080                 [DEFAULT_MAX_DTD + 2]);
00081         _availabilitySnapshotBlock.
00082             resize (boost::extents[lNumberOfSegmentCabins*lNumberOfClasses]
00083                 [DEFAULT_MAX_DTD + 2]);
00084     }
00085
00086     // //////////////////////////////////////
00087     const ClassIndex_T& SegmentSnapshotTable::
00088     getClassIndex (const MapKey_T& iKey) const {
00089         ClassIndexMap_T::const_iterator itVTIdx =
00090             _classIndexMap.find (iKey);
00091         assert (itVTIdx != _classIndexMap.end());
00092         return itVTIdx->second;
00093     }
00094
00095     // //////////////////////////////////////
00096     const SegmentDataID_T& SegmentSnapshotTable::
00097     getSegmentDataID (const SegmentCabin& iSegmentCabin) const {
00098         SegmentCabinIndexMap_T::const_iterator itSCIdx =
00099             _segmentCabinIndexMap.find (&iSegmentCabin);
00100         assert (itSCIdx != _segmentCabinIndexMap.end());
00101         return itSCIdx->second;
00102     }
00103
00104     // //////////////////////////////////////
00105     ConstSegmentCabinDTDSnapshotView_T
00106     SegmentSnapshotTable::
00107     getConstSegmentCabinDTDBookingSnapshotView (const
00108         SegmentDataID_T iSCIdxBegin,
00109         const SegmentDataID_T iSCIdxEnd,
00110         const DTD_T iDTD) const {
00111         const unsigned int lNbOfClasses = _classIndexMap.size();
00112         const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
00113         const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;
00114         return _bookingSnapshotBlock [ boost::indices[
00115             SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][iDTD] ];
00116     }
00117
00118     // //////////////////////////////////////
00119     ConstSegmentCabinDTDRangeSnapshotView_T

```

```

SegmentSnapshotTable::
00118   getConstSegmentCabinDTRangeBookingSnapshotView
00119   (const SegmentDataID_T iSCIdxBegin, const SegmentDataID_T iSCIdxEnd,
00120    const DTD_T iDTDBegin, const DTD_T iDTDEnd) const {
00121       const unsigned int lNbOfClasses = _classIndexMap.size();
00122       const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
00123       const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;
00124
00125       return _bookingSnapshotBlock [ boost::indices[SnapshotBlockRange_T(lClassIdxBegin,
lClassIdxEnd)][SnapshotBlockRange_T(iDTDBegin, iDTDEnd + 1)] ];
00126   }
00127
00128   // //////////////////////////////////////
00129   SegmentCabinDTRangeSnapshotView_T
SegmentSnapshotTable::
00130   getSegmentCabinDTRangeBookingSnapshotView (const
SegmentDataID_T iSCIdxBegin,
00131                                              const SegmentDataID_T iSCIdxEnd,
00132                                              const DTD_T iDTDEnd) {
00133       const unsigned int lNbOfClasses = _classIndexMap.size();
00134       const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
00135       const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;
00136
00137       return _bookingSnapshotBlock [ boost::indices[
SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][iDTDEnd] ];
00138   }
00139
00140   // //////////////////////////////////////
00141   SegmentCabinDTRangeSnapshotView_T
SegmentSnapshotTable::
00142   getSegmentCabinDTRangeBookingSnapshotView (const
SegmentDataID_T iSCIdxBegin,
00143                                              const SegmentDataID_T iSCIdxEnd,
00144                                              const DTD_T iDTDBegin,
00145                                              const DTD_T iDTDEnd) {
00146       const unsigned int lNbOfClasses = _classIndexMap.size();
00147       const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
00148       const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;
00149
00150       return _bookingSnapshotBlock [ boost::indices[
SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][
SnapshotBlockRange_T(iDTDBegin, iDTDEnd + 1)] ];
00151   }
00152
00153   // //////////////////////////////////////
00154   ConstSegmentCabinDTRangeSnapshotView_T
SegmentSnapshotTable::
00155   getConstSegmentCabinDTRangeCancellationSnapshotView (const
SegmentDataID_T iSCIdxBegin,
00156                                              const SegmentDataID_T iSCIdxEnd,
00157                                              const DTD_T iDTDEnd) const {
00158       const unsigned int lNbOfClasses = _classIndexMap.size();
00159       const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
00160       const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;
00161
00162       return _cancellationSnapshotBlock [ boost::indices[
SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][iDTDEnd] ];
00163   }
00164
00165   // //////////////////////////////////////
00166   ConstSegmentCabinDTRangeSnapshotView_T
SegmentSnapshotTable::
00167   getConstSegmentCabinDTRangeCancellationSnapshotView
00168   (const SegmentDataID_T iSCIdxBegin, const SegmentDataID_T iSCIdxEnd,
00169    const DTD_T iDTDBegin, const DTD_T iDTDEnd) const {
00170       const unsigned int lNbOfClasses = _classIndexMap.size();
00171       const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
00172       const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;
00173
00174       return _cancellationSnapshotBlock [ boost::indices[SnapshotBlockRange_T(
lClassIdxBegin, lClassIdxEnd)][SnapshotBlockRange_T(iDTDBegin, iDTDEnd + 1)] ];
00175   }
00176
00177   // //////////////////////////////////////
00178   SegmentCabinDTRangeSnapshotView_T
SegmentSnapshotTable::
00179   getSegmentCabinDTRangeCancellationSnapshotView (const
SegmentDataID_T iSCIdxBegin,
00180                                              const SegmentDataID_T iSCIdxEnd,
00181                                              const DTD_T iDTDEnd) {
00182       const unsigned int lNbOfClasses = _classIndexMap.size();
00183       const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
00184       const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;
00185
00186       return _cancellationSnapshotBlock [ boost::indices[
SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][iDTDEnd] ];
00187   }

```

```

00188
00189 ///////////////////////////////////////////////////////////////////
00190 SegmentCabinDTRangeSnapshotView_T
SegmentSnapshotTable::
00191 getSegmentCabinDTRangeCancellationSnapshotView(const
SegmentDataID_T iSCIdxBegin,
00192                                     const SegmentDataID_T iSCIdxEnd,
00193                                     const DTD_T iDTDBegin,
00194                                     const DTD_T iDTDEnd) {
00195     const unsigned int lNbOfClasses = _classIndexMap.size();
00196     const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
00197     const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;
00198
00199     return _cancellationSnapshotBlock [ boost::indices[
SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][
SnapshotBlockRange_T(iDTDBegin, iDTDEnd + 1)] ];
00200 }
00201
00202 ///////////////////////////////////////////////////////////////////
00203 ConstSegmentCabinDTRangeSnapshotView_T
SegmentSnapshotTable::
00204 getConstSegmentCabinDTRangeProductOrientedNetBookingSnapshotView
(const SegmentDataID_T iSCIdxBegin,
00205                                     const SegmentDataID_T iSCIdxEnd,
00206                                     const DTD_T iDTDBegin, const {
00207     const unsigned int lNbOfClasses = _classIndexMap.size();
00208     const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
00209     const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;
00210
00211     return _productOrientedNetBookingSnapshotBlock [ boost::indices[
SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][iDTD] ];
00212 }
00213
00214 ///////////////////////////////////////////////////////////////////
00215 ConstSegmentCabinDTRangeSnapshotView_T
SegmentSnapshotTable::
00216 getConstSegmentCabinDTRangeProductOrientedNetBookingSnapshotView
(const SegmentDataID_T iSCIdxBegin, const SegmentDataID_T iSCIdxEnd,
00217     const DTD_T iDTDBegin, const DTD_T iDTDEnd) const {
00218     const unsigned int lNbOfClasses = _classIndexMap.size();
00219     const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
00220     const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;
00221
00222     return _productOrientedNetBookingSnapshotBlock [ boost::indices[
SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][
SnapshotBlockRange_T(iDTDBegin, iDTDEnd + 1)] ];
00223 }
00224 }
00225
00226 ///////////////////////////////////////////////////////////////////
00227 SegmentCabinDTRangeSnapshotView_T
SegmentSnapshotTable::
00228 getSegmentCabinDTRangeProductOrientedNetBookingSnapshotView
(const SegmentDataID_T iSCIdxBegin,
00229                                     const SegmentDataID_T iSCIdxEnd,
00230                                     const DTD_T iDTDBegin, {
00231     const unsigned int lNbOfClasses = _classIndexMap.size();
00232     const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
00233     const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;
00234
00235     return _productOrientedNetBookingSnapshotBlock [ boost::indices[
SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][iDTD] ];
00236 }
00237
00238 ///////////////////////////////////////////////////////////////////
00239 SegmentCabinDTRangeSnapshotView_T
SegmentSnapshotTable::
00240 getSegmentCabinDTRangeProductOrientedNetBookingSnapshotView
(const SegmentDataID_T iSCIdxBegin,
00241                                     const SegmentDataID_T iSCIdxEnd,
00242                                     const DTD_T iDTDBegin,
00243                                     const DTD_T iDTDEnd) {
00244     const unsigned int lNbOfClasses = _classIndexMap.size();
00245     const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
00246     const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;
00247
00248     return _productOrientedNetBookingSnapshotBlock [ boost::indices[
SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][
SnapshotBlockRange_T(iDTDBegin, iDTDEnd + 1)] ];
00249 }
00250
00251 ///////////////////////////////////////////////////////////////////
00252 ConstSegmentCabinDTRangeSnapshotView_T
SegmentSnapshotTable::
00253 getConstSegmentCabinDTRangePriceOrientedNetBookingSnapshotView
(const SegmentDataID_T iSCIdxBegin,
00254                                     const SegmentDataID_T iSCIdxEnd,

```

```

00256                                     const DTD_T iDTD) const {
00257     const unsigned int lNbOfClasses = _classIndexMap.size();
00258     const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
00259     const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;
00260
00261     return _priceOrientedNetBookingSnapshotBlock [ boost::indices[
SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][iDTD] ];
00262 }
00263
00264 // //////////////////////////////////////
00265 ConstSegmentCabinDTDRangeSnapshotView_T
SegmentSnapshotTable::
00266 getConstSegmentCabinDTDRangePriceOrientedNetBookingSnapshotView
00267 (const SegmentDataID_T iSCIdxBegin, const SegmentDataID_T iSCIdxEnd,
00268  const DTD_T iDTDBegin, const DTD_T iDTDEnd) const {
00269     const unsigned int lNbOfClasses = _classIndexMap.size();
00270     const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
00271     const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;
00272
00273     return _priceOrientedNetBookingSnapshotBlock [ boost::indices[
SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][
SnapshotBlockRange_T(iDTDBegin, iDTDEnd + 1)] ];
00274 }
00275
00276 // //////////////////////////////////////
00277 SegmentCabinDTDSnapshotView_T
SegmentSnapshotTable::
00278 getSegmentCabinDTDRangePriceOrientedNetBookingSnapshotView
00279 (const SegmentDataID_T iSCIdxBegin,
                                const SegmentDataID_T iSCIdxEnd,
00280                                const DTD_T iDTDBegin,
00281                                const DTD_T iDTDEnd) {
00282     const unsigned int lNbOfClasses = _classIndexMap.size();
00283     const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
00284     const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;
00285
00286     return _priceOrientedNetBookingSnapshotBlock [ boost::indices[
SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][iDTD] ];
00287 }
00288 // //////////////////////////////////////
00289 SegmentCabinDTDRangeSnapshotView_T
SegmentSnapshotTable::
00290 getSegmentCabinDTDRangePriceOrientedNetBookingSnapshotView
00291 (const SegmentDataID_T iSCIdxBegin,
                                const SegmentDataID_T iSCIdxEnd,
00292                                const DTD_T iDTDBegin,
00293                                const DTD_T iDTDEnd) {
00294     const unsigned int lNbOfClasses = _classIndexMap.size();
00295     const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
00296     const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;
00297
00298     return _priceOrientedNetBookingSnapshotBlock [ boost::indices[
SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][
SnapshotBlockRange_T(iDTDBegin, iDTDEnd + 1)] ];
00299 }
00300
00301 // //////////////////////////////////////
00302 ConstSegmentCabinDTDSnapshotView_T
SegmentSnapshotTable::
00303 getConstSegmentCabinDTDRangeProductOrientedGrossBookingSnapshotView
00304 (const SegmentDataID_T iSCIdxBegin,
                                const SegmentDataID_T iSCIdxEnd,
00305                                const DTD_T iDTDBegin,
00306                                const DTD_T iDTDEnd) const {
00307     const unsigned int lNbOfClasses = _classIndexMap.size();
00308     const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
00309     const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;
00310
00311     return _productOrientedGrossBookingSnapshotBlock [
boost::indices[SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][iDTD] ];
00312 }
00313 // //////////////////////////////////////
00314 ConstSegmentCabinDTDRangeSnapshotView_T
SegmentSnapshotTable::
00315 getConstSegmentCabinDTDRangeProductOrientedGrossBookingSnapshotView
00316 (const SegmentDataID_T iSCIdxBegin, const SegmentDataID_T iSCIdxEnd,
00317  const DTD_T iDTDBegin, const DTD_T iDTDEnd) const {
00318     const unsigned int lNbOfClasses = _classIndexMap.size();
00319     const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
00320     const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;
00321
00322     return _productOrientedGrossBookingSnapshotBlock [ boost::indices[
SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][
SnapshotBlockRange_T(iDTDBegin, iDTDEnd + 1)] ];
00323 }
00324
00325 // //////////////////////////////////////

```

```

00326     SegmentCabinDTDSnapshotView_T
SegmentSnapshotTable::
00327     getSegmentCabinDTDPProductOrientedGrossBookingSnapshotView
(const SegmentDataID_T iSCIdxBegin,
00328                                     const SegmentDataID_T iSCIdxEnd,
00329                                     const DTD_T iDTD) {
00330         const unsigned int lNbOfClasses = _classIndexMap.size();
00331         const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
00332         const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;
00333
00334         return _productOrientedGrossBookingSnapshotBlock [
boost::indices[SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][iDTD] ];
00335     }
00336
00337     // //////////////////////////////////////
00338     SegmentCabinDTDRangeSnapshotView_T
SegmentSnapshotTable::
00339     getSegmentCabinDTDRangeProductOrientedGrossBookingSnapshotView
(const SegmentDataID_T iSCIdxBegin,
00340                                     const SegmentDataID_T iSCIdxEnd,
00341                                     const DTD_T iDTDBegin,
00342                                     const DTD_T iDTDEnd) {
00343         const unsigned int lNbOfClasses = _classIndexMap.size();
00344         const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
00345         const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;
00346
00347         return _productOrientedGrossBookingSnapshotBlock [
boost::indices[SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][
SnapshotBlockRange_T(iDTDBegin, iDTDEnd + 1)] ];
00348     }
00349
00350
00351     // //////////////////////////////////////
00352     ConstSegmentCabinDTDSnapshotView_T
SegmentSnapshotTable::
00353     getConstSegmentCabinDTDPPriceOrientedGrossBookingSnapshotView
(const SegmentDataID_T iSCIdxBegin,
00354                                     const SegmentDataID_T iSCIdxEnd,
00355                                     const DTD_T iDTD) const {
00356         const unsigned int lNbOfClasses = _classIndexMap.size();
00357         const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
00358         const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;
00359
00360         return _priceOrientedGrossBookingSnapshotBlock [ boost::indices[
SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][iDTD] ];
00361     }
00362
00363     // //////////////////////////////////////
00364     ConstSegmentCabinDTDRangeSnapshotView_T
SegmentSnapshotTable::
00365     getConstSegmentCabinDTDRangePriceOrientedGrossBookingSnapshotView
00366     (const SegmentDataID_T iSCIdxBegin, const SegmentDataID_T iSCIdxEnd,
00367     const DTD_T iDTDBegin, const DTD_T iDTDEnd) const {
00368         const unsigned int lNbOfClasses = _classIndexMap.size();
00369         const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
00370         const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;
00371
00372         return _priceOrientedGrossBookingSnapshotBlock [ boost::indices[
SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][
SnapshotBlockRange_T(iDTDBegin, iDTDEnd + 1)] ];
00373     }
00374
00375     // //////////////////////////////////////
00376     SegmentCabinDTDSnapshotView_T
SegmentSnapshotTable::
00377     getSegmentCabinDTDPPriceOrientedGrossBookingSnapshotView
00378     (const SegmentDataID_T iSCIdxBegin, const SegmentDataID_T iSCIdxEnd,
00379     const DTD_T iDTD) {
00380         const unsigned int lNbOfClasses = _classIndexMap.size();
00381         const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
00382         const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;
00383
00384         return _priceOrientedGrossBookingSnapshotBlock [ boost::indices[
SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][iDTD] ];
00385     }
00386
00387     // //////////////////////////////////////
00388     SegmentCabinDTDRangeSnapshotView_T
SegmentSnapshotTable::
00389     getSegmentCabinDTDRangePriceOrientedGrossBookingSnapshotView
00390     (const SegmentDataID_T iSCIdxBegin, const SegmentDataID_T iSCIdxEnd,
00391     const DTD_T iDTDBegin, const DTD_T iDTDEnd) {
00392         const unsigned int lNbOfClasses = _classIndexMap.size();
00393         const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
00394         const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;
00395
00396         return _priceOrientedGrossBookingSnapshotBlock [ boost::indices[

```

```

        SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][
        SnapshotBlockRange_T(iDTDBegin, iDTDEnd + 1)] ];
00397     }
00398
00399     // //////////////////////////////////////
00400     ConstSegmentCabinDTDSnapshotView_T
SegmentSnapshotTable::
00401     getConstSegmentCabinDTDAvailabilitySnapshotView
00402     (const SegmentDataID_T iSCIdxBegin, const SegmentDataID_T iSCIdxEnd,
00403      const DTD_T iDTD) const {
00404         const unsigned int lNbOfClasses = _classIndexMap.size();
00405         const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
00406         const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;
00407
00408         return _availabilitySnapshotBlock [ boost::indices[SnapshotBlockRange_T(
lClassIdxBegin, lClassIdxEnd)][iDTD] ];
00409     }
00410
00411     // //////////////////////////////////////
00412     ConstSegmentCabinDTRangeSnapshotView_T
SegmentSnapshotTable::
00413     getConstSegmentCabinDTRangeAvailabilitySnapshotView
00414     (const SegmentDataID_T iSCIdxBegin, const SegmentDataID_T iSCIdxEnd,
00415      const DTD_T iDTDBegin, const DTD_T iDTDEnd) const {
00416         const unsigned int lNbOfClasses = _classIndexMap.size();
00417         const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
00418         const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;
00419
00420         return _availabilitySnapshotBlock [ boost::indices[SnapshotBlockRange_T(
lClassIdxBegin, lClassIdxEnd)][SnapshotBlockRange_T(iDTDBegin, iDTDEnd + 1)] ];
00421     }
00422
00423     // //////////////////////////////////////
00424     SegmentCabinDTDSnapshotView_T
SegmentSnapshotTable::
00425     getSegmentCabinDTDAvailabilitySnapshotView (const
SegmentDataID_T iSCIdxBegin,
00426                                                  const SegmentDataID_T iSCIdxEnd,
00427                                                  const DTD_T iDTD) {
00428         const unsigned int lNbOfClasses = _classIndexMap.size();
00429         const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
00430         const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;
00431
00432         return _availabilitySnapshotBlock [ boost::indices[
SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][iDTD] ];
00433     }
00434
00435     // //////////////////////////////////////
00436     SegmentCabinDTRangeSnapshotView_T
SegmentSnapshotTable::
00437     getSegmentCabinDTRangeAvailabilitySnapshotView(const
SegmentDataID_T iSCIdxBegin,
00438                                                  const SegmentDataID_T iSCIdxEnd,
00439                                                  const DTD_T iDTDBegin,
00440                                                  const DTD_T iDTDEnd) {
00441         const unsigned int lNbOfClasses = _classIndexMap.size();
00442         const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
00443         const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;
00444
00445         return _availabilitySnapshotBlock [ boost::indices[
SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][
SnapshotBlockRange_T(iDTDBegin, iDTDEnd + 1)] ];
00446     }
00447
00448     // //////////////////////////////////////
00449     void SegmentSnapshotTable::serialisationImplementationExport() const {
00450         std::ostringstream oStr;
00451         boost::archive::text_oarchive oa (oStr);
00452         oa << *this;
00453     }
00454
00455     // //////////////////////////////////////
00456     void SegmentSnapshotTable::serialisationImplementationImport() {
00457         std::istringstream iStr;
00458         boost::archive::text_iarchive ia (iStr);
00459         ia >> *this;
00460     }
00461
00462     // //////////////////////////////////////
00463     template<class Archive>
00464     void SegmentSnapshotTable::serialize (Archive& ioArchive,
00465                                          const unsigned int iFileVersion) {
00466         ioArchive & _key;
00467     }
00468 }
00469
00470

```



### 33.471 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>
```

#### Classes

- class [stdair::SegmentSnapshotTable](#)  
*Class representing the actual attributes for an airline segment data tables.*

#### Namespaces

- [boost](#)  
*Forward declarations.*
- [boost::serialization](#)
- [stdair](#)  
*Handle on the StdAir library context.*

### 33.472 SegmentSnapshotTable.hpp

```
00001 #ifndef __STDAIR_BOM_SEGMENTSNAPOSHOTTABLE_HPP
00002 #define __STDAIR_BOM_SEGMENTSNAPOSHOTTABLE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/bom/BomAbstract.hpp>
00013 #include <stdair/bom/SegmentSnapshotTableKey.hpp>
00014 #include <stdair/bom/SegmentSnapshotTableTypes.hpp>
00015
00016 namespace boost {
00017     namespace serialization {
00018         class access;
00019     }
00020 }
00021
00022 namespace stdair {
00023     // Forward declarations
00024     class SegmentCabin;
00025
00026     class SegmentSnapshotTable : public BomAbstract {
00027     public:
00028         template <typename BOM> friend class FacBom;
00029         friend class FacBomManager;
00030         friend class boost::serialization::access;
00031
00032     public:
00033         // ////////////////////////////////// Type definitions //////////////////////////////////
00034         typedef SegmentSnapshotTableKey Key_T;
00035
00036     public:
00037         // ////////////////////////////////// Getters //////////////////////////////////
00038         const Key_T& getKey() const {
00039             return _key;
00040         }
00041
00042         BomAbstract* const getParent() const {
00043             return _parent;
00044         }
00045     };
00046 }
```

```

00057     const TableID_T& getTableID() const {
00058         return _key.getTableID();
00059     }
00060
00064     const HolderMap_T& getHolderMap() const {
00065         return _holderMap;
00066     }
00067
00069     const SegmentCabinIndexMap_T& getSegmentCabinIndexMap()
00070     const {
00071         return _segmentCabinIndexMap;
00072     }
00073
00074     const ClassIndexMap_T& getClassIndexMap() const {
00075         return _classIndexMap;
00076     }
00077
00079     const ClassIndex_T& getClassIndex (const MapKey_T&) const;
00080
00082     const SegmentDataID_T& getSegmentDataID (const
00083     SegmentCabin&) const;
00084
00086     ConstSegmentCabinDTDSnapshotView_T
00087     getConstSegmentCabinDTDBookingSnapshotView (const
00088     SegmentDataID_T,
00089                                     const SegmentDataID_T,
00090                                     const DTD_T) const;
00091
00093     ConstSegmentCabinDTDRangeSnapshotView_T
00094     getConstSegmentCabinDTDRangeBookingSnapshotView (const
00095     SegmentDataID_T,
00096                                     const SegmentDataID_T,
00097                                     const DTD_T,
00098                                     const DTD_T) const;
00099
00101     SegmentCabinDTDSnapshotView_T
00102     getSegmentCabinDTDBookingSnapshotView (const
00103     SegmentDataID_T,
00104                                     const SegmentDataID_T, const
00105     DTD_T);
00106
00107     SegmentCabinDTDRangeSnapshotView_T
00108     getSegmentCabinDTDRangeBookingSnapshotView (const
00109     SegmentDataID_T,
00110                                     const SegmentDataID_T,
00111                                     const DTD_T, const DTD_T);
00112
00114     ConstSegmentCabinDTDSnapshotView_T
00115     getConstSegmentCabinDTDCancellationSnapshotView (const
00116     SegmentDataID_T,
00117                                     const SegmentDataID_T,
00118                                     const DTD_T) const;
00119
00121     ConstSegmentCabinDTDRangeSnapshotView_T
00122     getConstSegmentCabinDTDRangeCancellationSnapshotView
00123     (const SegmentDataID_T,
00124                                     const SegmentDataID_T,
00125                                     const DTD_T,
00126                                     const DTD_T) const;
00127
00129     SegmentCabinDTDSnapshotView_T
00130     getSegmentCabinDTDCancellationSnapshotView (const
00131     SegmentDataID_T,
00132                                     const SegmentDataID_T,
00133                                     const DTD_T);
00134
00136     SegmentCabinDTDRangeSnapshotView_T
00137     getSegmentCabinDTDRangeCancellationSnapshotView (const
00138     SegmentDataID_T,
00139                                     const SegmentDataID_T,
00140                                     const DTD_T, const
00141     DTD_T);
00142
00143     ConstSegmentCabinDTDSnapshotView_T
00144     getConstSegmentCabinDTDProductOrientedNetBookingSnapshotView
00145     (const SegmentDataID_T, const SegmentDataID_T, const
00146     DTD_T) const;
00147
00149     ConstSegmentCabinDTDRangeSnapshotView_T
00150     getConstSegmentCabinDTDRangeProductOrientedNetBookingSnapshotView
00151     (const SegmentDataID_T, const SegmentDataID_T, const
00152     DTD_T, const DTD_T) const;
00153
00155     SegmentCabinDTDSnapshotView_T
00156     getSegmentCabinDTDProductOrientedNetBookingSnapshotView
00157     (const SegmentDataID_T, const SegmentDataID_T, const
00158     DTD_T);

```

```
00158
00161     SegmentCabinDTDRangeSnapshotView_T
00162     getSegmentCabinDTDRangeProductOrientedNetBookingSnapshotView
00163     (const SegmentDataID_T, const SegmentDataID_T, const
00164     DTD_T, const DTD_T);
00164
00167     ConstSegmentCabinDTDSnapshotView_T
00168     getConstSegmentCabinDTDPPriceOrientedNetBookingSnapshotView
00169     (const SegmentDataID_T, const SegmentDataID_T, const
00170     DTD_T) const;
00170
00173     ConstSegmentCabinDTDRangeSnapshotView_T
00174     getConstSegmentCabinDTDRangePriceOrientedNetBookingSnapshotView
00175     (const SegmentDataID_T, const SegmentDataID_T, const
00176     DTD_T, const DTD_T) const;
00176
00179     SegmentCabinDTDSnapshotView_T
00180     getSegmentCabinDTDPPriceOrientedNetBookingSnapshotView
00181     (const SegmentDataID_T, const SegmentDataID_T, const
00182     DTD_T);
00182
00185     SegmentCabinDTDRangeSnapshotView_T
00186     getSegmentCabinDTDRangePriceOrientedNetBookingSnapshotView
00187     (const SegmentDataID_T, const SegmentDataID_T, const
00188     DTD_T, const DTD_T);
00188
00189
00192     ConstSegmentCabinDTDSnapshotView_T
00193     getConstSegmentCabinDTDPProductOrientedGrossBookingSnapshotView
00194     (const SegmentDataID_T, const SegmentDataID_T, const
00195     DTD_T) const;
00195
00198     ConstSegmentCabinDTDRangeSnapshotView_T
00199     getConstSegmentCabinDTDRangeProductOrientedGrossBookingSnapshotView
00200     (const SegmentDataID_T, const SegmentDataID_T, const
00201     DTD_T, const DTD_T) const;
00201
00204     SegmentCabinDTDSnapshotView_T
00205     getSegmentCabinDTDPProductOrientedGrossBookingSnapshotView
00206     (const SegmentDataID_T, const SegmentDataID_T, const
00207     DTD_T);
00207
00210     SegmentCabinDTDRangeSnapshotView_T
00211     getSegmentCabinDTDRangeProductOrientedGrossBookingSnapshotView
00212     (const SegmentDataID_T, const SegmentDataID_T, const
00213     DTD_T, const DTD_T);
00213
00216     ConstSegmentCabinDTDSnapshotView_T
00217     getConstSegmentCabinDTDPPriceOrientedGrossBookingSnapshotView
00218     (const SegmentDataID_T, const SegmentDataID_T, const
00219     DTD_T) const;
00219
00222     ConstSegmentCabinDTDRangeSnapshotView_T
00223     getConstSegmentCabinDTDRangePriceOrientedGrossBookingSnapshotView
00224     (const SegmentDataID_T, const SegmentDataID_T, const
00225     DTD_T, const DTD_T) const;
00225
00228     SegmentCabinDTDSnapshotView_T
00229     getSegmentCabinDTDPPriceOrientedGrossBookingSnapshotView
00230     (const SegmentDataID_T, const SegmentDataID_T, const
00231     DTD_T);
00231
00234     SegmentCabinDTDRangeSnapshotView_T
00235     getSegmentCabinDTDRangePriceOrientedGrossBookingSnapshotView
00236     (const SegmentDataID_T, const SegmentDataID_T, const
00237     DTD_T, const DTD_T);
00237
00238
00241     ConstSegmentCabinDTDSnapshotView_T
00242     getConstSegmentCabinDTDAvailabilitySnapshotView (const
00243     SegmentDataID_T,
00244     const SegmentDataID_T,
00245     const DTD_T) const;
00245
00248     ConstSegmentCabinDTDRangeSnapshotView_T
00249     getConstSegmentCabinDTDRangeAvailabilitySnapshotView
00250     (const SegmentDataID_T,
00251     const SegmentDataID_T,
00252     const DTD_T) const;
00252
00253
00256     SegmentCabinDTDSnapshotView_T
00257     getSegmentCabinDTDAvailabilitySnapshotView (const
00258     SegmentDataID_T,
00259     const SegmentDataID_T,
00260     const DTD_T);
```

```

00263     SegmentCabinDTRangeSnapshotView_T
00264     getSegmentCabinDTRangeAvailabilitySnapshotView (const
SegmentDataID_T,
00265                                                     const SegmentDataID_T,
00266                                                     const DTD_T, const
DTD_T);
00267
00268
00269 public:
00270     // //////////// Setters ////////////
00273     void initSnapshotBlocks (const SegmentCabinIndexMap_T&,
00274                             const ClassIndexMap_T&);
00275
00276 public:
00277     // //////////// Display support methods ////////////
00283     void toStream (std::ostream& ioOut) const {
00284         ioOut << toString();
00285     }
00286
00292     void fromStream (std::istream& ioIn) {
00293     }
00294
00298     std::string toString() const;
00299
00303     const std::string describeKey() const {
00304         return _key.toString();
00305     }
00306
00307
00308 public:
00309     // //////////// (Boost) Serialisation support methods ////////////
00313     template<class Archive>
00314     void serialize (Archive& ar, const unsigned int iFileVersion);
00315
00316 private:
00321     void serialisationImplementationExport() const;
00322     void serialisationImplementationImport();
00323
00324
00325 protected:
00326     // //////////// Constructors and destructors ////////////
00330     SegmentSnapshotTable (const Key_T&);
00331
00335     virtual ~SegmentSnapshotTable();
00336
00337 private:
00341     SegmentSnapshotTable();
00342
00346     SegmentSnapshotTable (const SegmentSnapshotTable&);
00347
00348
00349 protected:
00350     // //////////// Attributes ////////////
00352     Key_T _key;
00353
00355     BomAbstract* _parent;
00356
00358     HolderMap_T _holderMap;
00359
00361     SegmentCabinIndexMap_T _segmentCabinIndexMap;
00362
00365     ClassIndexMap_T _classIndexMap;
00366
00368     SnapshotBlock_T _bookingSnapshotBlock;
00369
00371     SnapshotBlock_T _cancellationSnapshotBlock;
00372
00374     SnapshotBlock_T _productOrientedNetBookingSnapshotBlock
;
00375
00377     SnapshotBlock_T _priceOrientedNetBookingSnapshotBlock
;
00378
00380     SnapshotBlock_T _productOrientedGrossBookingSnapshotBlock
;
00381
00383     SnapshotBlock_T _priceOrientedGrossBookingSnapshotBlock
;
00384
00386     SnapshotBlock_T _availabilitySnapshotBlock;
00387 };
00388
00389 }
00390 #endif // __STDAIR_BOM_SEGMENTSNAPOSHOTTABLE_HPP
00391

```

### 33.473 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

- [stdair](#)  
*Handle on the StdAir library context.*

#### 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)

### 33.474 SegmentSnapshotTableKey.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/access.hpp>
00011 // StdAir
00012 #include <stdair/basic/BasConst_Inventory.hpp>
00013 #include <stdair/basic/BasConst_BomDisplay.hpp>
00014 #include <stdair/bom/SegmentSnapshotTableKey.hpp>
00015
00016 namespace stdair {
00017
00018 // //////////////////////////////////////
00019 SegmentSnapshotTableKey::SegmentSnapshotTableKey()
00020 : _tableID (DEFAULT_TABLE_ID) {
00021     assert (false);
00022 }
00023
00024 // //////////////////////////////////////
00025 SegmentSnapshotTableKey::
00026 SegmentSnapshotTableKey (const TableID_T& iTableID)
00027 : _tableID (iTableID) {
00028 }
00029
00030 // //////////////////////////////////////
00031 SegmentSnapshotTableKey::SegmentSnapshotTableKey (const
SegmentSnapshotTableKey& iKey)
00032 : _tableID (iKey._tableID) {
00033 }
00034
00035 // //////////////////////////////////////
00036 SegmentSnapshotTableKey::~SegmentSnapshotTableKey() {
00037 }
00038
00039 // //////////////////////////////////////
00040 void SegmentSnapshotTableKey::toStream (std::ostream& ioOut) const {
00041     ioOut << "SegmentSnapshotTableKey: " << toString();
00042 }
00043
00044 // //////////////////////////////////////
```

```

00045 void SegmentSnapshotTableKey::fromStream (std::istream& ioIn) {
00046 }
00047
00048 // //////////////////////////////////////
00049 const std::string SegmentSnapshotTableKey::toString() const {
00050     std::ostringstream oStr;
00051     oStr << _tableID;
00052     return oStr.str();
00053 }
00054
00055 // //////////////////////////////////////
00056 void SegmentSnapshotTableKey::serialisationImplementationExport() const {
00057     std::ostringstream oStr;
00058     boost::archive::text_oarchive oa (oStr);
00059     oa << *this;
00060 }
00061
00062 // //////////////////////////////////////
00063 void SegmentSnapshotTableKey::serialisationImplementationImport() {
00064     std::istringstream iStr;
00065     boost::archive::text_iarchive ia (iStr);
00066     ia >> *this;
00067 }
00068
00069 // //////////////////////////////////////
00070 template<class Archive>
00071 void SegmentSnapshotTableKey::serialize (Archive& ioArchive,
00072     const unsigned int iFileVersion) {
00073     ioArchive & _tableID;
00074 }
00075
00076 // //////////////////////////////////////
00077 // Explicit template instantiation
00078 namespace ba = boost::archive;
00079 template void SegmentSnapshotTableKey::
00080     serialize<ba::text_oarchive> (ba::text_oarchive&, unsigned int);
00081 template void SegmentSnapshotTableKey::
00082     serialize<ba::text_iarchive> (ba::text_iarchive&, unsigned int);
00083 // //////////////////////////////////////
00084
00085 }

```

### 33.475 stdair/bom/SegmentSnapshotTableKey.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>

```

#### Classes

- struct [stdair::SegmentSnapshotTableKey](#)  
Key of a given guillotine block, made of a guillotine number.

#### Namespaces

- [boost](#)  
Forward declarations.
- [boost::serialization](#)
- [stdair](#)  
Handle on the StdAir library context.

### 33.476 SegmentSnapshotTableKey.hpp

```

00001 #ifndef __STDAIR_BOM_SEGMENTSNAPOSHOTTABLEKEY_HPP
00002 #define __STDAIR_BOM_SEGMENTSNAPOSHOTTABLEKEY_HPP
00003
00004 // //////////////////////////////////////

```

```

00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_basic_types.hpp>
00012 #include <stdair/bom/KeyAbstract.hpp>
00013
00015 namespace boost {
00016     namespace serialization {
00017         class access;
00018     }
00019 }
00020
00021 namespace stdair {
00022
00026     struct SegmentSnapshotTableKey : public KeyAbstract {
00027         friend class boost::serialization::access;
00028
00029         // ////////// Constructors and destructors //////////
00030     private:
00031         SegmentSnapshotTableKey();
00032
00033     public:
00034         SegmentSnapshotTableKey (const TableID_T&);
00035
00036         SegmentSnapshotTableKey (const
SegmentSnapshotTableKey&);
00037
00038         ~SegmentSnapshotTableKey();
00039
00040     public:
00041         // ////////// Getters //////////
00042         const TableID_T& getTableID() const {
00043             return _tableID;
00044         }
00045
00046     public:
00047         // ////////// Display support methods //////////
00048         void toStream (std::ostream& ioOut) const;
00049
00050         void fromStream (std::istream& ioIn);
00051
00052         const std::string toString() const;
00053
00054     public:
00055         // ////////// (Boost) Serialisation support methods //////////
00056         template<class Archive>
00057         void serialize (Archive& ar, const unsigned int iFileVersion);
00058
00059     private:
00060         void serialisationImplementationExport() const;
00061         void serialisationImplementationImport();
00062
00063     private:
00064         // ////////// Attributes //////////
00065         TableID_T _tableID;
00066     };
00067 }
00068
00069 #endif // __STDAIR_BOM_SEGMENTSNAPOSHOTTABLEKEY_HPP

```

### 33.477 stdair/bom/SegmentSnapshotTableTypes.hpp File Reference

```

#include <map>
#include <list>
#include <boost/multi_array.hpp>
#include <stdair/bom/key_types.hpp>

```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 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](#)

## 33.478 SegmentSnapshotTableTypes.hpp

```

00001 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00002 #ifndef __STDAIR_BOM_SEGMENTSNAPOSHOTTABLETYPES_HPP
00003 #define __STDAIR_BOM_SEGMENTSNAPOSHOTTABLETYPES_HPP
00004
00005 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00006 // Import section
00007 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // BOOST
00012 #include <boost/multi_array.hpp>
00013 // StdAir
00014 #include <stdair/bom/key_types.hpp>
00015
00016 namespace stdair {
00017
00018     // Forward declarations
00019     class SegmentSnapshotTable;
00020     class SegmentCabin;
00021
00022     ////////////////////////////////////////////////////////////////// Type definitions //////////////////////////////////////////////////////////////////
00023     typedef std::list<SegmentSnapshotTable*> SegmentSnapshotTableList_T;
00024
00025     typedef std::map<const MapKey_T, SegmentSnapshotTable*>
00026     SegmentSnapshotTableMap_T;
00027
00028     typedef std::map<const SegmentCabin*, SegmentDataID_T> SegmentCabinIndexMap_T;
00029
00030     typedef std::map<const MapKey_T, ClassIndex_T> ClassIndexMap_T;
00031
00032 }
00033 #endif // __STDAIR_BOM_SEGMENTSNAPOSHOTTABLETYPES_HPP
00034

```

## 33.479 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

- [stdair](#)

*Handle on the StdAir library context.*



### 33.480 SimpleNestingStructure.cpp

```

00001 ///////////////////////////////////////////////////////////////////
00002 // Import section
00003 ///////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <sstream>
00006 #include <cassert>
00007 #include <iomanip>
00008 #include <iostream>
00009 //STDAIR
00010 #include <stdair/stdair_exceptions.hpp>
00011 #include <stdair/basic/BasConst_Inventory.hpp>
00012 #include <stdair/bom/BomManager.hpp>
00013 #include <stdair/bom/BookingClass.hpp>
00014 #include <stdair/bom/BookingClass.hpp>
00015 #include <stdair/bom/SimpleNestingStructure.hpp>
00016 #include <stdair/bom/NestingNode.hpp>
00017 #include <stdair/bom/NestingNodeTypes.hpp>
00018 #include <stdair/service/Logger.hpp>
00019
00020 namespace stdair {
00021
00022 ///////////////////////////////////////////////////////////////////
00023 SimpleNestingStructure::SimpleNestingStructure () :
00024     _key (DEFAULT_NESTING_STRUCTURE_CODE), _parent (NULL) {
00025     assert (false);
00026 }
00027
00028 ///////////////////////////////////////////////////////////////////
00029 SimpleNestingStructure::
00030 SimpleNestingStructure (const SimpleNestingStructure& iSimpleNestingStructure)
00031 : _key (DEFAULT_NESTING_STRUCTURE_CODE), _parent (NULL) {
00032     assert (false);
00033 }
00034
00035 ///////////////////////////////////////////////////////////////////
00036 SimpleNestingStructure::SimpleNestingStructure (const Key_T& iKey)
00037 : _key (iKey), _parent (NULL) {
00038 }
00039
00040 ///////////////////////////////////////////////////////////////////
00041 SimpleNestingStructure::~SimpleNestingStructure() {
00042 }
00043
00044 ///////////////////////////////////////////////////////////////////
00045 // const bool SimpleNestingStructure::insertBookingClassList(const Yield_T& iYield,
00046 // const BookingClassList_T& iBookingClassList) {
00047 //     const bool isBookinClassListEmpty = iBookingClassList.empty();
00048 //     if (isBookinClassListEmpty == true) {
00049 //         std::ostringstream ostr;
00050 //         ostr << "The booking class list is empty and it should not be. "
00051 //             "No insertion done in the nesting structure (";
00052 //         toStream(ostr);
00053 //         ostr << ").";
00054 //         STDAIR_LOG_DEBUG(ostr.str());
00055 //         throw BookingClassListEmptyInNestingStructException(ostr.str());
00056 //     }
00057 //     assert(isBookinClassListEmpty == false);
00058 //     NestingNodeMap_T::iterator itNestingNode = _nestingNodeMap.find (iYield);
00059 //     bool insertionSucceeded = false;
00060 //     // Search a node with the same yield and add the
00061 //     // booking classes to the booking class list of the node.
00062 //     // If there is not a node with the same yield, create it.
00063 //     if (itNestingNode == _nestingNodeMap.end()) {
00064 //         NestingNode_T lNode (iYield, iBookingClassList);
00065 //         insertionSucceeded = _nestingNodeMap.insert(lNode).second;
00066 //     } else {
00067 //         NestingNode_T& lNode = *itNestingNode;
00068 //         const Yield_T& lYield = lNode.first;
00069 //         assert(lYield == iYield);
00070 //         BookingClassList_T& lBCList = lNode.second;
00071 //         for (BookingClassList_T::const_iterator itBC = iBookingClassList.begin();
00072 //             itBC != iBookingClassList.end(); ++itBC) {
00073 //             BookingClass* lBC_ptr = *itBC;
00074 //             assert(lBC_ptr != NULL);
00075 //             lBCList.push_back(lBC_ptr);
00076 //         }
00077 //         insertionSucceeded = true;
00078 //     }
00079
00080 //     return insertionSucceeded;
00081 // }
00082
00083 ///////////////////////////////////////////////////////////////////
00084 // const bool SimpleNestingStructure::
00085 // alreadyInNestingStructure(const ClassCode_T& iClassCode) const {

```

```

00086 // bool isAlreadyInTheMap = false;
00087 // NestingNodeMap_T::const_iterator itMap = _nestingNodeMap.begin();
00088 // for(; itMap != _nestingNodeMap.end(); ++itMap) {
00089 //     const NestingNode_T& lNestingNode = *itMap;
00090 //     const BookingClassList_T& lBCList = lNestingNode.second;
00091 //     BookingClassList_T::const_iterator itBC = lBCList.begin();
00092 //     for (;itBC != lBCList.end(); ++itBC) {
00093 //         BookingClass* lBC_ptr = *itBC;
00094 //         assert(lBC_ptr != NULL);
00095 //         const BookingClassKey& lBookingClassKey = lBC_ptr->getKey();
00096 //         const ClassCode_T& lClassCode = lBookingClassKey.getClassCode();
00097 //         if (lClassCode == iClassCode) {
00098 //             isAlreadyInTheMap = true;
00099 //             return isAlreadyInTheMap;
00100 //         }
00101 //     }
00102 // }
00103 // return isAlreadyInTheMap;
00104 // }
00105
00106 // //////////////////////////////////////
00107 std::string SimpleNestingStructure::toString () const {
00108     std::ostringstream oStr;
00109     oStr << describeKey();
00110
00111     return oStr.str();
00112 }
00113
00114 // //////////////////////////////////////
00115 const NestingNodeList_T&
SimpleNestingStructure::getNestingNodeList () const {
00116     return BomManager::getList<NestingNode> (*this);
00117 }
00118
00119 }

```

### 33.481 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>

```

#### Classes

- class [stdair::SimpleNestingStructure](#)

#### Namespaces

- [boost](#)  
Forward declarations.
- [boost::serialization](#)
- [stdair](#)  
Handle on the StdAir library context.

### 33.482 SimpleNestingStructure.hpp

```

00001 #ifndef __STDAIR_BOM_SIMPLENESTINGSTRUCTURE_HPP
00002 #define __STDAIR_BOM_SIMPLENESTINGSTRUCTURE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009 #include <stdair/bom/BomAbstract.hpp>
00010 #include <stdair/bom/NestingNodeTypes.hpp>

```

```

00011 #include <stdair/bom/SimpleNestingStructureTypes.hpp>
00012 #include <stdair/bom/NestingStructureKey.hpp>
00013
00015 namespace boost {
00016     namespace serialization {
00017         class access;
00018     }
00019 }
00020
00021 namespace stdair {
00022
00026     class SimpleNestingStructure : public BomAbstract {
00027     template <typename BOM> friend class FacBom;
00028     friend class FacBomManager;
00029     friend class boost::serialization::access;
00030
00031     public:
00032         // /////////// Type definitions ///////////
00036         typedef NestingStructureKey Key_T;
00037
00038     public:
00039         // /////////// Getters ///////////
00041         const Key_T& getKey() const {
00042             return _key;
00043         }
00044
00046         BomAbstract* const getParent() const {
00047             return _parent;
00048         }
00049
00053         const HolderMap_T& getHolderMap() const {
00054             return _holderMap;
00055         }
00056
00060         const NestingNodeList_T& getNestingNodeList() const;
00061
00062     public:
00063         // /////////// Display support methods ///////////
00069         void toStream (std::ostream& ioOut) const {
00070             ioOut << toString();
00071         }
00072
00078         void fromStream (std::istream& ioIn) {
00079         }
00080
00084         std::string toString() const;
00085
00089         const std::string describeKey() const {
00090             return _key.toString();
00091         }
00092
00093     public:
00094         // /////////// (Boost) Serialisation support methods ///////////
00099         template<class Archive>
00100         void serialize (Archive& ar, const unsigned int iFileVersion);
00101
00102     private:
00110         void serialisationImplementationExport() const;
00111         void serialisationImplementationImport();
00112
00113     public:
00114         // /////////// Constructors and destructor. ///////////
00119         SimpleNestingStructure (const Key_T&);
00120
00124         virtual ~SimpleNestingStructure();
00125
00126     private:
00130         SimpleNestingStructure();
00131
00135         SimpleNestingStructure (const SimpleNestingStructure&);
00136
00137     private:
00141         Key_T _key;
00142
00146         BomAbstract* _parent;
00147
00151         HolderMap_T _holderMap;
00152     };
00153 }
00154 #endif // __STDAIR_BOM_SIMPLENESTINGSTRUCTURE_HPP

```

**33.483 stdair/bom/SimpleNestingStructureTypes.hpp File Reference**

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

**Namespaces**

- [stdair](#)  
*Handle on the StdAir library context.*

**Typedefs**

- typedef std::list< SimpleNestingStructure \* > [stdair::SimpleNestingStructureList\\_T](#)
- typedef std::map< const MapKey\_T, SimpleNestingStructure \* > [stdair::SimpleNestingStructureMap\\_T](#)

**33.484 SimpleNestingStructureTypes.hpp**

```
00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_SIMPLENESTINGSTRUCTURETYPES_HPP
00003 #define __STDAIR_BOM_SIMPLENESTINGSTRUCTURETYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // StdAir
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016     // Forward declarations.
00017     class SimpleNestingStructure;
00018
00019     typedef std::list<SimpleNestingStructure*> SimpleNestingStructureList_T;
00020
00021     typedef std::map<const MapKey_T, SimpleNestingStructure*>
00022     SimpleNestingStructureMap_T;
00023
00024 }
00025
00026 #endif // __STDAIR_BOM_SIMPLENESTINGSTRUCTURETYPES_HPP
```

**33.485 stdair/bom/SnapshotStruct.cpp File Reference**

```
#include <cassert>
#include <sstream>
#include <stdair/bom/SnapshotStruct.hpp>
```

**Namespaces**

- [stdair](#)  
*Handle on the StdAir library context.*

**33.486 SnapshotStruct.cpp**

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
```

```

00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/bom/SnapshotStruct.hpp>
00009
00010 namespace stdair {
00011
00012 // //////////////////////////////////////
00013 SnapshotStruct::SnapshotStruct() {
00014     assert (false);
00015 }
00016
00017 // //////////////////////////////////////
00018 SnapshotStruct::
00019 SnapshotStruct (const SnapshotStruct& iSnapshot)
00020     : _airlineCode (iSnapshot._airlineCode),
00021       _snapshotTime (iSnapshot._snapshotTime) {
00022 }
00023
00024 // //////////////////////////////////////
00025 SnapshotStruct::
00026 SnapshotStruct (const AirlineCode_T& iAirlineCode,
00027                const DateTime_T& iSnapshotTime)
00028     : _airlineCode (iAirlineCode), _snapshotTime (iSnapshotTime) {
00029 }
00030
00031 // //////////////////////////////////////
00032 SnapshotStruct::~SnapshotStruct() {
00033 }
00034
00035 // //////////////////////////////////////
00036 void SnapshotStruct::toStream (std::ostream& ioOut) const {
00037     ioOut << describe();
00038 }
00039
00040 // //////////////////////////////////////
00041 void SnapshotStruct::fromStream (std::istream& ioIn) {
00042 }
00043
00044 // //////////////////////////////////////
00045 const std::string SnapshotStruct::describe() const {
00046     std::ostringstream oStr;
00047     oStr << _airlineCode << ", " << _snapshotTime;
00048     return oStr.str();
00049 }
00050
00051 }

```

### 33.487 stdair/bom/SnapshotStruct.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/SnapshotTypes.hpp>

```

#### Classes

- struct [stdair::SnapshotStruct](#)

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.488 SnapshotStruct.hpp

```

00001 #ifndef __STDAIR_BOM_SNAPSHOTSTRUCT_HPP

```

```

00002 #define __STDAIR_BOM_SNAPSHOTSTRUCT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/stdair_demand_types.hpp>
00013 #include <stdair/basic/StructAbstract.hpp>
00014 #include <stdair/bom/SnapshotTypes.hpp>
00015
00016 namespace stdair {
00017
00019     struct SnapshotStruct : public StructAbstract {
00020     public:
00021         // ////////////////////////////////////// Getters //////////////////////////////////////
00023         const AirlineCode_T& getAirlineCode() const {
00024             return _airlineCode;
00025         }
00026
00028         const DateTime_T& getSnapshotTime() const {
00029             return _snapshotTime;
00030         }
00031
00032         // ////////////////////////////////////// Display support method //////////////////////////////////////
00033         void toStream (std::ostream& ioOut) const;
00036
00039         void fromStream (std::istream& ioIn);
00040
00042         const std::string describe() const;
00043
00044         // ////////////////////////////////////// Constructors and Destructors //////////////////////////////////////
00046     public:
00048         SnapshotStruct (const AirlineCode_T&, const
00049             DateTime_T&);
00051         SnapshotStruct (const SnapshotStruct&);
00052
00053     private:
00056         SnapshotStruct ();
00057
00058     public:
00060         ~SnapshotStruct ();
00061
00062
00063     private:
00064         // ////////////////////////////////////// Attributes //////////////////////////////////////
00066         const AirlineCode_T _airlineCode;
00067
00069         const DateTime_T _snapshotTime;
00070     };
00071
00072 }
00073 #endif // __STDAIR_BOM_SNAPSHOTSTRUCT_HPP

```

### 33.489 stdair/bom/SnapshotTypes.hpp File Reference

```
#include <boost/shared_ptr.hpp>
```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

#### Typedefs

- typedef boost::shared\_ptr< SnapshotStruct > [stdair::SnapshotPtr\\_T](#)

### 33.490 SnapshotTypes.hpp

```

00001 ///////////////////////////////////////////////////////////////////
00002 #ifndef __STDAIR_BOM_SNAPSHOTTYPES_HPP
00003 #define __STDAIR_BOM_SNAPSHOTTYPES_HPP
00004
00005 ///////////////////////////////////////////////////////////////////
00006 // Import section
00007 ///////////////////////////////////////////////////////////////////
00008 // Boost
00009 #include <boost/shared_ptr.hpp>
00010
00011 namespace stdair {
00012
00013     // Forward declarations
00014     struct SnapshotStruct;
00015
00016     /////////////////////////////////////////////////////////////////// Type definitions ///////////////////////////////////////////////////////////////////
00018     typedef boost::shared_ptr<SnapshotStruct> SnapshotPtr_T;
00019
00020 }
00021 #endif // __STDAIR_BOM_SNAPSHOTTYPES_HPP
00022

```

### 33.491 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

- [stdair](#)

*Handle on the StdAir library context.*

### 33.492 TimePeriod.cpp

```

00001 ///////////////////////////////////////////////////////////////////
00002 // Import section
00003 ///////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_General.hpp>
00009 #include <stdair/service/Logger.hpp>
00010 #include <stdair/bom/TimePeriod.hpp>
00011
00012 namespace stdair {
00013
00014     ///////////////////////////////////////////////////////////////////
00015     TimePeriod::TimePeriod()
00016     : _key (DEFAULT_EPSILON_DURATION,
00017     DEFAULT_EPSILON_DURATION),
00018     _parent (NULL) {
00019         // That constructor is used by the serialisation process
00020     }
00021
00022     ///////////////////////////////////////////////////////////////////
00023     TimePeriod::TimePeriod (const TimePeriod& iTimePeriod)
00024     : _key (iTimePeriod.getKey()), _parent (NULL) {
00025     }
00026
00027     ///////////////////////////////////////////////////////////////////
00028     TimePeriod::TimePeriod (const Key_T& iKey)
00029     : _key (iKey), _parent (NULL) {
00030     }
00031
00032     ///////////////////////////////////////////////////////////////////
00033     TimePeriod::~TimePeriod () {
00034     }
00035

```

```

00034
00035 ///////////////////////////////////////////////////////////////////
00036 std::string TimePeriod::toString() const {
00037     std::ostringstream oStr;
00038     oStr << describeKey();
00039     return oStr.str();
00040 }
00041
00042 ///////////////////////////////////////////////////////////////////
00043 bool TimePeriod::
00044 isDepartureTimeValid (const Time_T& iFlightTime) const {
00045
00046     const Time_T& lTimeRangeStart = getTimeRangeStart();
00047     const Time_T& lTimeRangeEnd = getTimeRangeEnd();
00048
00049     // Check if the departure time is within the time range.
00050     if (lTimeRangeStart >= iFlightTime) {
00051         // DEBUG
00052         STDAIR_LOG_DEBUG ("Time range begin: " << lTimeRangeStart << ", "
00053                         << "time: " << iFlightTime);
00054         return false;
00055     }
00056     if (lTimeRangeEnd <= iFlightTime) {
00057         // DEBUG
00058         STDAIR_LOG_DEBUG ("Time range end: " << lTimeRangeEnd << ", "
00059                         << "time: " << iFlightTime);
00060         return false;
00061     }
00062     return true;
00063 }
00064 }
00065 }
00066 }
00067

```

### 33.493 stdair/bom/TimePeriod.hpp File Reference

```

#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/TimePeriodKey.hpp>
#include <stdair/bom/TimePeriodTypes.hpp>

```

#### Classes

- class [stdair::TimePeriod](#)  
Class representing the actual attributes for a fare time-period.

#### Namespaces

- [stdair](#)  
Handle on the StdAir library context.

### 33.494 TimePeriod.hpp

```

00001 #ifndef __STDAIR_BOM_FARETIMEPERIOD_HPP
00002 #define __STDAIR_BOM_FARETIMEPERIOD_HPP
00003
00004 ///////////////////////////////////////////////////////////////////
00005 // Import section
00006 ///////////////////////////////////////////////////////////////////
00007 // STDAIR
00008 #include <stdair/bom/BomAbstract.hpp>
00009 #include <stdair/bom/TimePeriodKey.hpp>
00010 #include <stdair/bom/TimePeriodTypes.hpp>
00011
00012 // Forward declaration
00013 namespace stdair {
00014
00015     class TimePeriod : public BomAbstract {
00016     template <typename BOM> friend class FacBom;
00017     template <typename BOM> friend class FacCloneBom;
00018     friend class FacBomManager;
00019

```



```

00022
00023 public:
00024 // //////////// Type definitions ////////////
00028 typedef TimePeriodKey Key_T;
00029
00030 public:
00031 // //////////// Display support methods ////////////
00032 // //////////// Display support methods ////////////
00038 void toStream (std::ostream& ioOut) const {
00039     ioOut << toString();
00040 }
00041
00047 void fromStream (std::istream& ioIn) {
00048 }
00049
00053 std::string toString() const;
00054
00058 const std::string describeKey() const {
00059     return _key.toString();
00060 }
00061
00062 public:
00063 // //////////// Getters ////////////
00067 const Key_T& getKey() const {
00068     return _key;
00069 }
00070
00074 BomAbstract* const getParent() const {
00075     return _parent;
00076 }
00077
00081 const HolderMap_T& getHolderMap() const {
00082     return _holderMap;
00083 }
00084
00088 const Time_T& getTimeRangeStart() const {
00089     return _key.getTimeRangeStart();
00090 }
00091
00095 const Time_T& getTimeRangeEnd() const {
00096     return _key.getTimeRangeEnd();
00097 }
00098
00099 public:
00100 // //////////// Business methods ////////////
00105 bool isDepartureTimeValid (const Time_T&) const;
00106
00107 protected:
00108 // //////////// Constructors and destructors ////////////
00112 TimePeriod (const Key_T&);
00116 virtual ~TimePeriod();
00117
00118 private:
00122 TimePeriod();
00126 TimePeriod (const TimePeriod&);
00127
00128 protected:
00129 // //////////// Attributes ////////////
00133 Key_T _key;
00134
00138 BomAbstract* _parent;
00139
00143 HolderMap_T _holderMap;
00144
00145 };
00146
00147 }
00148 #endif // __STDAIR_BOM_FARETIMEPERIOD_HPP
00149

```

### 33.495 stdair/bom/TimePeriodKey.cpp File Reference

```

#include <ostream>
#include <sstream>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/TimePeriodKey.hpp>

```

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.496 TimePeriodKey.cpp

```

00001 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00002 // Import section
00003 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <ostream>
00006 #include <sstream>
00007 // STDAIR
00008 #include <stdair/basic/BasConst_General.hpp>
00009 #include <stdair/bom/TimePeriodKey.hpp>
00010
00011 namespace stdair {
00012
00013 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00014 TimePeriodKey::TimePeriodKey ()
00015 : _timeRangeStart (DEFAULT_EPSILON_DURATION),
00016   _timeRangeEnd (DEFAULT_EPSILON_DURATION) {
00017     assert (false);
00018 }
00019
00020 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00021 TimePeriodKey::TimePeriodKey (const Time_T& iTimeRangeStart,
00022                               const Time_T& iTimeRangeEnd)
00023 : _timeRangeStart(iTimeRangeStart),
00024   _timeRangeEnd(iTimeRangeEnd) {
00025 }
00026
00027 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00028 TimePeriodKey::TimePeriodKey (const TimePeriodKey& iKey)
00029 : _timeRangeStart(iKey.getTimeRangeStart()),
00030   _timeRangeEnd(iKey.getTimeRangeEnd()) {
00031 }
00032
00033 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00034 TimePeriodKey::~TimePeriodKey () {
00035 }
00036
00037 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00038 void TimePeriodKey::toStream (std::ostream& ioOut) const {
00039     ioOut << "TimePeriodKey: " << toString() << std::endl;
00040 }
00041
00042 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00043 void TimePeriodKey::fromStream (std::istream& ioIn) {
00044 }
00045
00046 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00047 const std::string TimePeriodKey::toString() const {
00048     std::ostringstream oStr;
00049     oStr << _timeRangeStart << "-" << _timeRangeEnd;
00050     return oStr.str();
00051 }
00052
00053 }

```

## 33.497 stdair/bom/TimePeriodKey.hpp File Reference

```

#include <stdair/bom/KeyAbstract.hpp>
#include <stdair/stdair_date_time_types.hpp>

```

## Classes

- struct [stdair::TimePeriodKey](#)

*Key of time-period.*

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.498 TimePeriodKey.hpp

```

00001 #ifndef __STDAIR_BOM_TIMEPERIODKEY_HPP
00002 #define __STDAIR_BOM_TIMEPERIODKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STDAIR
00008 #include <stdair/bom/KeyAbstract.hpp>
00009 #include <stdair/stdair_date_time_types.hpp>
00010
00011 namespace stdair {
00015     struct TimePeriodKey : public KeyAbstract {
00016
00017     public:
00018         // ////////////////////////////////// Construction //////////////////////////////////
00019         TimePeriodKey (const Time_T&,
00020                       const Time_T&);
00021         TimePeriodKey (const TimePeriodKey&);
00022         ~TimePeriodKey ();
00023     private:
00024         TimePeriodKey ();
00025
00026     public:
00027         // ////////////////////////////////// Getter //////////////////////////////////
00028         const Time_T& getTimeRangeStart() const {
00029             return _timeRangeStart;
00030         }
00031
00032         const Time_T& getTimeRangeEnd() const {
00033             return _timeRangeEnd;
00034         }
00035
00036         // ////////////////////////////////// Display support methods //////////////////////////////////
00037         void toStream (std::ostream& ioOut) const;
00038
00039         void fromStream (std::istream& ioIn);
00040
00041         const std::string toString() const;
00042     private:
00043         // ////////////////////////////////// Attributes //////////////////////////////////
00044         Time_T _timeRangeStart;
00045
00046         Time_T _timeRangeEnd;
00047     };
00048 }
00049 #endif // __STDAIR_BOM_TIMEPERIODKEY_HPP

```

### 33.499 stdair/bom/TimePeriodTypes.hpp File Reference

```

#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>

```

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## Typedefs

- typedef std::list< TimePeriod \* > [stdair::TimePeriodList\\_T](#)

- typedef std::map< const MapKey\_T, TimePeriod \* > [stdair::TimePeriodMap\\_T](#)
- typedef std::pair< MapKey\_T, TimePeriod \* > [stdair::TimePeriodWithKey\\_T](#)
- typedef std::list< TimePeriodWithKey\_T > [stdair::TimePeriodDetailedList\\_T](#)

### 33.500 TimePeriodTypes.hpp

```

00001 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00002 #ifndef __STDAIR_BOM_TIMEPERIODTYPES_HPP
00003 #define __STDAIR_BOM_TIMEPERIODTYPES_HPP
00004
00005 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00006 // Import section
00007 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // STDAIR
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016     // Forward declarations.
00017     class TimePeriod;
00018
00019     typedef std::list<TimePeriod*> TimePeriodList_T;
00020
00021
00022     typedef std::map<const MapKey_T, TimePeriod*> TimePeriodMap_T;
00023
00024
00025     typedef std::pair<MapKey_T, TimePeriod*> TimePeriodWithKey_T;
00026     typedef std::list<TimePeriodWithKey_T> TimePeriodDetailedList_T;
00027 }
00028
00029 #endif // __STDAIR_BOM_TIMEPERIODTYPES_HPP
00030

```

### 33.501 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

- [stdair](#)  
*Handle on the StdAir library context.*

### 33.502 TravelSolutionStruct.cpp

```

00001 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00002 // Import section
00003 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_BookingClass.hpp>
00009 #include <stdair/bom/TravelSolutionStruct.hpp>
00010 #include <stdair/bom/BomKeyManager.hpp>
00011 #include <stdair/bom/ParsedKey.hpp>
00012
00013 namespace stdair {
00014
00015     //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00016     TravelSolutionStruct::TravelSolutionStruct() :
00017         _chosenFareOption (NULL) {
00018     }
00019
00020 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

```

```

00019 TravelSolutionStruct::~TravelSolutionStruct() {
00020 }
00021
00022 // //////////////////////////////////////
00023 void TravelSolutionStruct::toStream (std::ostream& ioOut) const {
00024     ioOut << describe();
00025 }
00026
00027 // //////////////////////////////////////
00028 void TravelSolutionStruct::fromStream (std::istream& ioIn) {
00029 }
00030
00031 // //////////////////////////////////////
00032 const std::string TravelSolutionStruct::describeSegmentPath()
const {
00033     std::ostringstream oStr;
00034
00035     //
00036     oStr << "Segment path: ";
00037     unsigned short idx = 0;
00038     for (SegmentPath_T::const_iterator lItSegmentPath = _segmentPath.begin();
00039         lItSegmentPath != _segmentPath.end(); ++lItSegmentPath, ++idx) {
00040         if (idx != 0) {
00041             oStr << " - ";
00042         }
00043         const std::string& lSegmentPathString = *lItSegmentPath;
00044         const stdair::ParsedKey& lSegmentParsedKey =
00045             stdair::BomKeyManager::extractKeys (lSegmentPathString);
00046         const std::string& lSegmentKey = lSegmentParsedKey.toString();
00047         oStr << lSegmentKey;
00048     }
00049     return oStr.str();
00050 }
00051
00052 // //////////////////////////////////////
00053 const std::string TravelSolutionStruct::describe() const {
00054     std::ostringstream oStr;
00055
00056     //
00057     oStr << "Segment path: ";
00058     unsigned short idx = 0;
00059     for (SegmentPath_T::const_iterator lItSegmentPath = _segmentPath.begin();
00060         lItSegmentPath != _segmentPath.end(); ++lItSegmentPath, ++idx) {
00061         if (idx != 0) {
00062             oStr << "-";
00063         }
00064         const std::string& lSegmentPathString = *lItSegmentPath;
00065         const stdair::ParsedKey& lSegmentParsedKey =
00066             stdair::BomKeyManager::extractKeys (lSegmentPathString);
00067         const std::string& lSegmentKey = lSegmentParsedKey.toString();
00068         oStr << lSegmentKey;
00069     }
00070     oStr << " ### ";
00071
00072     //
00073     if (_chosenFareOption != NULL) {
00074         oStr << "Chosen fare option: " << _chosenFareOption->describe()
00075             << " ## Among: ";
00076     } else {
00077         oStr << "Fare options: ";
00078     }
00079
00080     //
00081     idx = 0;
00082     for (FareOptionList_T::const_iterator lItFareOption= _fareOptionList.begin();
00083         lItFareOption != _fareOptionList.end(); ++lItFareOption, ++idx) {
00084         if (idx != 0) {
00085             oStr << " , ";
00086         }
00087         const FareOptionStruct& lFareOption = *lItFareOption;
00088         oStr << lFareOption.describe();
00089     }
00090
00091     return oStr.str();
00092 }
00093
00094 // //////////////////////////////////////
00095 const std::string TravelSolutionStruct::display() const {
00096     std::ostringstream oStr;
00097
00098     // List of segment keys (one per segment)
00099     unsigned short idx = 0;
00100     for (SegmentPath_T::const_iterator itSegPath = _segmentPath.begin();
00101         itSegPath != _segmentPath.end(); ++itSegPath, ++idx) {
00102         if (idx != 0) {
00103             oStr << " ; ";
00104         }

```

```

00105     const std::string& lSegmentPathString = *itSegPath;
00106     const stdair::ParsedKey& lSegmentParsedKey =
00107         stdair::BomKeyManager::extractKeys (lSegmentPathString);
00108     const std::string& lSegmentKey = lSegmentParsedKey.toString();
00109     oStr << "[" << idx << "]" << lSegmentKey;
00110 }
00111
00112 // List of fare options (for the whole O&D)
00113 oStr << " --- ";
00114 idx = 0;
00115 for (FareOptionList_T::const_iterator itFareOption = _fareOptionList.begin();
00116      itFareOption != _fareOptionList.end(); ++itFareOption, ++idx) {
00117     if (idx != 0) {
00118         oStr << " , ";
00119     }
00120     const FareOptionStruct& lFareOption = *itFareOption;
00121     oStr << lFareOption.display();
00122 }
00123
00124 // List of booking class availability maps: one map per segment
00125 oStr << " --- ";
00126 idx = 0;
00127 for (ClassAvailabilityMapHolder_T::const_iterator itSegMap =
00128      _classAvailabilityMapHolder.begin();
00129      itSegMap != _classAvailabilityMapHolder.end(); ++itSegMap, ++idx) {
00130     if (idx != 0) {
00131         oStr << " ; ";
00132     }
00133     // Retrieve the booking class availability map
00134     const ClassAvailabilityMap_T& lClassAvlMap = *itSegMap;
00135     oStr << "[" << idx << "]" ";";
00136
00137     // List (map) of booking class availabilities
00138     unsigned short jdx = 0;
00139     for (ClassAvailabilityMap_T::const_iterator itClass = lClassAvlMap.begin();
00140          itClass != lClassAvlMap.end(); ++itClass, ++jdx) {
00141         if (jdx != 0) {
00142             oStr << " ";
00143         }
00144         const ClassCode_T& lClassCode = itClass->first;
00145         const Availability_T& lAvl = itClass->second;
00146         oStr << lClassCode << ":" << lAvl;
00147     }
00148 }
00149
00150 return oStr.str();
00151 }
00152
00153 // //////////////////////////////////////
00154 void TravelSolutionStruct::addSegment (const std::string& iKey) {
00155     _segmentPath.push_back (iKey);
00156 }
00157
00158 // //////////////////////////////////////
00159 void TravelSolutionStruct::
00160 addClassAvailabilityMap (const ClassAvailabilityMap_T&
00161 iMap) {
00162     _classAvailabilityMapHolder.push_back (iMap);
00163 }
00164 // //////////////////////////////////////
00165 void TravelSolutionStruct::
00166 addClassObjectIDMap (const ClassObjectIDMap_T& iMap) {
00167     _classObjectIDMapHolder.push_back (iMap);
00168 }
00169
00170 // //////////////////////////////////////
00171 void TravelSolutionStruct::
00172 addClassYieldMap (const ClassYieldMap_T& iMap) {
00173     _classYieldMapHolder.push_back (iMap);
00174 }
00175
00176 // //////////////////////////////////////
00177 void TravelSolutionStruct::
00178 addBidPriceVector (const BidPriceVector_T& iBpv) {
00179     _bidPriceVectorHolder.push_back (iBpv);
00180 }
00181
00182 // //////////////////////////////////////
00183 void TravelSolutionStruct::
00184 addClassBpvMap (const ClassBpvMap_T& iMap) {
00185     _classBpvMapHolder.push_back (iMap);
00186 }
00187
00188 // //////////////////////////////////////
00189 void TravelSolutionStruct::
00190 addFareOption (const FareOptionStruct& iFareOption) {

```

```

00191     _fareOptionList.push_back (iFareOption);
00192 }
00193
00194 }

```

### 33.503 stdair/bom/TravelSolutionStruct.hpp File Reference

```

#include <iosfwd>
#include <string>
#include <vector>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/BookingClassTypes.hpp>
#include <stdair/bom/FareOptionStruct.hpp>
#include <stdair/bom/FareOptionTypes.hpp>
#include <stdair/bom/TravelSolutionTypes.hpp>

```

#### Classes

- struct [stdair::TravelSolutionStruct](#)  
*Structure holding the elements of a travel solution.*

#### Namespaces

- [stdair](#)  
*Handle on the StdAir library context.*

### 33.504 TravelSolutionStruct.hpp

```

00001 #ifndef __STDAIR_BOM_TRAVELSOLUTIONSTRUCT_HPP
00002 #define __STDAIR_BOM_TRAVELSOLUTIONSTRUCT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 #include <vector>
00011 // StdAir
00012 #include <stdair/stdair_basic_types.hpp>
00013 #include <stdair/basic/StructAbstract.hpp>
00014 #include <stdair/bom/BookingClassTypes.hpp>
00015 #include <stdair/bom/FareOptionStruct.hpp>
00016 #include <stdair/bom/FareOptionTypes.hpp>
00017 #include <stdair/bom/TravelSolutionTypes.hpp>
00018
00019 namespace stdair {
00020
00024     struct TravelSolutionStruct : public StructAbstract {
00025     public:
00026         // ////////////////////////////////// Getters //////////////////////////////////
00028         const SegmentPath_T& getSegmentPath() const {
00029             return _segmentPath;
00030         }
00031
00033         const ClassAvailabilityMapHolder_T&
00034         getClassAvailabilityMapHolder() const {
00035             return _classAvailabilityMapHolder;
00036         }
00038         const ClassObjectIDMapHolder_T&
00039         getClassObjectIDMapHolder() const {
00040             return _classObjectIDMapHolder;
00041         }
00043         const ClassYieldMapHolder_T& getClassYieldMapHolder() const
00044     {

```

```

00044         return _classYieldMapHolder;
00045     }
00046
00048     const BidPriceVectorHolder_T& getBidPriceVectorHolder()
const {
00049         return _bidPriceVectorHolder;
00050     }
00051
00053     const ClassBpvMapHolder_T& getClassBpvMapHolder() const {
00054         return _classBpvMapHolder;
00055     }
00056
00058     const FareOptionList_T& getFareOptionList() const {
00059         return _fareOptionList;
00060     }
00061
00063     FareOptionList_T& getFareOptionListRef() {
00064         return _fareOptionList;
00065     }
00066
00068     const FareOptionStruct& getChosenFareOption() const {
00069         assert (_chosenFareOption != NULL);
00070         return *_chosenFareOption;
00071     }
00072
00073 public:
00074     // //////////// Setters ////////////
00076     void addSegment (const std::string&);
00077
00079     void addClassAvailabilityMap (const
ClassAvailabilityMap_T&);
00080
00082     void addClassObjectIDMap (const ClassObjectIDMap_T&);
00083
00085     void addClassYieldMap (const ClassYieldMap_T&);
00086
00088     void addBidPriceVector (const BidPriceVector_T&);
00089
00091     void addClassBpvMap (const ClassBpvMap_T&);
00092
00094     void addFareOption (const FareOptionStruct&);
00095
00097     void setChosenFareOption (const FareOptionStruct& iChosenFO) {
00098         _chosenFareOption = &iChosenFO;
00099     }
00100
00101
00102 public:
00103     // //////////// Display support method ////////////
00109     void toStream (std::ostream& ioOut) const;
00110
00115     void fromStream (std::istream& ioIn);
00116
00120     const std::string describe() const;
00121
00125     const std::string display() const;
00126
00130     const std::string describeSegmentPath() const;
00131
00132
00133 public:
00134     // //////////// Constructors & Destructor ////////////
00138     TravelSolutionStruct();
00139
00143     ~TravelSolutionStruct();
00144
00145
00146 private:
00147     // //////////// Attributes ////////////
00151     SegmentPath_T _segmentPath;
00152
00156     ClassAvailabilityMapHolder_T _classAvailabilityMapHolder;
00157
00161     ClassObjectIDMapHolder_T _classObjectIDMapHolder;
00162
00166     ClassYieldMapHolder_T _classYieldMapHolder;
00167
00171     BidPriceVectorHolder_T _bidPriceVectorHolder;
00172
00176     ClassBpvMapHolder_T _classBpvMapHolder;
00177
00181     FareOptionList_T _fareOptionList;
00182
00186     const FareOptionStruct* _chosenFareOption;
00187 };
00188
00189 }

```



```
00190 #endif // __STDAIR_BOM_TRAVELSOLUTIONSTRUCT_HPP
```

### 33.505 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

- [stdair](#)  
*Handle on the StdAir library context.*

#### Typedefs

- typedef std::list< TravelSolutionStruct > [stdair::TravelSolutionList\\_T](#)
- typedef KeyList\_T [stdair::SegmentPath\\_T](#)
- typedef std::list< SegmentPath\_T > [stdair::SegmentPathList\\_T](#)
- typedef std::map< const ClassCode\_T, Availability\_T > [stdair::ClassAvailabilityMap\\_T](#)
- typedef std::list< ClassAvailabilityMap\_T > [stdair::ClassAvailabilityMapHolder\\_T](#)
- typedef std::map< const ClassCode\_T, 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](#)

### 33.506 TravelSolutionTypes.hpp

```
00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_TRAVELSOLUTIONTYPES_HPP
00003 #define __STDAIR_BOM_TRAVELSOLUTIONTYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <list>
00010 #include <map>
00011 // StdAir
00012 #include <stdair/stdair_basic_types.hpp>
00013 #include <stdair/bom/key_types.hpp>
00014 #include <stdair/stdair_inventory_types.hpp> // bid price related types.
00015 #include <stdair/bom/BomIDTypes.hpp>
00016
00017 namespace stdair {
00018
00019     // Forward declarations.
00020     struct TravelSolutionStruct;
00021
00022     typedef std::list<TravelSolutionStruct> TravelSolutionList_T;
00023
00024     typedef KeyList_T SegmentPath_T;
00025
00026     typedef std::list<SegmentPath_T> SegmentPathList_T;
00027
00028     typedef std::map<const ClassCode_T, Availability_T> ClassAvailabilityMap_T;
00029
00030     typedef std::list<ClassAvailabilityMap_T> ClassAvailabilityMapHolder_T;
00031
00032     typedef std::map<const ClassCode_T, BookingClassID_T> ClassObjectIDMap_T;
00033
00034     typedef std::list<ClassObjectIDMap_T> ClassObjectIDMapHolder_T;
00035
00036     typedef std::map<const ClassCode_T, YieldValue_T> ClassYieldMap_T;
00037
00038     typedef std::list<ClassYieldMap_T> ClassYieldMapHolder_T;
00039
00040     typedef std::list<BidPriceVector_T> BidPriceVectorHolder_T;
00041
00042     typedef std::map<const ClassCode_T, const BidPriceVector_T * > ClassBpvMap_T;
00043
00044     typedef std::list<ClassBpvMap_T> ClassBpvMapHolder_T;
00045
00046 }
```

```

00036
00038     typedef std::map<const ClassCode_T, BookingClassID_T> ClassObjectIDMap_T;
00039
00041     typedef std::list<ClassObjectIDMap_T> ClassObjectIDMapHolder_T;
00042
00044     typedef std::map<const ClassCode_T, YieldValue_T> ClassYieldMap_T;
00045
00047     typedef std::list<ClassYieldMap_T> ClassYieldMapHolder_T;
00048
00050     typedef std::list<BidPriceVector_T> BidPriceVectorHolder_T;
00051
00053     typedef std::map<const ClassCode_T, const BidPriceVector_T*> ClassBpvMap_T;
00054
00056     typedef std::list<ClassBpvMap_T> ClassBpvMapHolder_T;
00057 }
00058 #endif // __STDAIR_BOM_TRAVELSOLUTIONTYPES_HPP
00059

```

### 33.507 stdair/bom/VirtualClassStruct.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/bom/VirtualClassStruct.hpp>
#include <stdair/bom/BookingClass.hpp>

```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.508 VirtualClassStruct.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/bom/VirtualClassStruct.hpp>
00009 #include <stdair/bom/BookingClass.hpp>
00010
00011 namespace stdair {
00012
00013     // //////////////////////////////////////
00014     VirtualClassStruct::VirtualClassStruct() {
00015         assert (false);
00016     }
00017
00018     // //////////////////////////////////////
00019     VirtualClassStruct::VirtualClassStruct (const VirtualClassStruct& iVC)
00020         : _bookingClassList (iVC._bookingClassList), _yield (iVC._yield),
00021           _mean (iVC._mean), _stdDev (iVC._stdDev) {
00022     }
00023
00024     // //////////////////////////////////////
00025     VirtualClassStruct::
00026     VirtualClassStruct (const BookingClassList_T& ioBookingClassList) {
00027         _bookingClassList = ioBookingClassList;
00028     }
00029
00030     // //////////////////////////////////////
00031     VirtualClassStruct::~VirtualClassStruct() {
00032     }
00033
00034
00035     // //////////////////////////////////////
00036     void VirtualClassStruct::toStream (std::ostream& ioOut) const {
00037         ioOut << describe();
00038     }
00039
00040     // //////////////////////////////////////
00041     void VirtualClassStruct::fromStream (std::istream& ioIn) {
00042     }

```

```

00043
00044 // //////////////////////////////////////
00045 const std::string VirtualClassStruct::describe() const {
00046     std::ostringstream ostr;
00047     ostr << "Yield: " << _yield
00048     << ", Demand N (" << _mean << ", " << _stdDev << ")";
00049     return ostr.str();
00050 }
00051
00052 // //////////////////////////////////////
00053 const GeneratedDemandVector_T VirtualClassStruct::
00054 getGeneratedDemandVector() const {
00055     GeneratedDemandVector_T lDemandVector;
00056     const bool isBookingClassListEmpty = _bookingClassList.empty();
00057     if (isBookingClassListEmpty == false) {
00058         assert (isBookingClassListEmpty == false);
00059         BookingClassList_T::const_iterator itBC = _bookingClassList.begin();
00060         BookingClass* lBC_ptr = *itBC;
00061         const GeneratedDemandVector_T& lFirstDemandVector =
00062             lBC_ptr->getGeneratedDemandVector();
00063         const unsigned int lFirstDemandVectorSize = lFirstDemandVector.size();
00064         for (unsigned int i = 0; i < lFirstDemandVectorSize; ++i) {
00065             const double& lDemand = lFirstDemandVector[i];
00066             lDemandVector.push_back(lDemand);
00067         }
00068         const unsigned int& lDemandVectorSize = lDemandVector.size();
00069         ++itBC;
00070         for (; itBC != _bookingClassList.end(); ++ itBC) {
00071             lBC_ptr = *itBC;
00072             assert(lBC_ptr != NULL);
00073             const GeneratedDemandVector_T& lCurrentDemandVector =
00074                 lBC_ptr->getGeneratedDemandVector();
00075             const unsigned int& lCurrentDemandVectorSize =
00076                 lCurrentDemandVector.size();
00077             assert(lDemandVectorSize == lCurrentDemandVectorSize);
00078             for (unsigned int i = 0; i < lDemandVectorSize; ++i) {
00079                 lDemandVector[i] += lCurrentDemandVector[i];
00080             }
00081         }
00082     }
00083     return lDemandVector;
00084 }
00085 }

```

### 33.509 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>

```

#### Classes

- struct [stdair::VirtualClassStruct](#)

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.510 VirtualClassStruct.hpp

```
00001 #ifndef __STDAIR_BOM_VIRTUALCLASSSTRUCT_HPP
```

```

00002 #define __STDAIR_BOM_VIRTUALCLASSSTRUCT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 #include <vector>
00011 // StdAir
00012 #include <stdair/stdair_basic_types.hpp>
00013 #include <stdair/stdair_inventory_types.hpp>
00014 #include <stdair/stdair_maths_types.hpp>
00015 #include <stdair/stdair_rm_types.hpp>
00016 #include <stdair/basic/StructAbstract.hpp>
00017 #include <stdair/bom/BookingClassTypes.hpp>
00018
00019 namespace stdair {
00020     // Forward declarations.
00021     class BookingClass;
00022
00023     struct VirtualClassStruct : public StructAbstract {
00024     public:
00025         // ////////////////////////////////// Getters //////////////////////////////////
00026         const BookingClassList_T& getBookingClassList() const {
00027             return _bookingClassList;
00028         }
00029
00030         const Yield_T& getYield() const {
00031             return _yield;
00032         }
00033
00034         const MeanValue_T& getMean() const {
00035             return _mean;
00036         }
00037
00038         const StdDevValue_T& getStdDev() const {
00039             return _stdDev;
00040         }
00041
00042         const BookingLimit_T& getCumulatedBookingLimit () const {
00043             return _cumulatedBookingLimit;
00044         }
00045
00046         const ProtectionLevel_T& getCumulatedProtection () const {
00047             return _cumulatedProtection;
00048         }
00049
00050         const GeneratedDemandVector_T
00051         getGeneratedDemandVector () const;
00052
00053     public:
00054         // ////////////////////////////////// Setters //////////////////////////////////
00055         void setYield (const Yield_T& iYield) {
00056             _yield = iYield;
00057         }
00058
00059         void setMean (const MeanValue_T& iMean) {
00060             _mean = iMean;
00061         }
00062
00063         void setStdDev (const StdDevValue_T& iStdDev) {
00064             _stdDev = iStdDev;
00065         }
00066
00067         void setCumulatedBookingLimit (const BookingLimit_T& iBL) {
00068             _cumulatedBookingLimit = iBL;
00069         }
00070
00071         void setCumulatedProtection (const ProtectionLevel_T& iP) {
00072             _cumulatedProtection = iP;
00073         }
00074
00075         void addBookingClass (BookingClass& iBookingClass) {
00076             _bookingClassList.push_back(&iBookingClass);
00077         }
00078
00079     public:
00080         // ////////////////////////////////// Display support method //////////////////////////////////
00081         void toStream (std::ostream& ioOut) const;
00082
00083         void fromStream (std::istream& ioIn);
00084
00085         const std::string describe() const;
00086
00087     public:

```

```

00111 // ////////// Constructors & Destructor //////////
00113 VirtualClassStruct (const VirtualClassStruct&);
00115 VirtualClassStruct (const BookingClassList_T&);
00117 ~VirtualClassStruct ();
00118
00119 private:
00121     VirtualClassStruct ();
00122
00123
00124 private:
00125 // ////////// Attributes //////////
00127     BookingClassList_T _bookingClassList;
00128
00130     Yield_T _yield;
00131
00133     MeanValue_T _mean;
00134
00136     StdDevValue_T _stdDev;
00137
00139     BookingLimit_T _cumulatedBookingLimit;
00140
00142     ProtectionLevel_T _cumulatedProtection;
00143 };
00144
00145 }
00146 #endif // __STDAIR_BOM_VIRTUALCLASSSTRUCT_HPP

```

### 33.511 stdair/bom/VirtualClassTypes.hpp File Reference

```

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

```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

#### Typedefs

- typedef std::list< VirtualClassStruct > [stdair::VirtualClassList\\_T](#)
- typedef std::map< const YieldLevel\_T, VirtualClassStruct > [stdair::VirtualClassMap\\_T](#)

### 33.512 VirtualClassTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_VIRTUALCLASSTYPES_HPP
00003 #define __STDAIR_BOM_VIRTUALCLASSTYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <list>
00010 #include <map>
00011 // STDAIR
00012 #include <stdair/stdair_basic_types.hpp>
00013
00014 namespace stdair {
00015
00016     // Forward declarations.
00017     struct VirtualClassStruct;
00018
00020     typedef std::list<VirtualClassStruct> VirtualClassList_T;
00021
00023     typedef std::map<const YieldLevel_T, VirtualClassStruct> VirtualClassMap_T;
00024 }
00025 #endif // __STDAIR_BOM_VIRTUALCLASSTYPES_HPP
00026

```

## 33.513 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

- [stdair](#)

*Handle on the StdAir library context.*

## 33.514 YieldFeatures.cpp

```
00001 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00002 // Import section
00003 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_Request.hpp>
00009 #include <stdair/service/Logger.hpp>
00010 #include <stdair/bom/YieldFeatures.hpp>
00011
00012 namespace stdair {
00013
00014 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00015 YieldFeatures::YieldFeatures()
00016 : _key (TRIP_TYPE_ONE_WAY,
00017        DEFAULT_PREFERRED_CABIN),
00018   _parent (NULL) {
00019     // That constructor is used by the serialisation process
00020 }
00021
00022 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00023 YieldFeatures::YieldFeatures (const YieldFeatures& iFeatures)
00024 : _key (iFeatures.getKey(), _parent (NULL) {
00025 }
00026
00027 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00028 YieldFeatures::YieldFeatures (const Key_T& iKey)
00029 : _key (iKey), _parent (NULL) {
00030 }
00031
00032 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00033 YieldFeatures::~YieldFeatures () {
00034 }
00035
00036 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00037 std::string YieldFeatures::toString() const {
00038     std::ostringstream ostr;
00039     ostr << describeKey();
00040     return ostr.str();
00041 }
00042
00043 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00044 bool YieldFeatures::
00045 isTripTypeValid (const TripType_T& iBookingRequestTripType) const {
00046     bool oIsTripTypeValidFlag = true;
00047
00048     // Check whether the yield trip type is the same as the booking request
00049     // trip type
00050     const TripType_T& lYieldTripType = getTripType();
00051     if (iBookingRequestTripType == lYieldTripType) {
00052         // One way case
00053         return oIsTripTypeValidFlag;
00054     }
00055
00056     if (iBookingRequestTripType == TRIP_TYPE_INBOUND ||
00057         iBookingRequestTripType == TRIP_TYPE_OUTBOUND) {
00058         // Round trip case.
00059         if (lYieldTripType == TRIP_TYPE_ROUND_TRIP) {
00060             return oIsTripTypeValidFlag;
00061         }
00062     }
00063 }
```

```

00062     }
00063
00064     oIsTripTypeValidFlag = false;
00065     return false;
00066 }
00067
00068 }
00069

```

### 33.515 stdair/bom/YieldFeatures.hpp File Reference

```

#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/YieldFeaturesKey.hpp>
#include <stdair/bom/YieldFeaturesTypes.hpp>

```

#### Classes

- class [stdair::YieldFeatures](#)  
Class representing the actual attributes for a yield date-period.

#### Namespaces

- [stdair](#)  
Handle on the StdAir library context.

### 33.516 YieldFeatures.hpp

```

00001 #ifndef __STDAIR_BOM_YIELDFEATURES_HPP
00002 #define __STDAIR_BOM_YIELDFEATURES_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/bom/BomAbstract.hpp>
00009 #include <stdair/bom/YieldFeaturesKey.hpp>
00010 #include <stdair/bom/YieldFeaturesTypes.hpp>
00011
00012 // Forward declaration
00013 namespace stdair {
00014
00015     class YieldFeatures : public BomAbstract {
00020     template <typename BOM> friend class FacBom;
00021     template <typename BOM> friend class FacCloneBom;
00022     friend class FacBomManager;
00023
00024     public:
00025         // ////////// Type definitions
00029         typedef YieldFeaturesKey Key_T;
00030
00031     public:
00032         // ////////// Display support methods //////////
00038         void toStream (std::ostream& ioOut) const {
00039             ioOut << toString();
00040         }
00041
00047         void fromStream (std::istream& ioIn) {
00048         }
00049
00053         std::string toString() const;
00054
00058         const std::string describeKey() const {
00059             return _key.toString();
00060         }
00061
00062     public:
00063         // ////////// Getters //////////
00067         const Key_T& getKey() const {
00068             return _key;
00069         }
00070

```

```

00074     BomAbstract* const getParent() const {
00075         return _parent;
00076     }
00077
00081     const HolderMap_T& getHolderMap() const {
00082         return _holderMap;
00083     }
00084
00088     const CabinCode_T& getCabinCode() const {
00089         return _key.getCabinCode();
00090     }
00091
00095     const TripType_T& getTripType() const {
00096         return _key.getTripType();
00097     }
00098
00099
00100 public:
00101     // ////////// Business methods //////////
00106     bool isTripTypeValid (const TripType_T&) const;
00107
00108
00109 protected:
00110     // ////////// Constructors and destructors //////////
00114     YieldFeatures (const Key_T&);
00115
00119     virtual ~YieldFeatures();
00120
00121 private:
00125     YieldFeatures();
00126
00130     YieldFeatures (const YieldFeatures&);
00131
00132
00133 protected:
00134     // ////////// Attributes //////////
00138     Key_T _key;
00139
00143     BomAbstract* _parent;
00144
00148     HolderMap_T _holderMap;
00149 };
00150
00151 }
00152 #endif // __STDAIR_BOM_YIELDFEATURES_HPP
00153

```

### 33.517 stdair/bom/YieldFeaturesKey.cpp File Reference

```

#include <ostream>
#include <sstream>
#include <stdair/basic/BasConst_Request.hpp>
#include <stdair/bom/YieldFeaturesKey.hpp>

```

#### Namespaces

- [stdair](#)  
Handle on the StdAir library context.

### 33.518 YieldFeaturesKey.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <ostream>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_Request.hpp>
00009 #include <stdair/bom/YieldFeaturesKey.hpp>
00010
00011 namespace stdair {
00012
00013     // //////////////////////////////////////

```



```

00014 YieldFeaturesKey::YieldFeaturesKey()
00015 : _tripType (TRIP_TYPE_ONE_WAY),
00016   _cabinCode (DEFAULT_PREFERRED_CABIN) {
00017     assert (false);
00018 }
00019
00020 // //////////////////////////////////////
00021 YieldFeaturesKey::YieldFeaturesKey (const stdair::TripType_T& iTripType,
00022                                     const stdair::CabinCode_T& iCabin)
00023 : _tripType (iTripType), _cabinCode (iCabin) {
00024 }
00025
00026 // //////////////////////////////////////
00027 YieldFeaturesKey::YieldFeaturesKey (const YieldFeaturesKey& iKey)
00028 : _tripType (iKey.getTripType()), _cabinCode (iKey.getCabinCode()) {
00029 }
00030
00031 // //////////////////////////////////////
00032 YieldFeaturesKey::~YieldFeaturesKey () {
00033 }
00034
00035 // //////////////////////////////////////
00036 void YieldFeaturesKey::toStream (std::ostream& ioOut) const {
00037   ioOut << "YieldFeaturesKey: " << toString() << std::endl;
00038 }
00039
00040 // //////////////////////////////////////
00041 void YieldFeaturesKey::fromStream (std::istream& ioIn) {
00042 }
00043
00044 // //////////////////////////////////////
00045 const std::string YieldFeaturesKey::toString() const {
00046   std::ostringstream oStr;
00047   oStr << _tripType << " -- " << _cabinCode;
00048   return oStr.str();
00049 }
00050
00051 }

```

### 33.519 stdair/bom/YieldFeaturesKey.hpp File Reference

```

#include <stdair/bom/KeyAbstract.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/stdair_inventory_types.hpp>

```

#### Classes

- struct [stdair::YieldFeaturesKey](#)

*Key of date-period.*

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.520 YieldFeaturesKey.hpp

```

00001 #ifndef __STDAIR_BOM_YIELDFEATURESKEY_HPP
00002 #define __STDAIR_BOM_YIELDFEATURESKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/bom/KeyAbstract.hpp>
00009 #include <stdair/stdair_date_time_types.hpp>
00010 #include <stdair/stdair_demand_types.hpp>
00011 #include <stdair/stdair_inventory_types.hpp>
00012

```

```

00013 namespace stdair {
00014
00018     struct YieldFeaturesKey : public KeyAbstract {
00019     public:
00020         // ////////// Construction //////////
00024         YieldFeaturesKey (const TripType_T&, const
CabinCode_T&);
00028         YieldFeaturesKey (const YieldFeaturesKey&);
00032         ~YieldFeaturesKey ();
00033     private:
00037         YieldFeaturesKey ();
00038
00039     public:
00040         // ////////// Getters //////////
00044         const TripType_T& getTripType() const {
00045             return _tripType;
00046         }
00047
00051         const CabinCode_T& getCabinCode() const {
00052             return _cabinCode;
00053         }
00054
00055     public:
00056         // ////////// Display support methods //////////
00061         void toStream (std::ostream& ioOut) const;
00062
00067         void fromStream (std::istream& ioIn);
00068
00074         const std::string toString() const;
00075
00076     private:
00077         // ////////// Attributes //////////
00081         TripType_T _tripType;
00082
00086         CabinCode_T _cabinCode;
00087     };
00088
00089 }
00090 #endif // __STDAIR_BOM_YIELDFEATURESKEY_HPP

```

## 33.521 stdair/bom/YieldFeaturesTypes.hpp File Reference

```

#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>

```

### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### Typedefs

- typedef std::list< YieldFeatures \* > [stdair::YieldFeaturesList\\_T](#)
- typedef std::map< const MapKey\_T, YieldFeatures \* > [stdair::YieldFeaturesMap\\_T](#)
- typedef std::pair< MapKey\_T, YieldFeatures \* > [stdair::YieldFeaturesWithKey\\_T](#)
- typedef std::list< YieldFeaturesWithKey\_T > [stdair::YieldFeaturesDetailedList\\_T](#)

## 33.522 YieldFeaturesTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_YIELDFEATURESTYPES_HPP
00003 #define __STDAIR_BOM_YIELDFEATURESTYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>

```

```

00011 // STDAIR
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016     // Forward declarations.
00017     class YieldFeatures;
00018
00020     typedef std::list<YieldFeatures*> YieldFeaturesList_T;
00021
00023     typedef std::map<const MapKey_T, YieldFeatures*> YieldFeaturesMap_T;
00024
00026     typedef std::pair<MapKey_T, YieldFeatures*> YieldFeaturesWithKey_T;
00027     typedef std::list<YieldFeaturesWithKey_T> YieldFeaturesDetailedList_T;
00028 }
00029 #endif // __STDAIR_BOM_YIELDFEATURESTYPES_HPP
00030

```

### 33.523 stdair/bom/YieldStore.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/bom/YieldStore.hpp>

```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.524 YieldStore.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/bom/YieldStore.hpp>
00009
00010 namespace stdair {
00011
00012     // //////////////////////////////////////
00013     YieldStore::YieldStore (const Key_T& iKey) : _key (iKey), _parent (NULL) {
00014     }
00015
00016     // //////////////////////////////////////
00017     YieldStore::~YieldStore () {
00018     }
00019
00020     // //////////////////////////////////////
00021     std::string YieldStore::toString() const {
00022         std::ostringstream oStr;
00023         oStr << _key.toString();
00024         return oStr.str();
00025     }
00026 }
00027
00028

```

### 33.525 stdair/bom/YieldStore.hpp File Reference

```

#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/YieldStoreKey.hpp>
#include <stdair/bom/YieldStoreTypes.hpp>

```

## Classes

- class [stdair::YieldStore](#)

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.526 YieldStore.hpp

```

00001 #ifndef __STDAIR_BOM_YIELDSTORE_HPP
00002 #define __STDAIR_BOM_YIELDSTORE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/stdair_inventory_types.hpp>
00011 #include <stdair/bom/BomAbstract.hpp>
00012 #include <stdair/bom/YieldStoreKey.hpp>
00013 #include <stdair/bom/YieldStoreTypes.hpp>
00014
00015 namespace stdair {
00016
00017     class YieldStore : public BomAbstract {
00018     public:
00019         template <typename BOM> friend class FacBom;
00020         friend class FacBomManager;
00021
00022     public:
00023         // Type definitions
00024         typedef YieldStoreKey Key_T;
00025
00026     public:
00027         // ////////////////////////////////// Display support methods //////////////////////////////////
00028         void toStream (std::ostream& ioOut) const { ioOut << toString(); }
00029
00030         BomAbstract* const getParent() const { return _parent; }
00031
00032         void fromStream (std::istream& ioIn) { }
00033
00034         std::string toString() const;
00035
00036         const std::string describeKey() const { return _key.toString(); }
00037
00038     public:
00039         // ////////////////////////////////// Getters //////////////////////////////////
00040         const Key_T& getKey() const { return _key; }
00041
00042         const AirlineCode_T& getAirlineCode () const {
00043             return _key.getAirlineCode();
00044         }
00045
00046     protected:
00047         YieldStore (const Key_T&);
00048         YieldStore (const YieldStore&);
00049         ~YieldStore ();
00050
00051     protected:
00052         // Attributes
00053         Key_T _key;
00054         BomAbstract* _parent;
00055     };
00056
00057 }
00058
00059 #endif // __STDAIR_BOM_YIELDSTORE_HPP
00060

```

## 33.527 stdair/bom/YieldStoreKey.cpp File Reference

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

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.528 YieldStoreKey.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // StdAir
00005 #include <stdair/bom/YieldStoreKey.hpp>
00006
00007 namespace stdair {
00008
00009     // //////////////////////////////////////
00010     YieldStoreKey::YieldStoreKey (const AirlineCode_T& iAirlineCode)
00011         : _airlineCode (iAirlineCode) {
00012     }
00013     // //////////////////////////////////////
00014     YieldStoreKey::YieldStoreKey (const YieldStoreKey& iKey)
00015         : _airlineCode (iKey._airlineCode) {
00016     }
00017
00018     // //////////////////////////////////////
00019     YieldStoreKey::~YieldStoreKey () {
00020     }
00021
00022     // //////////////////////////////////////
00023     void YieldStoreKey::toStream (std::ostream& ioOut) const {
00024         ioOut << "YieldStoreKey: " << toString() << std::endl;
00025     }
00026
00027     // //////////////////////////////////////
00028     void YieldStoreKey::fromStream (std::istream& ioIn) {
00029     }
00030
00031     // //////////////////////////////////////
00032     const std::string YieldStoreKey::toString() const {
00033         std::ostringstream oStr;
00034         oStr << _airlineCode;
00035         return oStr.str();
00036     }
00037
00038 }

```

## 33.529 stdair/bom/YieldStoreKey.hpp File Reference

```

#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>

```

## Classes

- struct [stdair::YieldStoreKey](#)

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.530 YieldStoreKey.hpp

```

00001 #ifndef __STDAIR_BOM_YIELDSTOREKEY_HPP
00002 #define __STDAIR_BOM_YIELDSTOREKEY_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section

```

```

00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_inventory_types.hpp>
00009 #include <stdair/bom/KeyAbstract.hpp>
00010
00011 namespace stdair {
00012
00014     struct YieldStoreKey : public KeyAbstract {
00015
00016     private:
00017         // /////////// Default constructor ///////////
00018         YieldStoreKey () { };
00019
00020     public:
00021         // /////////// Construction ///////////
00023         YieldStoreKey (const AirlineCode_T& iAirlineCode);
00024         YieldStoreKey (const YieldStoreKey&);
00026         ~YieldStoreKey ();
00027
00028         // /////////// Getters ///////////
00030         const AirlineCode_T& getAirlineCode() const {
00031             return _airlineCode;
00032         }
00033
00034         // /////////// Display support methods ///////////
00037         void toStream (std::ostream& ioOut) const;
00038
00041         void fromStream (std::istream& ioIn);
00042
00048         const std::string toString() const;
00049
00050     private:
00051         // Attributes
00053         AirlineCode_T _airlineCode;
00054     };
00055
00056 }
00057 #endif // __STDAIR_BOM_YIELDSTOREKEY_HPP

```

### 33.531 stdair/bom/YieldStoreTypes.hpp File Reference

```

#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>

```

#### Namespaces

- [stdair](#)  
Handle on the StdAir library context.

#### Typedefs

- typedef std::list< YieldStore \* > [stdair::YieldStoreList\\_T](#)
- typedef std::map< const MapKey\_T, YieldStore \* > [stdair::YieldStoreMap\\_T](#)

### 33.532 YieldStoreTypes.hpp

```

00001 // //////////////////////////////////////
00002 #ifndef __STDAIR_BOM_YIELDSTORETYPES_HPP
00003 #define __STDAIR_BOM_YIELDSTORETYPES_HPP
00004
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // StdAir
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {

```

```

00015
00016 // Forward declarations.
00017 class YieldStore;
00018
00020 typedef std::list<YieldStore*> YieldStoreList_T;
00021
00023 typedef std::map<const MapKey_T, YieldStore*> YieldStoreMap_T;
00024
00025 }
00026 #endif // __STDAIR_BOM_YIELDSTORETYPES_HPP
00027

```

### 33.533 stdair/command/CmdAbstract.cpp File Reference

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

#### Namespaces

- [stdair](#)  
*Handle on the StdAir library context.*

### 33.534 CmdAbstract.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // StdAir
00005 #include <stdair/command/CmdAbstract.hpp>
00006
00007 namespace stdair {
00008
00009 }

```

### 33.535 stdair/command/CmdAbstract.hpp File Reference

#### Classes

- class [stdair::CmdAbstract](#)

#### Namespaces

- [stdair](#)  
*Handle on the StdAir library context.*

### 33.536 CmdAbstract.hpp

```

00001 #ifndef __STDAIR_CMD_CMDABSTRACT_HPP
00002 #define __STDAIR_CMD_CMDABSTRACT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007
00008 namespace stdair {
00009
00011 class CmdAbstract {
00012 public:
00013
00014 };
00015
00016 }
00017 #endif // __STDAIR_CMD_CMDABSTRACT_HPP

```

## 33.537 stdair/command/CmdBomManager.cpp File Reference

## 33.538 CmdBomManager.cpp

```

00001
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <cassert>
00010 #include <sstream>
00011 // StdAir
00012 #include <stdair/basic/BasConst_General.hpp>
00013 #include <stdair/basic/BasConst_DefaultObject.hpp>
00014 #include <stdair/basic/BasConst_Request.hpp>
00015 #include <stdair/basic/BasConst_Inventory.hpp>
00016 #include <stdair/bom/BomRetriever.hpp>
00017 #include <stdair/bom/BomRoot.hpp>
00018 #include <stdair/bom/Inventory.hpp>
00019 #include <stdair/bom/AirlineFeature.hpp>
00020 #include <stdair/bom/FlightDate.hpp>
00021 #include <stdair/bom/LegDate.hpp>
00022 #include <stdair/bom/LegCabin.hpp>
00023 #include <stdair/bom/SegmentDate.hpp>
00024 #include <stdair/bom/SegmentCabin.hpp>
00025 #include <stdair/bom/FareFamily.hpp>
00026 #include <stdair/bom/BookingClass.hpp>
00027 #include <stdair/bom/AirportPair.hpp>
00028 #include <stdair/bom/PosChannel.hpp>
00029 #include <stdair/bom/DatePeriod.hpp>
00030 #include <stdair/bom/TimePeriod.hpp>
00031 #include <stdair/bom/FareFeatures.hpp>
00032 #include <stdair/bom/YieldFeatures.hpp>
00033 #include <stdair/bom/AirlineClassList.hpp>
00034 #include <stdair/bom/BomManager.hpp>
00035 #include <stdair/bom/TravelSolutionStruct.hpp>
00036 #include <stdair/bom/BookingRequestStruct.hpp>
00037 #include <stdair/factory/FacBomManager.hpp>
00038 #include <stdair/factory/FacBom.hpp>
00039 #include <stdair/command/CmdBomManager.hpp>
00040 #include <stdair/service/Logger.hpp>
00041 #include <stdair/bom/OnDDate.hpp>
00042 #include <stdair/bom/SegmentPeriod.hpp>
00043 #include <stdair/bom/FlightPeriod.hpp>
00044
00045 namespace stdair {
00046
00047 // //////////////////////////////////////
00048 void CmdBomManager::buildSampleBom (BomRoot& ioBomRoot) {
00049
00050     // DEBUG
00051     STDAIR_LOG_DEBUG ("StdAir is building the BOM tree from built-in "
00052         << "specifications.");
00053
00054     // ===== Basic Bom Tree =====
00055     // Build the inventory (flight-dates) and the schedule (flight period)
00056     // parts.
00057     buildSampleInventorySchedule (ioBomRoot);
00058
00059     // Build the pricing (fare rules) and revenue accounting (yields) parts.
00060     buildSamplePricing (ioBomRoot);
00061
00062     // ===== Partnership Bom Tree =====
00063     // Build the inventory (flight-dates) and the schedule (flight period)
00064     // parts.
00065     buildPartnershipsSampleInventoryAndRM (ioBomRoot);
00066
00067     // Build the pricing (fare rules) and revenue accounting (yields) parts.
00068     buildPartnershipsSamplePricing (ioBomRoot);
00069
00070     // Build a dummy inventory, needed by RMOL.
00071     buildCompleteDummyInventory (ioBomRoot);
00072
00073     // ===== Fare Families Bom Tree =====
00074     // Build the inventory (flight-dates) and the schedule (flight period)
00075     // parts with fare families.
00076     buildSampleInventoryScheduleForFareFamilies (ioBomRoot);
00077
00078     // Build the pricing (fare rules) and revenue accounting (yields) parts.
00079     buildSamplePricingForFareFamilies (ioBomRoot);
00080
00081     // Build a dummy inventory, needed by RMOL.
00082     buildCompleteDummyInventoryForFareFamilies (ioBomRoot);
00083 }
00084

```



```

00085 // //////////////////////////////////////
00086 void CmdBomManager::buildSampleInventorySchedule (BomRoot& ioBomRoot) {
00087
00088     // Inventory
00089     // Step 0.1: Inventory level
00090     // Create an Inventory for BA
00091     const AirlineCode_T lAirlineCodeBA ("BA");
00092     const InventoryKey lBAKey (lAirlineCodeBA);
00093     Inventory& lBAInv = FacBom<Inventory>::instance().
create (lBAKey);
00094     FacBomManager::addToListAndMap (ioBomRoot, lBAInv);
00095     FacBomManager::linkWithParent (ioBomRoot, lBAInv);
00096
00097     // Add the airline feature object to the BA inventory
00098     const AirlineFeatureKey lAirlineFeatureBAKey (lAirlineCodeBA);
00099     AirlineFeature& lAirlineFeatureBA =
00100         FacBom<AirlineFeature>::instance().create (lAirlineFeatureBAKey
);
00101     FacBomManager::setAirlineFeature (lBAInv, lAirlineFeatureBA);
00102     FacBomManager::linkWithParent (lBAInv, lAirlineFeatureBA);
00103     // Link the airline feature object with the top of the BOM tree
00104     FacBomManager::addToListAndMap (ioBomRoot, lAirlineFeatureBA);
00105
00106     // Create an Inventory for AF
00107     const AirlineCode_T lAirlineCodeAF ("AF");
00108     const InventoryKey lAFKey (lAirlineCodeAF);
00109     Inventory& lAFInv = FacBom<Inventory>::instance().
create (lAFKey);
00110     FacBomManager::addToListAndMap (ioBomRoot, lAFInv);
00111     FacBomManager::linkWithParent (ioBomRoot, lAFInv);
00112
00113     // Add the airline feature object to the AF inventory
00114     const AirlineFeatureKey lAirlineFeatureAFKey (lAirlineCodeAF);
00115     AirlineFeature& lAirlineFeatureAF =
00116         FacBom<AirlineFeature>::instance().create (lAirlineFeatureAFKey
);
00117     FacBomManager::setAirlineFeature (lAFInv, lAirlineFeatureAF);
00118     FacBomManager::linkWithParent (lAFInv, lAirlineFeatureAF);
00119     // Link the airline feature object with the top of the BOM tree
00120     FacBomManager::addToListAndMap (ioBomRoot, lAirlineFeatureAF);
00121
00122     // BA
00123     // Step 0.2: Flight-date level
00124     // Create a FlightDate (BA9/10-JUN-2011) for BA's Inventory
00125     FlightNumber_T lFlightNumber = 9;
00126     Date_T lDate (2011, 6, 10);
00127     FlightDateKey lFlightDateKey (lFlightNumber, lDate);
00128
00129     FlightDate& lBA9_20110610_FD =
00130         FacBom<FlightDate>::instance().create (lFlightDateKey);
00131     FacBomManager::addToListAndMap (lBAInv, lBA9_20110610_FD);
00132     FacBomManager::linkWithParent (lBAInv, lBA9_20110610_FD);
00133
00134     // Display the flight-date
00135     // STDAIR_LOG_DEBUG ("FlightDate: " << lBA9_20110610_FD.toString());
00136
00137     // Step 0.3: Segment-date level
00138     // Create a first SegmentDate (LHR-SYD) for BA's Inventory
00139     // See
http://www.britishairways.com/travel/flightinformation/public/fr_fr?&Carrier=BA&FlightNumber=0009&from=LHR&to=SYD&depD
00140     const AirportCode_T lLHR ("LHR");
00141     const AirportCode_T lSYD ("SYD");
00142     const DateOffset_T l1Day (1);
00143     const DateOffset_T l2Days (2);
00144     const Duration_T l2135 (21, 45, 0);
00145     const Duration_T l0610 (6, 10, 0);
00146     const Duration_T l2205 (22, 05, 0);
00147     SegmentDateKey lSegmentDateKey (lLHR, lSYD);
00148
00149     SegmentDate& lLHRSYDSegment =
00150         FacBom<SegmentDate>::instance().create (lSegmentDateKey);
00151     FacBomManager::addToListAndMap (lBA9_20110610_FD, lLHRSYDSegment);
00152     FacBomManager::linkWithParent (lBA9_20110610_FD, lLHRSYDSegment);
00153
00154     // Add the routing leg keys to the LHR-SYD segment.
00155     const std::string lBALHRRoutingLegStr = "BA;9;2011-Jun-10;LHR";
00156     const std::string lBABKKRoutingLegStr = "BA;9;2011-Jun-10;BKK";
00157     lLHRSYDSegment.addLegKey (lBALHRRoutingLegStr);
00158     lLHRSYDSegment.addLegKey (lBABKKRoutingLegStr);
00159
00160     // Fill the SegmentDate content
00161     lLHRSYDSegment.setBoardingDate (lDate);
00162     lLHRSYDSegment.setOffDate (lDate + l2Days);
00163     lLHRSYDSegment.setBoardingTime (l2135);
00164     lLHRSYDSegment.setOffTime (l0610);
00165     lLHRSYDSegment.setElapsedTime (l2135);
00166

```

```

00167 // Display the segment-date
00168 // STDAIR_LOG_DEBUG ("SegmentDate: " << lLHRSYDSegment);
00169
00170 // Create a second SegmentDate (LHR-BKK) for BA's Inventory
00171 // See
00172 http://www.britishairways.com/travel/flightinformation/public/fr_fr?Carrier=BA&FlightNumber=0009&from=LHR&to=BKK&depD
00173 const AirportCode_T lBKK ("BKK");
00174 const Duration_T l1540 (15, 40, 0);
00175 const Duration_T l1105 (11, 5, 0);
00176 lSegmentDateKey = SegmentDateKey (lLHR, lBKK);
00177
00178 SegmentDate& lLHRBKKSegment =
00179     FacBom<SegmentDate>::instance().create (lSegmentDateKey);
00180 FacBomManager::addToListAndMap (lBA9_20110610_FD, lLHRBKKSegment);
00181 FacBomManager::linkWithParent (lBA9_20110610_FD, lLHRBKKSegment);
00182
00183 // Add the routing leg key to the LHR-BKK segment.
00184 lLHRBKKSegment.addLegKey (lBALHRRoutingLegStr);
00185
00186 // Fill the SegmentDate content
00187 lLHRBKKSegment.setBoardingDate (lDate);
00188 lLHRBKKSegment.setOffDate (lDate + l1Day);
00189 lLHRBKKSegment.setBoardingTime (l2135);
00190 lLHRBKKSegment.setOffTime (l1540);
00191 lLHRBKKSegment.setElapsedTime (l1105);
00192
00193 // Display the segment-date
00194 // STDAIR_LOG_DEBUG ("SegmentDate: " << lLHRBKKSegment);
00195
00196 // Create a third SegmentDate (BKK-SYD) for BA's Inventory
00197 // See
00198 http://www.britishairways.com/travel/flightinformation/public/fr_fr?Carrier=BA&FlightNumber=0009&from=BKK&to=SYD&depD
00199 const Duration_T l1705 (17, 5, 0);
00200 const Duration_T l0905 (9, 5, 0);
00201 lSegmentDateKey = SegmentDateKey (lBKK, lSYD);
00202
00203 SegmentDate& lBKKSYSYDSegment =
00204     FacBom<SegmentDate>::instance().create (lSegmentDateKey);
00205 FacBomManager::addToListAndMap (lBA9_20110610_FD, lBKKSYSYDSegment);
00206 FacBomManager::linkWithParent (lBA9_20110610_FD, lBKKSYSYDSegment);
00207
00208 // Add the routing leg key to the BKK-SYD segment.
00209 lBKKSYSYDSegment.addLegKey (lBABKKRoutingLegStr);
00210
00211 // Fill the SegmentDate content
00212 lBKKSYSYDSegment.setBoardingDate (lDate + l1Day);
00213 lBKKSYSYDSegment.setOffDate (lDate + l2Days);
00214 lBKKSYSYDSegment.setBoardingTime (l1705);
00215 lBKKSYSYDSegment.setOffTime (l1540);
00216 lBKKSYSYDSegment.setElapsedTime (l0905);
00217
00218 // Display the segment-date
00219 // STDAIR_LOG_DEBUG ("SegmentDate: " << lBKKSYSYDSegment);
00220
00221 // Step 0.4: Leg-date level
00222 // Create a first LegDate (LHR) for BA's Inventory
00223 LegDateKey lLegDateKey (lLHR);
00224
00225 LegDate& lLHRLeg = FacBom<LegDate>::instance().
00226     create (lLegDateKey);
00227 FacBomManager::addToListAndMap (lBA9_20110610_FD, lLHRLeg);
00228 FacBomManager::linkWithParent (lBA9_20110610_FD, lLHRLeg);
00229
00230 // Fill the LegDate content
00231 lLHRLeg.setOffPoint (lBKK);
00232 lLHRLeg.setBoardingDate (lDate);
00233 lLHRLeg.setOffDate (lDate + l1Day);
00234 lLHRLeg.setBoardingTime (l2135);
00235 lLHRLeg.setOffTime (l1540);
00236 lLHRLeg.setElapsedTime (l1105);
00237
00238 // Display the leg-date
00239 // STDAIR_LOG_DEBUG ("LegDate: " << lLHRLeg.toString());
00240
00241 // Create a second LegDate (BKK)
00242 lLegDateKey = LegDateKey (lBKK);
00243
00244 LegDate& lBKKLeg = FacBom<LegDate>::instance().
00245     create (lLegDateKey);
00246 FacBomManager::addToListAndMap (lBA9_20110610_FD, lBKKLeg);
00247 FacBomManager::linkWithParent (lBA9_20110610_FD, lBKKLeg);
00248
00249 // Display the leg-date
00250 // STDAIR_LOG_DEBUG ("LegDate: " << lBKKLeg.toString());
00251
00252 // Fill the LegDate content
00253 lBKKLeg.setOffPoint (lSYD);

```

```

00250     1BKKLeg.setBoardingDate (lDate + 1lDay);
00251     1BKKLeg.setOffDate (lDate + 12Days);
00252     1BKKLeg.setBoardingTime (11705);
00253     1BKKLeg.setOffTime (11540);
00254     1BKKLeg.setElapsedTime (10905);
00255
00256     // Step 0.5: segment-cabin level
00257     // Create a SegmentCabin (Y) for the Segment LHR-BKK of BA's Inventory
00258     const CabinCode_T lY ("Y");
00259     SegmentCabinKey lYSegmentCabinKey (lY);
00260
00261     SegmentCabin& 1LHRBKKSegmentYCabin =
00262         FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
00263     FacBomManager::addToListAndMap (1LHRBKKSegment, 1LHRBKKSegmentYCabin);
00264     FacBomManager::linkWithParent (1LHRBKKSegment, 1LHRBKKSegmentYCabin);
00265
00266     // Display the segment-cabin
00267     // STDAIR_LOG_DEBUG ("SegmentCabin: " << 1LHRBKKSegmentYCabin.toString());
00268
00269     // Create a SegmentCabin (Y) of the Segment BKK-SYD;
00270     SegmentCabin& 1BKKSYSYDSegmentYCabin =
00271         FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
00272     FacBomManager::addToListAndMap (1BKKSYSYDSegment, 1BKKSYSYDSegmentYCabin);
00273     FacBomManager::linkWithParent (1BKKSYSYDSegment, 1BKKSYSYDSegmentYCabin);
00274
00275
00276     // Display the segment-cabin
00277     // STDAIR_LOG_DEBUG ("SegmentCabin: " << 1BKKSYSYDSegmentYCabin.toString());
00278
00279     // Create a SegmentCabin (Y) of the Segment LHR-SYD;
00280     SegmentCabin& 1LHRSYDSegmentYCabin =
00281         FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
00282     FacBomManager::addToListAndMap (1LHRSYDSegment, 1LHRSYDSegmentYCabin);
00283     FacBomManager::linkWithParent (1LHRSYDSegment, 1LHRSYDSegmentYCabin);
00284
00285     // Display the segment-cabin
00286     // STDAIR_LOG_DEBUG ("SegmentCabin: " << 1LHRSYDSegmentYCabin.toString());
00287
00288     // Step 0.6: leg-cabin level
00289     // Create a LegCabin (Y) for the Leg LHR-BKK on BA's Inventory
00290     LegCabinKey lYLegCabinKey (lY);
00291
00292     LegCabin& 1LHRLegYCabin =
00293         FacBom<LegCabin>::instance().create (lYLegCabinKey);
00294     FacBomManager::addToListAndMap (1LHRLeg, 1LHRLegYCabin);
00295     FacBomManager::linkWithParent (1LHRLeg, 1LHRLegYCabin);
00296
00297     // Display the leg-cabin
00298     // STDAIR_LOG_DEBUG ("LegCabin: " << 1LHRLegYCabin.toString());
00299
00300     // Create a LegCabin (Y) for the Leg BKK-SYD
00301     LegCabin& 1BKKLegYCabin =
00302         FacBom<LegCabin>::instance().create (lYLegCabinKey);
00303     FacBomManager::addToListAndMap (1BKKLeg, 1BKKLegYCabin);
00304     FacBomManager::linkWithParent (1BKKLeg, 1BKKLegYCabin);
00305
00306     // Display the leg-cabin
00307     // STDAIR_LOG_DEBUG ("LegCabin: " << 1BKKLegYCabin.toString());
00308
00309     // Step 0.7: fare family level
00310     // Create a FareFamily (1) for the Segment LHR-BKK, cabin Y on BA's Inv
00311     const FamilyCode_T l1 ("EcoSaver");
00312     FareFamilyKey l1FareFamilyKey (l1);
00313
00314     FareFamily& 1LHRBKKSegmentYCabin1Family =
00315         FacBom<FareFamily>::instance().create (l1FareFamilyKey);
00316     FacBomManager::addToListAndMap (1LHRBKKSegmentYCabin,
00317                                     1LHRBKKSegmentYCabin1Family);
00318     FacBomManager::linkWithParent (1LHRBKKSegmentYCabin,
00319                                     1LHRBKKSegmentYCabin1Family);
00320
00321     // Display the booking class
00322     // STDAIR_LOG_DEBUG ("FareFamily: "
00323     //                    << 1LHRBKKSegmentYCabin1Family.toString());
00324
00325     // Create a FareFamily (1) for the Segment BKK-SYD, cabin Y on BA's Inv
00326     FareFamily& 1BKKSYSYDSegmentYCabin1Family =
00327         FacBom<FareFamily>::instance().create (l1FareFamilyKey);
00328     FacBomManager::addToListAndMap (1BKKSYSYDSegmentYCabin,
00329                                     1BKKSYSYDSegmentYCabin1Family);
00330     FacBomManager::linkWithParent (1BKKSYSYDSegmentYCabin,
00331                                     1BKKSYSYDSegmentYCabin1Family);
00332
00333     // Display the booking class
00334     // STDAIR_LOG_DEBUG ("FareFamily: "
00335     //                    << 1LHRBKKSegmentYCabin1Family.toString());
00336
00337     // Create a FareFamily (1) for the Segment LHR-SYD, cabin Y on BA's Inv

```

```

00337     FareFamily& lLHRSYDSegmentYCabinlFamily =
00338         FacBom<FareFamily>::instance().create (lFareFamilyKey);
00339     FacBomManager::addToListAndMap (lLHRSYDSegmentYCabin,
00340         lLHRSYDSegmentYCabinlFamily);
00341     FacBomManager::linkWithParent (lLHRSYDSegmentYCabin,
00342         lLHRSYDSegmentYCabinlFamily);
00343
00344     // Display the booking class
00345     // STDAIR_LOG_DEBUG ("FareFamily: "
00346     //     << lLHRBKKSegmentYCabinlFamily.toString());
00347
00348
00349     // Step 0.8: booking class level
00350     // Create a BookingClass (Q) for the Segment LHR-BKK, cabin Y,
00351     // fare family 1 on BA's Inv
00352     const ClassCode_T lQ ("Q");
00353     BookingClassKey lQBookingClassKey (lQ);
00354
00355     BookingClass& lLHRBKKSegmentYCabinlFamilyQClass =
00356         FacBom<BookingClass>::instance().create (lQBookingClassKey);
00357     FacBomManager::addToListAndMap (lLHRBKKSegmentYCabinlFamily,
00358         lLHRBKKSegmentYCabinlFamilyQClass);
00359     FacBomManager::linkWithParent (lLHRBKKSegmentYCabinlFamily,
00360         lLHRBKKSegmentYCabinlFamilyQClass);
00361
00362     FacBomManager::addToListAndMap (lLHRBKKSegmentYCabin,
00363         lLHRBKKSegmentYCabinlFamilyQClass);
00364     FacBomManager::addToListAndMap (lLHRBKKSegment,
00365         lLHRBKKSegmentYCabinlFamilyQClass);
00366
00367     // Display the booking class
00368     // STDAIR_LOG_DEBUG ("BookingClass: "
00369     //     << lLHRBKKSegmentYCabinlFamilyQClass.toString());
00370
00371     // Create a BookingClass (Q) for the Segment BKK-SYD, cabin Y,
00372     // fare family 1 on BA's Inv
00373     BookingClass& lBKKSYSYDSegmentYCabinlFamilyQClass =
00374         FacBom<BookingClass>::instance().create (lQBookingClassKey);
00375     FacBomManager::addToListAndMap (lBKKSYSYDSegmentYCabinlFamily,
00376         lBKKSYSYDSegmentYCabinlFamilyQClass);
00377     FacBomManager::linkWithParent (lBKKSYSYDSegmentYCabinlFamily,
00378         lBKKSYSYDSegmentYCabinlFamilyQClass);
00379
00380     FacBomManager::addToListAndMap (lBKKSYSYDSegmentYCabin,
00381         lBKKSYSYDSegmentYCabinlFamilyQClass);
00382     FacBomManager::addToListAndMap (lBKKSYSYDSegment,
00383         lBKKSYSYDSegmentYCabinlFamilyQClass);
00384
00385     // Display the booking class
00386     // STDAIR_LOG_DEBUG ("BookingClass: "
00387     //     << lLHRBKKSegmentYCabinlFamilyQClass.toString());
00388
00389     // Create a BookingClass (Q) for the Segment LHR-SYD, cabin Y,
00390     // fare family 1 on BA's Inv
00391     BookingClass& lLHRSYDSegmentYCabinlFamilyQClass =
00392         FacBom<BookingClass>::instance().create (lQBookingClassKey);
00393     FacBomManager::addToListAndMap (lLHRSYDSegmentYCabinlFamily,
00394         lLHRSYDSegmentYCabinlFamilyQClass);
00395     FacBomManager::linkWithParent (lLHRSYDSegmentYCabinlFamily,
00396         lLHRSYDSegmentYCabinlFamilyQClass);
00397
00398     FacBomManager::addToListAndMap (lLHRSYDSegmentYCabin,
00399         lLHRSYDSegmentYCabinlFamilyQClass);
00400     FacBomManager::addToListAndMap (lLHRSYDSegment,
00401         lLHRSYDSegmentYCabinlFamilyQClass);
00402
00403     // Display the booking class
00404     // STDAIR_LOG_DEBUG ("BookingClass: "
00405     //     << lLHRBKKSegmentYCabinlFamilyQClass.toString());
00406
00407
00408     // ===== AF =====
00409     // Step 0.2: Flight-date level
00410     // Create a FlightDate (AF084/20-MAR-2011) for AF's Inventory
00411     lFlightNumber = 84;
00412     lDate = Date_T (2011, 3, 20);
00413     lFlightDateKey = FlightDateKey (lFlightNumber, lDate);
00414
00415     FlightDate& lAF084_20110320_FD =
00416         FacBom<FlightDate>::instance().create (lFlightDateKey);
00417     FacBomManager::addToListAndMap (lAFInv, lAF084_20110320_FD);
00418     FacBomManager::linkWithParent (lAFInv, lAF084_20110320_FD);
00419
00420     // Display the flight-date
00421     // STDAIR_LOG_DEBUG ("FlightDate: " << lAF084_20110320_FD.toString());
00422
00423     // Step 0.3: Segment-date level

```

```

00424 // Create a SegmentDate (CDG-SFO) for AF's Inventory
00425 const AirportCode_T lCDG ("CDG");
00426 const AirportCode_T lSFO ("SFO");
00427 const Duration_T l1040 (10, 40, 0);
00428 const Duration_T l1250 (12, 50, 0);
00429 const Duration_T l1110 (11, 10, 0);
00430 lSegmentDateKey = SegmentDateKey (lCDG, lSFO);
00431
00432 SegmentDate& lCDGSFOSegment =
00433     FacBom<SegmentDate>::instance().create (lSegmentDateKey);
00434 FacBomManager::addToListAndMap (lAF084_20110320_FD, lCDGSFOSegment);
00435 FacBomManager::linkWithParent (lAF084_20110320_FD, lCDGSFOSegment);
00436
00437 // Add the routing leg key to the CDG-SFO segment.
00438 const std::string lAFCDGRoutingLegStr = "AF;84;2011-Mar-20;CDG";
00439 lCDGSFOSegment.addLegKey (lAFCDGRoutingLegStr);
00440
00441 // Display the segment-date
00442 // STDAIR_LOG_DEBUG ("SegmentDate: " << lCDGSFOSegment.toString());
00443
00444 // Fill the SegmentDate content
00445 lCDGSFOSegment.setBoardingDate (lDate);
00446 lCDGSFOSegment.setOffDate (lDate);
00447 lCDGSFOSegment.setBoardingTime (l1040);
00448 lCDGSFOSegment.setOffTime (l1250);
00449 lCDGSFOSegment.setElapsedTime (l1110);
00450
00451 // Step 0.4: Leg-date level
00452 // Create a LegDate (CDG) for AF's Inventory
00453 lLegDateKey = LegDateKey (lCDG);
00454
00455 LegDate& lCDGLeg = FacBom<LegDate>::instance().
00456     create (lLegDateKey);
00457 FacBomManager::addToListAndMap (lAF084_20110320_FD, lCDGLeg);
00458 FacBomManager::linkWithParent (lAF084_20110320_FD, lCDGLeg);
00459
00460 // Fill the LegDate content
00461 lCDGLeg.setOffPoint (lSFO);
00462 lCDGLeg.setBoardingDate (lDate);
00463 lCDGLeg.setOffDate (lDate);
00464 lCDGLeg.setBoardingTime (l1040);
00465 lCDGLeg.setOffTime (l1250);
00466 lCDGLeg.setElapsedTime (l1110);
00467
00468 // Display the leg-date
00469 // STDAIR_LOG_DEBUG ("LegDate: " << lCDGLeg.toString());
00470
00471 // Step 0.5: segment-cabin level
00472 // Create a SegmentCabin (Y) for the Segment CDG-SFO of AF's Inventory
00473 SegmentCabin& lCDGSFOSegmentYCabin =
00474     FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
00475 FacBomManager::addToListAndMap (lCDGSFOSegment, lCDGSFOSegmentYCabin);
00476 FacBomManager::linkWithParent (lCDGSFOSegment, lCDGSFOSegmentYCabin);
00477
00478 // Display the segment-cabin
00479 // STDAIR_LOG_DEBUG ("SegmentCabin: " << lCDGSFOSegmentYCabin.toString());
00480
00481 // Step 0.6: leg-cabin level
00482 // Create a LegCabin (Y) for the Leg CDG-SFO on AF's Inventory
00483 LegCabin& lCDGLegYCabin =
00484     FacBom<LegCabin>::instance().create (lYLegCabinKey);
00485 FacBomManager::addToListAndMap (lCDGLeg, lCDGLegYCabin);
00486 FacBomManager::linkWithParent (lCDGLeg, lCDGLegYCabin);
00487
00488 // Display the leg-cabin
00489 // STDAIR_LOG_DEBUG ("LegCabin: " << lLHRLegYCabin.toString());
00490
00491 // Step 0.7: fare family level
00492 // Create a fareFamily (1) for the Segment CDG-SFO, cabin Y on AF's Inv
00493 FareFamily& lCDGSFOSegmentYCabin1Family =
00494     FacBom<FareFamily>::instance().create (l1FareFamilyKey);
00495 FacBomManager::addToListAndMap (lCDGSFOSegmentYCabin,
00496     lCDGSFOSegmentYCabin1Family);
00497 FacBomManager::linkWithParent (lCDGSFOSegmentYCabin,
00498     lCDGSFOSegmentYCabin1Family);
00499
00500 // Display the fare family
00501 // STDAIR_LOG_DEBUG ("fareFamily: "
00502 //
00503 // << lCDGSFOSegmentYCabin1Family.toString());
00504
00505 // Step 0.8: booking class level Create a BookingClass (Q) for the
00506 // Segment CDG-SFO, cabin Y, fare family 1 on AF's Inv
00507 BookingClass& lCDGSFOSegmentYCabin1FamilyQClass =
00508     FacBom<BookingClass>::instance().create (lQBookingClassKey);
00509 FacBomManager::addToListAndMap (lCDGSFOSegmentYCabin1Family,
00510     lCDGSFOSegmentYCabin1FamilyQClass);

```

```

00510     FacBomManager::linkWithParent (lCDGSFOSegmentYCabinlFamily,
00511                                     lCDGSFOSegmentYCabinlFamilyQClass);
00512
00513     FacBomManager::addToListAndMap (lCDGSFOSegmentYCabin,
00514                                     lCDGSFOSegmentYCabinlFamilyQClass);
00515     FacBomManager::addToListAndMap (lCDGSFOSegment,
00516                                     lCDGSFOSegmentYCabinlFamilyQClass);
00517
00518     // Display the booking class
00519     // STDAIR_LOG_DEBUG ("BookingClass: "
00520     //                     << lCDGSFOSegmentYCabinlFamilyQClass.toString());
00521
00522     /*=====
00523     =====
00524     =====*/
00525     // Schedule:
00526     // BA:
00527     // Step 1: flight period level
00528     // Create a flight period for BA9:
00529     const DoWSrtuct lDoWSrtuct ("1111111");
00530     const Date_T lBA9DateRangeStart (2010, boost::gregorian::Jun, 6);
00531     const Date_T lBA9DateRangeEnd (2010, boost::gregorian::Jun, 7);
00532     const DatePeriod_T lBA9DatePeriod (lBA9DateRangeStart, lBA9DateRangeEnd);
00533     const PeriodStruct lBA9PeriodStruct (lBA9DatePeriod, lDoWSrtuct);
00534
00535     lFlightNumber = FlightNumber_T (9);
00536
00537     FlightPeriodKey lBA9FlightPeriodKey (lFlightNumber, lBA9PeriodStruct);
00538
00539     FlightPeriod& lBA9FlightPeriod =
00540         FacBom<FlightPeriod>::instance().create (lBA9FlightPeriodKey);
00541     FacBomManager::addToListAndMap (lBAInv, lBA9FlightPeriod);
00542     FacBomManager::linkWithParent (lBAInv, lBA9FlightPeriod);
00543
00544     // Step 2: segment period level
00545     // Create a segment period for LHR-SYD:
00546
00547     SegmentPeriodKey lLHRSYDSegmentPeriodKey (lLHR, lSYD);
00548
00549     SegmentPeriod& lLHRSYDSegmentPeriod =
00550         FacBom<SegmentPeriod>::instance().create (
00551             lLHRSYDSegmentPeriodKey);
00552     FacBomManager::addToListAndMap (lBA9FlightPeriod, lLHRSYDSegmentPeriod);
00553     FacBomManager::linkWithParent (lBA9FlightPeriod, lLHRSYDSegmentPeriod);
00554
00555     lLHRSYDSegmentPeriod.setBoardingTime (l2135);
00556     lLHRSYDSegmentPeriod.setOffTime (l1540);
00557     lLHRSYDSegmentPeriod.setElapsedTime (l1105);
00558     ClassList_String_T lYM ("YM");
00559     lLHRSYDSegmentPeriod.addCabinBookingClassList (lY, lYM);
00560
00561     // AF:
00562     // Step 1: flight period level
00563     // Create a flight period for AF84:
00564     const Date_T lAF84DateRangeStart (2011, boost::gregorian::Mar, 20);
00565     const Date_T lAF84DateRangeEnd (2011, boost::gregorian::Mar, 21);
00566     const DatePeriod_T lAF84DatePeriod (lAF84DateRangeStart, lAF84DateRangeEnd);
00567     const PeriodStruct lAF84PeriodStruct (lAF84DatePeriod, lDoWSrtuct);
00568
00569     lFlightNumber = FlightNumber_T (84);
00570
00571     FlightPeriodKey lAF84FlightPeriodKey (lFlightNumber, lAF84PeriodStruct);
00572
00573     FlightPeriod& lAF84FlightPeriod =
00574         FacBom<FlightPeriod>::instance().create (lAF84FlightPeriodKey);
00575     FacBomManager::addToListAndMap (lAFInv, lAF84FlightPeriod);
00576     FacBomManager::linkWithParent (lAFInv, lAF84FlightPeriod);
00577
00578     // Step 2: segment period level
00579     // Create a segment period for CDG-SFO:
00580
00581     SegmentPeriodKey lCDGSFOSegmentPeriodKey (lCDG, lSFO);
00582
00583     SegmentPeriod& lCDGSFOSegmentPeriod =
00584         FacBom<SegmentPeriod>::instance().create (
00585             lCDGSFOSegmentPeriodKey);
00586     FacBomManager::addToListAndMap (lAF84FlightPeriod, lCDGSFOSegmentPeriod);
00587     FacBomManager::linkWithParent (lAF84FlightPeriod, lCDGSFOSegmentPeriod);
00588
00589     lCDGSFOSegmentPeriod.setBoardingTime (l1040);
00590     lCDGSFOSegmentPeriod.setOffTime (l1250);
00591     lCDGSFOSegmentPeriod.setElapsedTime (l1110);
00592     lCDGSFOSegmentPeriod.addCabinBookingClassList (lY, lYM);
00593
00594     /*=====
00595     =====
00596     =====*/

```

```

00595 // O&D
00596 // Create an O&D Date (BA;9,2010-Jun-06;LHR,SYD) for BA's Inventory
00597 OnDString_T lBALHRSYDOnDStr = "BA;9,2010-Jun-06;LHR,SYD";
00598 OnDStringList_T lBAOnDStrList;
00599 lBAOnDStrList.push_back (lBALHRSYDOnDStr);
00600
00601 OnDDateKey lBAOnDDateKey (lBAOnDStrList);
00602 OnDDate& lBA_LHRSYD_OnDDate =
00603     FacBom<OnDDate>::instance().create (lBAOnDDateKey);
00604 // Link to the inventory
00605 FacBomManager::addToListAndMap (lBAInv, lBA_LHRSYD_OnDDate);
00606 FacBomManager::linkWithParent (lBAInv, lBA_LHRSYD_OnDDate);
00607
00608 // Add the segment
00609 FacBomManager::addToListAndMap (lBA_LHRSYD_OnDDate, lLHRSYDSegment);
00610
00611 // Add total forecast info for cabin Y.
00612 const MeanStdDevPair_T lMean60StdDev6 (60.0, 6.0);
00613 const WTP_T lWTP750 = 750.0;
00614 const WTPDemandPair_T lWTP750Mean60StdDev6 (lWTP750, lMean60StdDev6);
00615 lBA_LHRSYD_OnDDate.setTotalForecast (lY, lWTP750Mean60StdDev6);
00616
00617 // Create an O&D Date (AF;84,2011-Mar-21;CDG,SFO) for AF's Inventory
00618 OnDString_T lAFHRSYDOnDStr = "AF;9,2011-Mar-20;CDG,SFO";
00619 OnDStringList_T lAFOnDStrList;
00620 lAFOnDStrList.push_back (lAFHRSYDOnDStr);
00621
00622 OnDDateKey lAFOnDDateKey (lAFOnDStrList);
00623 OnDDate& lAF_LHRSYD_OnDDate =
00624     FacBom<OnDDate>::instance().create (lAFOnDDateKey);
00625 // Link to the inventory
00626 FacBomManager::addToListAndMap (lAFInv, lAF_LHRSYD_OnDDate);
00627 FacBomManager::linkWithParent (lAFInv, lAF_LHRSYD_OnDDate);
00628
00629 // Add the segment
00630 FacBomManager::addToListAndMap (lAF_LHRSYD_OnDDate, lLHRSYDSegment);
00631
00632 // Add total forecast info for cabin Y.
00633 lAF_LHRSYD_OnDDate.setTotalForecast (lY, lWTP750Mean60StdDev6);
00634
00635 }
00636
00637 // //////////////////////////////////////
00638 void CmdBomManager::
00639 buildSampleInventoryScheduleForFareFamilies (BomRoot& ioBomRoot) {
00640
00641     // Inventory
00642     // Step 0.1: Inventory level
00643     // Get the Inventory SQ (already built by construction)
00644     const InventoryKey lSQKey ("SQ");
00645     Inventory& lSQInv = BomManager::getObject<Inventory>(ioBomRoot,
00646                                                         lSQKey.toString());
00647
00648     // SQ
00649     // Step 0.2: Flight-date level
00650     // Create a FlightDate (SQ747/8-FEB-2010) for SQ's Inventory
00651     const FlightNumber_T lFlightNumber747 = 747;
00652     const Date_T lDate (2010, 2, 8);
00653     const FlightDateKey lFlightDateKey (lFlightNumber747, lDate);
00654
00655     FlightDate& lSQ747_20100208_FD =
00656         FacBom<FlightDate>::instance().create (lFlightDateKey);
00657     FacBomManager::addToListAndMap (lSQInv, lSQ747_20100208_FD);
00658     FacBomManager::linkWithParent (lSQInv, lSQ747_20100208_FD);
00659
00660     // Display the flight-date
00661     // STDAIR_LOG_DEBUG ("FlightDate: " << lSQ747_20100208_FD.toString());
00662
00663     // Step 0.3: Segment-date level
00664     // Create a SegmentDate (SIN-BKK) for SQ's Inventory
00665     const AirportCode_T lSIN ("SIN");
00666     const AirportCode_T lBKK ("BKK");
00667     const Duration_T l0635 (6, 35, 0);
00668     const Duration_T l0800 (8, 0, 0);
00669     const Duration_T l0225 (2, 25, 0);
00670     const SegmentDateKey lSegmentDateKey (lSIN, lBKK);
00671
00672     SegmentDate& lSINBKKSegment =
00673         FacBom<SegmentDate>::instance().create (lSegmentDateKey);
00674     FacBomManager::addToListAndMap (lSQ747_20100208_FD, lSINBKKSegment);
00675     FacBomManager::linkWithParent (lSQ747_20100208_FD, lSINBKKSegment);
00676
00677     // Add the routing leg key to the SIN-BKK segment.
00678     const std::string lQSINRoutingLegStr = "SQ;747;2010-Feb-8;SIN";
00679     lSINBKKSegment.addLegKey (lQSINRoutingLegStr);
00680
00681     // Fill the SegmentDate content

```



```

00682     lSINBKKSegment.setBoardingDate (lDate);
00683     lSINBKKSegment.setOffDate (lDate);
00684     lSINBKKSegment.setBoardingTime (l0635);
00685     lSINBKKSegment.setOffTime (l0800);
00686     lSINBKKSegment.setElapsedTime (l0225);
00687
00688     // Display the segment-date
00689     // STDAIR_LOG_DEBUG ("SegmentDate: " << lSINBKKSegment);
00690
00691     // Step 0.4: Leg-date level
00692     // Create a LegDate (SIN) for SQ's Inventory
00693     const LegDateKey lLegDateKey (lSIN);
00694
00695     LegDate& lSINLeg = FacBom<LegDate>::instance().
create (lLegDateKey);
00696     FacBomManager::addToListAndMap (lSQ747_20100208_FD, lSINLeg);
00697     FacBomManager::linkWithParent (lSQ747_20100208_FD, lSINLeg);
00698
00699     // Fill the LegDate content
00700     lSINLeg.setOffPoint (lBKK);
00701     lSINLeg.setBoardingDate (lDate);
00702     lSINLeg.setOffDate (lDate);
00703     lSINLeg.setBoardingTime (l0635);
00704     lSINLeg.setOffTime (l0800);
00705     lSINLeg.setElapsedTime (l0225);
00706
00707     // Display the leg-date
00708     // STDAIR_LOG_DEBUG ("LegDate: " << lSINLeg.toString());
00709
00710     // Step 0.5: segment-cabin level
00711     // Create a SegmentCabin (Y) for the Segment SIN-BKK of SQ's Inventory
00712     const CabinCode_T lY ("Y");
00713     const SegmentCabinKey lYSegmentCabinKey (lY);
00714     SegmentCabin& lSINBKKSegmentYCabin =
00715         FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
00716     FacBomManager::addToListAndMap (lSINBKKSegment, lSINBKKSegmentYCabin);
00717     FacBomManager::linkWithParent (lSINBKKSegment, lSINBKKSegmentYCabin);
00718     lSINBKKSegmentYCabin.activateFareFamily ();
00719
00720     // Display the segment-cabin
00721     // STDAIR_LOG_DEBUG ("SegmentCabin: " << lSINBKKSegmentYCabin.toString());
00722
00723     // Step 0.6: leg-cabin level
00724     // Create a LegCabin (Y) for the Leg SIN-BKK on SQ's Inventory
00725     const LegCabinKey lYLegCabinKey (lY);
00726     LegCabin& lSINLegYCabin =
00727         FacBom<LegCabin>::instance().create (lYLegCabinKey);
00728     FacBomManager::addToListAndMap (lSINLeg, lSINLegYCabin);
00729     FacBomManager::linkWithParent (lSINLeg, lSINLegYCabin);
00730
00731     // Display the leg-cabin
00732     // STDAIR_LOG_DEBUG ("LegCabin: " << lSINLegYCabin.toString());
00733
00734     // Step 0.7: fare family level
00735     // Create a FareFamily (1) for the Segment SIN-BKK, cabin Y on SQ's Inv
00736     const FamilyCode_T l1 ("1");
00737     const FareFamilyKey l1FareFamilyKey (l1);
00738     FareFamily& lSINBKKSegmentYCabin1Family =
00739         FacBom<FareFamily>::instance().create (l1FareFamilyKey);
00740     FacBomManager::addToListAndMap (lSINBKKSegmentYCabin,
00741                                     lSINBKKSegmentYCabin1Family);
00742     FacBomManager::linkWithParent (lSINBKKSegmentYCabin,
00743                                     lSINBKKSegmentYCabin1Family);
00744
00745     // Display the booking class
00746     // STDAIR_LOG_DEBUG ("FareFamily: "
00747     //                     << lSINBKKSegmentYCabin1Family.toString());
00748
00749     // Create a FareFamily (2) for the Segment SIN-BKK, cabin Y on SQ's Inv
00750     const FamilyCode_T l2 ("2");
00751     const FareFamilyKey l2FareFamilyKey (l2);
00752     FareFamily& lSINBKKSegmentYCabin2Family =
00753         FacBom<FareFamily>::instance().create (l2FareFamilyKey);
00754     FacBomManager::addToListAndMap (lSINBKKSegmentYCabin,
00755                                     lSINBKKSegmentYCabin2Family);
00756     FacBomManager::linkWithParent (lSINBKKSegmentYCabin,
00757                                     lSINBKKSegmentYCabin2Family);
00758
00759     // Display the booking class
00760     // STDAIR_LOG_DEBUG ("FareFamily: "
00761     //                     << lSINBKKSegmentYCabin2Family.toString());
00762
00763     // Step 0.8: booking class level
00764     // Create a BookingClass (Y) for the Segment SIN-BKK, cabin Y,
00765     // fare family 2 on SQ's Inv
00766     const ClassCode_T lClassY ("Y");
00767     const BookingClassKey lYBookingClassKey (lClassY);

```



```

00768 BookingClass& lSINBKKSegmentYCabin2FamilyYClass =
00769     FacBom<BookingClass>::instance().create (lYBookingClassKey);
00770 FacBomManager::addToListAndMap (lSINBKKSegmentYCabin2Family,
00771     lSINBKKSegmentYCabin2FamilyYClass);
00772 FacBomManager::linkWithParent (lSINBKKSegmentYCabin2Family,
00773     lSINBKKSegmentYCabin2FamilyYClass);
00774
00775 FacBomManager::addToListAndMap (lSINBKKSegmentYCabin,
00776     lSINBKKSegmentYCabin2FamilyYClass);
00777 FacBomManager::addToListAndMap (lSINBKKSegment,
00778     lSINBKKSegmentYCabin2FamilyYClass);
00779 lSINBKKSegmentYCabin2FamilyYClass.setYield(1200);
00780
00781 // Display the booking class
00782 // STDAIR_LOG_DEBUG ("BookingClass: "
00783 //     << lSINBKKSegmentYCabin2FamilyYClass.toString());
00784
00785 // Create a BookingClass (B) for the Segment SIN-BKK, cabin Y,
00786 // fare family 2 on SQ's Inv
00787 const ClassCode_T lB ("B");
00788 const BookingClassKey lBBookingClassKey (lB);
00789 BookingClass& lSINBKKSegmentYCabin2FamilyBClass =
00790     FacBom<BookingClass>::instance().create (lBBookingClassKey);
00791 FacBomManager::addToListAndMap (lSINBKKSegmentYCabin2Family,
00792     lSINBKKSegmentYCabin2FamilyBClass);
00793 FacBomManager::linkWithParent (lSINBKKSegmentYCabin2Family,
00794     lSINBKKSegmentYCabin2FamilyBClass);
00795
00796 FacBomManager::addToListAndMap (lSINBKKSegmentYCabin,
00797     lSINBKKSegmentYCabin2FamilyBClass);
00798 FacBomManager::addToListAndMap (lSINBKKSegment,
00799     lSINBKKSegmentYCabin2FamilyBClass);
00800 lSINBKKSegmentYCabin2FamilyBClass.setYield(800);
00801
00802 // Display the booking class
00803 // STDAIR_LOG_DEBUG ("BookingClass: "
00804 //     << lSINBKKSegmentYCabin2FamilyBClass.toString());
00805
00806 // Create a BookingClass (M) for the Segment SIN-BKK, cabin Y,
00807 // fare family 1 on SQ's Inv
00808 const ClassCode_T lM ("M");
00809 const BookingClassKey lMBookingClassKey (lM);
00810 BookingClass& lSINBKKSegmentYCabin1FamilyMClass =
00811     FacBom<BookingClass>::instance().create (lMBookingClassKey);
00812 FacBomManager::addToListAndMap (lSINBKKSegmentYCabin1Family,
00813     lSINBKKSegmentYCabin1FamilyMClass);
00814 FacBomManager::linkWithParent (lSINBKKSegmentYCabin1Family,
00815     lSINBKKSegmentYCabin1FamilyMClass);
00816
00817 FacBomManager::addToListAndMap (lSINBKKSegmentYCabin,
00818     lSINBKKSegmentYCabin1FamilyMClass);
00819 FacBomManager::addToListAndMap (lSINBKKSegment,
00820     lSINBKKSegmentYCabin1FamilyMClass);
00821 lSINBKKSegmentYCabin1FamilyMClass.setYield(900);
00822
00823 // Display the booking class
00824 // STDAIR_LOG_DEBUG ("BookingClass: "
00825 //     << lSINBKKSegmentYCabin1FamilyMClass.toString());
00826
00827 // Create a BookingClass (Q) for the Segment SIN-BKK, cabin Y,
00828 // fare family 1 on SQ's Inv
00829 const ClassCode_T lQ ("Q");
00830 const BookingClassKey lQBookingClassKey (lQ);
00831 BookingClass& lSINBKKSegmentYCabin1FamilyQClass =
00832     FacBom<BookingClass>::instance().create (lQBookingClassKey);
00833 FacBomManager::addToListAndMap (lSINBKKSegmentYCabin1Family,
00834     lSINBKKSegmentYCabin1FamilyQClass);
00835 FacBomManager::linkWithParent (lSINBKKSegmentYCabin1Family,
00836     lSINBKKSegmentYCabin1FamilyQClass);
00837
00838 FacBomManager::addToListAndMap (lSINBKKSegmentYCabin,
00839     lSINBKKSegmentYCabin1FamilyQClass);
00840 FacBomManager::addToListAndMap (lSINBKKSegment,
00841     lSINBKKSegmentYCabin1FamilyQClass);
00842 lSINBKKSegmentYCabin1FamilyQClass.setYield(600);
00843
00844
00845 // Display the booking class
00846 // STDAIR_LOG_DEBUG ("BookingClass: "
00847 //     << lSINBKKSegmentYCabin1FamilyQClass.toString());
00848
00849 /*=====
00850 =====*/
00851
00852 // Schedule:
00853 // SQ:
00854 // Step 1: flight period level

```

```

00855 // Create a flight period for SQ747:
00856 const DoWStruct lDoWSrtuct ("1111111");
00857 const Date_T lSQ747DateRangeStart (2010, boost::gregorian::Feb, 8);
00858 const Date_T lSQ747DateRangeEnd (2010, boost::gregorian::Feb, 9);
00859 const DatePeriod_T lSQ747DatePeriod (lSQ747DateRangeStart,
00860                                     lSQ747DateRangeEnd);
00861 const PeriodStruct lSQ747PeriodStruct (lSQ747DatePeriod, lDoWSrtuct);
00862
00863 const FlightPeriodKey lSQ747FlightPeriodKey (lFlightNumber747,
00864                                             lSQ747PeriodStruct);
00865 FlightPeriod& lSQ747FlightPeriod =
00866     FacBom<FlightPeriod>::instance().create (lSQ747FlightPeriodKey);
00867 FacBomManager::addToListAndMap (lSQInv, lSQ747FlightPeriod);
00868 FacBomManager::linkWithParent (lSQInv, lSQ747FlightPeriod);
00869
00870 // Step 2: segment period level
00871 // Create a segment period for SIN-BKK:
00872
00873 const SegmentPeriodKey lSINBKKSegmentPeriodKey (lSIN, lBKK);
00874 SegmentPeriod& lSINBKKSegmentPeriod =
00875     FacBom<SegmentPeriod>::instance().create (
00876         lSINBKKSegmentPeriodKey);
00877 FacBomManager::addToListAndMap (lSQ747FlightPeriod, lSINBKKSegmentPeriod);
00878
00879 FacBomManager::linkWithParent (lSQ747FlightPeriod, lSINBKKSegmentPeriod);
00880
00881 ClassList_String_T lYBMQ ("YBMQ");
00882 lSINBKKSegmentPeriod.addCabinBookingClassList (lY, lYBMQ);
00883 lSINBKKSegmentPeriod.setBoardingTime (l0635);
00884 lSINBKKSegmentPeriod.setOffTime (l0800);
00885 lSINBKKSegmentPeriod.setElapsedTime (l0225);
00886
00887 /*=====
00888 // O&D
00889 // Create an O&D Date (SQ;747,2011-Feb-14;SIN,BKK) for SQ's Inventory
00890 const OnDString_T lSQSINBKKOnDStr = "SQ;747,2011-Feb-14;SIN,BKK";
00891 OnDStringList_T lSQOnDStrList;
00892 lSQOnDStrList.push_back (lSQSINBKKOnDStr);
00893
00894 const OnDDateKey lSQOnDDateKey (lSQOnDStrList);
00895 OnDDate& lSQ_SINBKK_OnDDate =
00896     FacBom<OnDDate>::instance().create (lSQOnDDateKey);
00897 // Link to the inventory
00898 FacBomManager::addToListAndMap (lSQInv, lSQ_SINBKK_OnDDate);
00899 FacBomManager::linkWithParent (lSQInv, lSQ_SINBKK_OnDDate);
00900 // Add total forecast info for cabin Y.
00901 const MeanStdDevPair_T lMean120StdDev12 (l20.0, l2.0);
00902 const WTP_T lWTP1000 = l000.0;
00903 const WTPDemandPair_T lWTP1000Mean120StdDev12 (lWTP1000, lMean120StdDev12);
00904 lSQ_SINBKK_OnDDate.setTotalForecast (lY, lWTP1000Mean120StdDev12);
00905
00906 // Add the segment
00907 FacBomManager::addToListAndMap (lSQ_SINBKK_OnDDate, lSINBKKSegment);
00908 }
00909
00910 // //////////////////////////////////////
00911 void CmdBomManager::buildDummyLegSegmentAccesses (BomRoot& ioBomRoot) {
00912
00913     /* Build the direct accesses between the dummy segment cabins and the dummy
00914      * leg cabins within the dummy flight dates (the dummy fare family
00915      * flight date and the classic dummy flight date).
00916      *
00917      * As for now (May 2012), that method is called only by RMOL.
00918      * It is a substitute for the code doing it automatically located in AirInv.
00919      * See the AIRINV::InventoryManager::createDirectAccesses command.
00920      */
00921
00922     // ///// Dummy Inventory Leg Segment Accesses /////
00923     // Retrieve the (sample) segment-cabin.
00924     SegmentCabin& lDummySegmentCabin =
00925         BomRetriever::retrieveDummySegmentCabin (ioBomRoot);
00926
00927     // Retrieve the (sample) leg-cabin.
00928     LegCabin& lDummyLegCabin =
00929         BomRetriever::retrieveDummyLegCabin (ioBomRoot);
00930
00931     // Links between the segment-date and the leg-date
00932     FacBomManager::addToListAndMap (lDummyLegCabin, lDummySegmentCabin);
00933     FacBomManager::addToListAndMap (lDummySegmentCabin, lDummyLegCabin);
00934
00935     // ///// Fare Families Dummy Inventory Leg Segment Accesses /////
00936     const bool isForFareFamilies = true;
00937     // Retrieve the (sample) segment-cabin for fare families.
00938     SegmentCabin& lFFDummySegmentCabin =
00939         BomRetriever::retrieveDummySegmentCabin (ioBomRoot,

```

```

isForFareFamilies);
00940
00941 // Retrieve the (sample) leg-cabin for fare families.
00942 stdair::LegCabin& lFFDummyLegCabin =
00943     stdair::BomRetriever::retrieveDummyLegCabin (ioBomRoot,
00944         isForFareFamilies);
00945
00946 // Links between the segment-date and the leg-date for fare families.
00947 FacBomManager::addToListAndMap (lFFDummyLegCabin, lFFDummySegmentCabin);
00948 FacBomManager::addToListAndMap (lFFDummySegmentCabin, lFFDummyLegCabin);
00949 }
00950
00951 // //////////////////////////////////////
00952 void CmdBomManager::buildCompleteDummyInventory (BomRoot& ioBomRoot) {
00953
00954     // Build a dummy inventory, containing a dummy flight-date with a
00955     // single segment-cabin and a single leg-cabin.
00956     const CabinCapacity_T lCapacity = DEFAULT_CABIN_CAPACITY;
00957     buildDummyInventory (ioBomRoot, lCapacity);
00958
00959     // Retrieve the (sample) segment-cabin.
00960     SegmentCabin& lDummySegmentCabin =
00961         BomRetriever::retrieveDummySegmentCabin (ioBomRoot);
00962
00963     // Retrieve the (sample) leg-cabin.
00964     LegCabin& lDummyLegCabin =
00965         BomRetriever::retrieveDummyLegCabin (ioBomRoot);
00966
00967     // Add some booking classes to the dummy segment-cabin and some
00968     // virtual ones to the dummy leg-cabin.
00969     // First booking class yield and demand information.
00970     Yield_T lYield = 100;
00971     MeanValue_T lMean = 20;
00972     StdDevValue_T lStdDev = 9;
00973     BookingClassKey lBCKey (DEFAULT_CLASS_CODE);
00974
00975     BookingClass& lDummyBookingClass =
00976         FacBom<BookingClass>::instance().create (lBCKey);
00977     lDummyBookingClass.setYield (lYield);
00978     lDummyBookingClass.setMean (lMean);
00979     lDummyBookingClass.setStdDev (lStdDev);
00980     // Add a booking class to the segment-cabin.
00981     FacBomManager::addToList (lDummySegmentCabin, lDummyBookingClass);
00982     BookingClassList_T lDummyBookingClassList;
00983     lDummyBookingClassList.push_back (&lDummyBookingClass);
00984
00985     VirtualClassStruct lDummyVirtualClass (lDummyBookingClassList);
00986     lDummyVirtualClass.setYield (lYield);
00987     lDummyVirtualClass.setMean (lMean);
00988     lDummyVirtualClass.setStdDev (lStdDev);
00989     // Add the corresponding virtual class to the leg-cabin.
00990     lDummyLegCabin.addVirtualClass (lDummyVirtualClass);
00991
00992     // Second booking class yield and demand information.
00993     lYield = 70;
00994     lMean = 45;
00995     lStdDev = 12;
00996     lDummyBookingClass.setYield (lYield);
00997     lDummyBookingClass.setMean (lMean);
00998     lDummyBookingClass.setStdDev (lStdDev);
00999     // Add a booking class to the segment-cabin.
01000     FacBomManager::addToList (lDummySegmentCabin, lDummyBookingClass);
01001
01002     lDummyVirtualClass.setYield (lYield);
01003     lDummyVirtualClass.setMean (lMean);
01004     lDummyVirtualClass.setStdDev (lStdDev);
01005     // Add the corresponding virtual class to the leg-cabin.
01006     lDummyLegCabin.addVirtualClass (lDummyVirtualClass);
01007
01008     // Third booking class yield and demand information.
01009     lYield = 42;
01010     lMean = 80;
01011     lStdDev = 16;
01012     lDummyBookingClass.setYield (lYield);
01013     lDummyBookingClass.setMean (lMean);
01014     lDummyBookingClass.setStdDev (lStdDev);
01015     // Add a booking class to the segment-cabin.
01016     FacBomManager::addToList (lDummySegmentCabin, lDummyBookingClass);
01017
01018     lDummyVirtualClass.setYield (lYield);
01019     lDummyVirtualClass.setMean (lMean);
01020     lDummyVirtualClass.setStdDev (lStdDev);
01021     // Add the corresponding virtual class to the leg-cabin.
01022     lDummyLegCabin.addVirtualClass (lDummyVirtualClass);
01023
01024 }
01025

```

```

01026 // //////////////////////////////////////
01027 void CmdBomManager::buildDummyInventory (BomRoot& ioBomRoot,
01028                                         const CabinCapacity_T& iCapacity) {
01029     // Inventory
01030     const InventoryKey lInventoryKey (DEFAULT_AIRLINE_CODE);
01031     Inventory& lInv = FacBom<Inventory>::instance().
create (lInventoryKey);
01032     FacBomManager::addToListAndMap (ioBomRoot, lInv);
01033     FacBomManager::linkWithParent (ioBomRoot, lInv);
01034
01035     // Add the airline feature object to the dummy inventory
01036     const AirlineFeatureKey lAirlineFeatureKey (DEFAULT_AIRLINE_CODE);
01037     AirlineFeature& lAirlineFeature =
01038         FacBom<AirlineFeature>::instance().create (lAirlineFeatureKey);
01039     FacBomManager::setAirlineFeature (lInv, lAirlineFeature);
01040     FacBomManager::linkWithParent (lInv, lAirlineFeature);
01041     // Link the airline feature object with the top of the BOM tree
01042     FacBomManager::addToListAndMap (ioBomRoot, lAirlineFeature);
01043
01044     // Flight-date
01045     FlightDateKey lFlightDateKey (DEFAULT_FLIGHT_NUMBER,
DEFAULT_DEPARTURE_DATE);
01046     FlightDate& lFlightDate =
01047         FacBom<FlightDate>::instance().create (lFlightDateKey);
01048     FacBomManager::addToListAndMap (lInv, lFlightDate);
01049     FacBomManager::linkWithParent (lInv, lFlightDate);
01050
01051     // Leg-date
01052     LegDateKey lLegDateKey (DEFAULT_ORIGIN);
01053     LegDate& lLeg = FacBom<LegDate>::instance().create (lLegDateKey);
01054     FacBomManager::addToListAndMap (lFlightDate, lLeg);
01055     FacBomManager::linkWithParent (lFlightDate, lLeg);
01056
01057     // Fill the LegDate content
01058     lLeg.setOffPoint (DEFAULT_DESTINATION);
01059     lLeg.setBoardingDate (DEFAULT_DEPARTURE_DATE);
01060     lLeg.setOffDate (DEFAULT_DEPARTURE_DATE);
01061     lLeg.setBoardingTime (Duration_T (14, 0, 0));
01062     lLeg.setOffTime (Duration_T (16, 0, 0));
01063     lLeg.setElapsedTime (Duration_T (8, 0, 0));
01064
01065     // Leg-cabin
01066     LegCabinKey lLegCabinKey (DEFAULT_CABIN_CODE);
01067     LegCabin& lLegCabin = FacBom<LegCabin>::instance().
create (lLegCabinKey);
01068     FacBomManager::addToListAndMap (lLeg, lLegCabin);
01069     FacBomManager::linkWithParent (lLeg, lLegCabin);
01070
01071     lLegCabin.setCapacities (iCapacity);
01072     lLegCabin.setAvailabilityPool (iCapacity);
01073
01074     // Segment-date
01075     SegmentDateKey lSegmentDateKey (DEFAULT_ORIGIN,
DEFAULT_DESTINATION);
01076     SegmentDate& lSegment =
01077         FacBom<SegmentDate>::instance().create (lSegmentDateKey);
01078     FacBomManager::addToListAndMap (lFlightDate, lSegment);
01079     FacBomManager::linkWithParent (lFlightDate, lSegment);
01080
01081     // Add the routing leg key to the dummy segment.
01082     std::ostringstream oStr;
01083     oStr << DEFAULT_AIRLINE_CODE << ";";
01084     << DEFAULT_FLIGHT_NUMBER << ";";
01085     << DEFAULT_DEPARTURE_DATE << ";";
01086     << DEFAULT_ORIGIN;
01087     lSegment.addLegKey (oStr.str());
01088
01089     // Fill the SegmentDate content
01090     lSegment.setBoardingDate (DEFAULT_DEPARTURE_DATE);
01091     lSegment.setOffDate (DEFAULT_DEPARTURE_DATE);
01092     lSegment.setBoardingTime (Duration_T (14, 0, 0));
01093     lSegment.setOffTime (Duration_T (16, 0, 0));
01094     lSegment.setElapsedTime (Duration_T (8, 0, 0));
01095
01096     // Segment-cabin
01097     SegmentCabinKey lSegmentCabinKey (DEFAULT_CABIN_CODE);
01098     SegmentCabin& lSegmentCabin =
01099         FacBom<SegmentCabin>::instance().create (lSegmentCabinKey);
01100     FacBomManager::addToListAndMap (lSegment, lSegmentCabin);
01101     FacBomManager::linkWithParent (lSegment, lSegmentCabin);
01102
01103     // Create a FareFamily (1) for the Segment LHR-BKK, cabin Y on BA's Inv
01104     const FamilyCode_T l1 ("EcoSaver");
01105     FareFamilyKey llFareFamilyKey (l1);
01106
01107     FareFamily& lSegmentYCabin1Family =
01108         FacBom<FareFamily>::instance().create (llFareFamilyKey);

```

```

01109     FacBomManager::addToListAndMap (lSegmentCabin, lSegmentYCabinlFamily);
01110     FacBomManager::linkWithParent (lSegmentCabin, lSegmentYCabinlFamily);
01111
01112     // Create a booking-class
01113     const ClassCode_T lQ ("Q");
01114     BookingClassKey lQBookingClassKey (lQ);
01115
01116     BookingClass& lSegmentYCabinlFamilyQClass =
01117         FacBom<BookingClass>::instance().create (lQBookingClassKey);
01118     FacBomManager::addToListAndMap (lSegmentYCabinlFamily,
01119                                     lSegmentYCabinlFamilyQClass);
01120     FacBomManager::linkWithParent (lSegmentYCabinlFamily,
01121                                     lSegmentYCabinlFamilyQClass);
01122
01123     FacBomManager::addToListAndMap (lSegmentCabin,
01124                                     lSegmentYCabinlFamilyQClass);
01125     FacBomManager::addToListAndMap (lSegment, lSegmentYCabinlFamilyQClass);
01126
01127     /*=====
01128     =====*/
01129     // Schedule:
01130     // XX:
01131     // Step 1: flight period level
01132     // Create a flight period for XX:
01133     const DoWStruct lDoWStruct ("111111");
01134     const Date_T lXXDateRangeStart (DEFAULT_DEPARTURE_DATE);
01135     const Date_T lXXDateRangeEnd (DEFAULT_DEPARTURE_DATE);
01136     const DatePeriod_T lXXDatePeriod (lXXDateRangeStart, lXXDateRangeEnd);
01137     const PeriodStruct lXXPeriodStruct (lXXDatePeriod, lDoWStruct);
01138
01139     FlightPeriodKey lXXFlightPeriodKey (DEFAULT_FLIGHT_NUMBER, lXXPeriodStruct);
01140
01141     FlightPeriod& lXXFlightPeriod =
01142         FacBom<FlightPeriod>::instance().create (lXXFlightPeriodKey);
01143     FacBomManager::addToListAndMap (lInv, lXXFlightPeriod);
01144     FacBomManager::linkWithParent (lInv, lXXFlightPeriod);
01145
01146     // Step 2: segment period level
01147     // Create a segment period
01148
01149     SegmentPeriodKey lXXSegmentPeriodKey (DEFAULT_ORIGIN, DEFAULT_DESTINATION);
01150
01151     SegmentPeriod& lXXSegmentPeriod =
01152         FacBom<SegmentPeriod>::instance().create (lXXSegmentPeriodKey);
01153     FacBomManager::addToListAndMap (lXXFlightPeriod, lXXSegmentPeriod);
01154     FacBomManager::linkWithParent (lXXFlightPeriod, lXXSegmentPeriod);
01155
01156     lXXSegmentPeriod.setBoardingTime (Duration_T (14, 0, 0));
01157     lXXSegmentPeriod.setOffTime (Duration_T (16, 0, 0));
01158     lXXSegmentPeriod.setElapsedTime (Duration_T (8, 0, 0));
01159     const CabinCode_T lY ("Y");
01160     const ClassList_String_T lYQ ("YQ");
01161     lXXSegmentPeriod.addCabinBookingClassList (lY,lYQ);
01162
01163 }
01164
01165
01166 // =====
01167 void CmdBomManager::
01168 buildCompleteDummyInventoryForFareFamilies (BomRoot& ioBomRoot) {
01169
01170     // Build a dummy inventory, containing a dummy flight-date with a
01171     // single segment-cabin and a single leg-cabin (for fare families
01172     // algorithms)
01173
01174     // Get the default Inventory object (already built in by construction)
01175     const InventoryKey lInventoryKey (DEFAULT_AIRLINE_CODE);
01176     Inventory& lInv = BomManager::getObject<Inventory>(ioBomRoot,
01177                                                         lInventoryKey.toString());
01178
01179     // Create a dummy Flight-date
01180     const FlightDateKey lFlightDateKey(DEFAULT_FLIGHT_NUMBER_FF,
01181                                         DEFAULT_DEPARTURE_DATE);
01182
01183     FlightDate& lFlightDate =
01184         FacBom<FlightDate>::instance().create (lFlightDateKey);
01185     FacBomManager::addToListAndMap (lInv, lFlightDate);
01186     FacBomManager::linkWithParent (lInv, lFlightDate);
01187
01188     // Create a dummy Leg-date
01189     LegDateKey lLegDateKey (DEFAULT_ORIGIN);
01190     LegDate& lLeg = FacBom<LegDate>::instance().create (lLegDateKey);
01191     FacBomManager::addToListAndMap (lFlightDate, lLeg);
01192     FacBomManager::linkWithParent (lFlightDate, lLeg);
01193
01194     // Fill the LegDate content
01195     lLeg.setOffPoint (DEFAULT_DESTINATION);

```

```

01195     lLeg.setBoardingDate (DEFAULT_DEPARTURE_DATE);
01196     lLeg.setOffDate (DEFAULT_DEPARTURE_DATE);
01197     lLeg.setBoardingTime (Duration_T (14, 0, 0));
01198     lLeg.setOffTime (Duration_T (16, 0, 0));
01199     lLeg.setElapsedTime (Duration_T (8, 0, 0));
01200
01201     // Create a dummy Leg-cabin
01202     const LegCabinKey lLegCabinKey (DEFAULT_CABIN_CODE);
01203     LegCabin& lLegCabin = FacBom<LegCabin>::instance().
create (lLegCabinKey);
01204     FacBomManager::addToListAndMap (lLeg, lLegCabin);
01205     FacBomManager::linkWithParent (lLeg, lLegCabin);
01206     const CabinCapacity_T lCapacity = DEFAULT_CABIN_CAPACITY;
01207     lLegCabin.setCapacities (lCapacity);
01208     lLegCabin.setAvailabilityPool (lCapacity);
01209
01210     // Create a dummy Segment-date
01211     const SegmentDateKey lSegmentDateKey (DEFAULT_ORIGIN, DEFAULT_DESTINATION);
01212     SegmentDate& lSegment =
        FacBom<SegmentDate>::instance().create (lSegmentDateKey);
01213     FacBomManager::addToListAndMap (lFlightDate, lSegment);
01214     FacBomManager::linkWithParent (lFlightDate, lSegment);
01215
01216     // Add the routing leg key to the dummy segment.
01217     std::ostringstream oStr;
01218     oStr << DEFAULT_AIRLINE_CODE << ";";
01219     oStr << DEFAULT_FLIGHT_NUMBER << ";";
01220     oStr << DEFAULT_DEPARTURE_DATE << ";";
01221     oStr << DEFAULT_ORIGIN;
01222     lSegment.addLegKey (oStr.str());
01223
01224     // Fill the SegmentDate content
01225     lSegment.setBoardingDate (DEFAULT_DEPARTURE_DATE);
01226     lSegment.setOffDate (DEFAULT_DEPARTURE_DATE);
01227     lSegment.setBoardingTime (Duration_T (14, 0, 0));
01228     lSegment.setOffTime (Duration_T (16, 0, 0));
01229     lSegment.setElapsedTime (Duration_T (8, 0, 0));
01230
01231     // Create a dummy Segment-cabin
01232     const SegmentCabinKey lSegmentCabinKey (DEFAULT_CABIN_CODE);
01233     SegmentCabin& lSegmentCabin =
        FacBom<SegmentCabin>::instance().create (lSegmentCabinKey);
01234     FacBomManager::addToListAndMap (lSegment, lSegmentCabin);
01235     FacBomManager::linkWithParent (lSegment, lSegmentCabin);
01236
01237     // Create a dummy FareFamily (FF1)
01238     const FamilyCode_T l1 ("FF1");
01239     const FareFamilyKey l1FareFamilyKey (l1);
01240
01241     FareFamily& lSegmentYCabinlFamily =
        FacBom<FareFamily>::instance().create (l1FareFamilyKey);
01242     // Set the forecasted demand
01243     // TODO change the size (hard code)
01244     MeanStdDevPairVector_T lDemandVectorlFareFamily;
01245     const unsigned int size = 16;
01246     for (unsigned int idx = 0; idx < size; ++idx) {
01247         double i = static_cast<double> (idx);
01248         MeanStdDevPair_T lMeanStdDevPair (i/4.0, i/20.0);
01249         lDemandVectorlFareFamily.push_back (lMeanStdDevPair);
01250     }
01251     lSegmentYCabinlFamily.setMeanStdDev (lDemandVectorlFareFamily);
01252     FacBomManager::addToListAndMap (lSegmentCabin, lSegmentYCabinlFamily);
01253     FacBomManager::linkWithParent (lSegmentCabin, lSegmentYCabinlFamily);
01254
01255     // Create a dummy booking-class
01256     const ClassCode_T lY ("Y");
01257     const BookingClassKey lYBookingClassKey (lY);
01258
01259     BookingClass& lSegmentYCabinlFamilyYClass =
        FacBom<BookingClass>::instance().create (lYBookingClassKey);
01260     Yield_T lYield = 1000;
01261     lSegmentYCabinlFamilyYClass.setYield (lYield);
01262     FacBomManager::addToListAndMap (lSegmentYCabinlFamily,
        lSegmentYCabinlFamilyYClass);
01263     FacBomManager::linkWithParent (lSegmentYCabinlFamily,
        lSegmentYCabinlFamilyYClass);
01264
01265     FacBomManager::addToListAndMap (lSegmentCabin,
        lSegmentYCabinlFamilyYClass);
01266     FacBomManager::addToListAndMap (lSegment, lSegmentYCabinlFamilyYClass);
01267
01268     // Create a second dummy booking-class
01269     const ClassCode_T lU ("U");
01270     const BookingClassKey lUBookingClassKey (lU);
01271
01272     BookingClass& lSegmentYCabinlFamilyUClass =
        FacBom<BookingClass>::instance().create (lUBookingClassKey);
01273

```

```

01280     lYield = 600;
01281     lSegmentYCabin1FamilyUClass.setYield(lYield);
01282     FacBomManager::addToListAndMap (lSegmentYCabin1Family,
01283                                     lSegmentYCabin1FamilyUClass);
01284     FacBomManager::linkWithParent (lSegmentYCabin1Family,
01285                                     lSegmentYCabin1FamilyUClass);
01286
01287     FacBomManager::addToListAndMap (lSegmentCabin,
01288     lSegmentYCabin1FamilyUClass);
01289     FacBomManager::addToListAndMap (lSegment, lSegmentYCabin1FamilyUClass);
01290
01291     // Create a second dummy FareFamily (2)
01292     const FamilyCode_T l2 ("FF2");
01293     const FareFamilyKey l2FareFamilyKey (l2);
01294
01295     FareFamily& lSegmentYCabin2Family =
01296     FacBom<FareFamily>::instance().create (l2FareFamilyKey);
01297     // Set the forecasted demand
01298     // TODO change the size (hard code)
01299     MeanStdDevPairVector_T lDemandVector2FareFamily;
01300     for (unsigned int idx = 0; idx < size; ++idx) {
01301         double i = static_cast<double> (idx);
01302         MeanStdDevPair_T lMeanStdDevPair (i/2.0, i/10.0);
01303         lDemandVector2FareFamily.push_back(lMeanStdDevPair);
01304     }
01305     lSegmentYCabin2Family.setMeanStdDev(lDemandVector2FareFamily);
01306
01307     FacBomManager::addToListAndMap (lSegmentCabin, lSegmentYCabin2Family);
01308     FacBomManager::linkWithParent (lSegmentCabin, lSegmentYCabin2Family);
01309
01310     // Create a third dummy booking-class
01311     const ClassCode_T l0 ("O");
01312     const BookingClassKey l0BookingClassKey (l0);
01313
01314     BookingClass& lSegmentYCabin2FamilyOClass =
01315     FacBom<BookingClass>::instance().create (l0BookingClassKey);
01316     lYield = 750;
01317     lSegmentYCabin2FamilyOClass.setYield(lYield);
01318     FacBomManager::addToListAndMap (lSegmentYCabin2Family,
01319                                     lSegmentYCabin2FamilyOClass);
01320     FacBomManager::linkWithParent (lSegmentYCabin2Family,
01321                                     lSegmentYCabin2FamilyOClass);
01322
01323     FacBomManager::addToListAndMap (lSegmentCabin,
01324     lSegmentYCabin2FamilyOClass);
01325     FacBomManager::addToListAndMap (lSegment, lSegmentYCabin2FamilyOClass);
01326
01327     // Create a fourth dummy booking-class
01328     const ClassCode_T lQ ("Q");
01329     const BookingClassKey lQBookingClassKey (lQ);
01330
01331     BookingClass& lSegmentYCabin2FamilyQClass =
01332     FacBom<BookingClass>::instance().create (lQBookingClassKey);
01333     lYield = 400;
01334     lSegmentYCabin2FamilyQClass.setYield(lYield);
01335     FacBomManager::addToListAndMap (lSegmentYCabin2Family,
01336                                     lSegmentYCabin2FamilyQClass);
01337     FacBomManager::linkWithParent (lSegmentYCabin2Family,
01338                                     lSegmentYCabin2FamilyQClass);
01339
01340     FacBomManager::addToListAndMap (lSegmentCabin,
01341     lSegmentYCabin2FamilyQClass);
01342     FacBomManager::addToListAndMap (lSegment, lSegmentYCabin2FamilyQClass);
01343
01344     // Schedule:
01345     // XX:
01346     // Step 1: flight period level
01347     // Create a flight period for XX:
01348     const DoWStruct lDoWSrtuct ("1111111");
01349     const Date_T lXXDateRangeStart (DEFAULT_DEPARTURE_DATE);
01350     const Date_T lXXDateRangeEnd (DEFAULT_DEPARTURE_DATE);
01351     const DatePeriod_T lXXDatePeriod (lXXDateRangeStart, lXXDateRangeEnd);
01352     const PeriodStruct lXXPeriodStruct (lXXDatePeriod, lDoWSrtuct);
01353
01354     const FlightPeriodKey lXXFlightPeriodKey (DEFAULT_FLIGHT_NUMBER_FF,
01355     lXXPeriodStruct);
01356
01357     FlightPeriod& lXXFlightPeriod =
01358     FacBom<FlightPeriod>::instance().create (lXXFlightPeriodKey);
01359     FacBomManager::addToListAndMap (lInv, lXXFlightPeriod);
01360     FacBomManager::linkWithParent (lInv, lXXFlightPeriod);
01361
01362     // Step 2: segment period level

```



```

01364 // Create a segment period
01365 const SegmentPeriodKey lXXSegmentPeriodKey (DEFAULT_ORIGIN,
01366                                             DEFAULT_DESTINATION);
01367
01368 SegmentPeriod& lXXSegmentPeriod =
01369     FacBom<SegmentPeriod>::instance().create (lXXSegmentPeriodKey);
01370 FacBomManager::addToListAndMap (lXXFlightPeriod, lXXSegmentPeriod);
01371 FacBomManager::linkWithParent (lXXFlightPeriod, lXXSegmentPeriod);
01372
01373 lXXSegmentPeriod.setBoardingTime (Duration_T (14, 0, 0));
01374 lXXSegmentPeriod.setOffTime (Duration_T (16, 0, 0));
01375 lXXSegmentPeriod.setElapsedTime (Duration_T (8, 0, 0));
01376 const CabinCode_T lYCabin ("Y");
01377 const ClassList_String_T lYUOQ ("YUOQ");
01378 lXXSegmentPeriod.addCabinBookingClassList (lYCabin,lYUOQ);
01379
01380 }
01381
01382 // //////////////////////////////////////
01383 void CmdBomManager::buildSamplePricing (BomRoot& ioBomRoot) {
01384
01385     // Set the airport-pair primary key.
01386     const AirportPairKey lAirportPairKey (AIRPORT_LHR, AIRPORT_SYD);
01387
01388     // Create the AirportPairKey object and link it to the BOM tree root.
01389     AirportPair& lAirportPair =
01390         FacBom<AirportPair>::instance().create (lAirportPairKey);
01391     FacBomManager::addToListAndMap (ioBomRoot, lAirportPair);
01392     FacBomManager::linkWithParent (ioBomRoot, lAirportPair);
01393
01394     // Set the fare date-period primary key.
01395     const Date_T lDateRangeStart (2011, boost::gregorian::Jan, 15);
01396     const Date_T lDateRangeEnd (2011, boost::gregorian::Dec, 31);
01397     const DatePeriod_T lDateRange (lDateRangeStart, lDateRangeEnd);
01398     const DatePeriodKey lDatePeriodKey (lDateRange);
01399
01400     // Create the DatePeriodKey object and link it to the PosChannel object.
01401     DatePeriod& lDatePeriod =
01402         FacBom<DatePeriod>::instance().create (lDatePeriodKey);
01403     FacBomManager::addToListAndMap (lAirportPair, lDatePeriod);
01404     FacBomManager::linkWithParent (lAirportPair, lDatePeriod);
01405
01406     // Set the point-of-sale-channel primary key.
01407     const PosChannelKey lPosChannelKey (POS_LHR, CHANNEL_DN);
01408
01409     // Create the PositionKey object and link it to the AirportPair object.
01410     PosChannel& lPosChannel =
01411         FacBom<PosChannel>::instance().create (lPosChannelKey);
01412     FacBomManager::addToListAndMap (lDatePeriod, lPosChannel);
01413     FacBomManager::linkWithParent (lDatePeriod, lPosChannel);
01414
01415     // Set the fare time-period primary key.
01416     const Time_T lTimeRangeStart (0, 0, 0);
01417     const Time_T lTimeRangeEnd (23, 0, 0);
01418     const TimePeriodKey lTimePeriodKey (lTimeRangeStart, lTimeRangeEnd);
01419
01420     // Create the TimePeriodKey and link it to the DatePeriod object.
01421     TimePeriod& lTimePeriod =
01422         FacBom<TimePeriod>::instance().create (lTimePeriodKey);
01423     FacBomManager::addToListAndMap (lPosChannel, lTimePeriod);
01424     FacBomManager::linkWithParent (lPosChannel, lTimePeriod);
01425
01426     // Pricing -- Generate the FareRule
01427     const FareFeaturesKey lFareFeaturesKey (TRIP_TYPE_ROUND_TRIP,
01428                                           NO_ADVANCE_PURCHASE,
01429                                           SATURDAY_STAY,
01430                                           CHANGE_FEES,
01431                                           NON_REFUNDABLE,
01432                                           NO_STAY_DURATION);
01433
01434     // Create the FareFeaturesKey and link it to the TimePeriod object.
01435     FareFeatures& lFareFeatures =
01436         FacBom<FareFeatures>::instance().create (lFareFeaturesKey);
01437     FacBomManager::addToListAndMap (lTimePeriod, lFareFeatures);
01438     FacBomManager::linkWithParent (lTimePeriod, lFareFeatures);
01439
01440     // Revenue Accounting -- Generate the YieldRule
01441     const YieldFeaturesKey lYieldFeaturesKey (TRIP_TYPE_ROUND_TRIP,
01442                                              CABIN_Y);
01443
01444     // Create the YieldFeaturesKey and link it to the TimePeriod object.
01445     YieldFeatures& lYieldFeatures =
01446         FacBom<YieldFeatures>::instance().create (lYieldFeaturesKey);
01447     FacBomManager::addToListAndMap (lTimePeriod, lYieldFeatures);
01448     FacBomManager::linkWithParent (lTimePeriod, lYieldFeatures);
01449
01450     // Generate Segment Features and link them to their respective

```



```

01451 // fare and yield rules.
01452 AirlineCodeList_T lAirlineCodeList;
01453 lAirlineCodeList.push_back (AIRLINE_CODE_BA);
01454 ClassList_StringList_T lClassCodeList;
01455 lClassCodeList.push_back (CLASS_CODE_Y);
01456 const AirlineClassListKey lAirlineClassListKey (lAirlineCodeList,
01457                                                  lClassCodeList);
01458
01459 // Create the AirlineClassList
01460 AirlineClassList& lAirlineClassList =
01461     FacBom<AirlineClassList>::instance().
01462     create (lAirlineClassListKey);
01463 // Link the AirlineClassList to the FareFeatures object
01464 lAirlineClassList.setFare (900);
01465 FacBomManager::addToListAndMap (lFareFeatures, lAirlineClassList);
01466 FacBomManager::linkWithParent (lFareFeatures, lAirlineClassList);
01467
01468 // Link the AirlineClassList to the YieldFeatures object
01469 lAirlineClassList.setYield (900);
01470 FacBomManager::addToListAndMap (lYieldFeatures, lAirlineClassList);
01471 // \todo (gsabatier): the following calls overrides the parent for
01472 // lAirlineClassList. Check that it is what is actually wanted.
01473 FacBomManager::linkWithParent (lYieldFeatures, lAirlineClassList);
01474 }
01475
01476 // //////////////////////////////////////
01477 void CmdBomManager::buildSamplePricingForFareFamilies (BomRoot& ioBomRoot) {
01478
01479     // Get the airport-pair primary key SIN-BKK
01480     // (already built by construction)
01481     const AirportPairKey lAirportPairKey ("SIN", "BKK");
01482     AirportPair& lAirportPair =
01483         BomManager::getObject<AirportPair>(ioBomRoot, lAirportPairKey.toString());
01484
01485     // Set the fare date-period primary key.
01486     const Date_T lDateRangeStart (2010, boost::gregorian::Feb, 1);
01487     const Date_T lDateRangeEnd (2011, boost::gregorian::Feb, 15);
01488     const DatePeriod_T lDateRange (lDateRangeStart, lDateRangeEnd);
01489     const DatePeriodKey lDatePeriodKey (lDateRange);
01490
01491     // Create the DatePeriod object and link it to the PosChannel object.
01492     DatePeriod& lDatePeriod =
01493         FacBom<DatePeriod>::instance().create (lDatePeriodKey);
01494     FacBomManager::addToListAndMap (lAirportPair, lDatePeriod);
01495     FacBomManager::linkWithParent (lAirportPair, lDatePeriod);
01496
01497     // Set the point-of-sale-channel primary key.
01498     const PosChannelKey lPosChannelKey ("SIN", CHANNEL_IN);
01499
01500     // Create the PositionKey object and link it to the AirportPair object.
01501     PosChannel& lPosChannel =
01502         FacBom<PosChannel>::instance().create (lPosChannelKey);
01503     FacBomManager::addToListAndMap (lDatePeriod, lPosChannel);
01504     FacBomManager::linkWithParent (lDatePeriod, lPosChannel);
01505
01506     // Set the fare time-period primary key.
01507     const Time_T lTimeRangeStart (0, 0, 0);
01508     const Time_T lTimeRangeEnd (23, 0, 0);
01509     const TimePeriodKey lTimePeriodKey (lTimeRangeStart, lTimeRangeEnd);
01510
01511     // Create the TimePeriodKey and link it to the DatePeriod object.
01512     TimePeriod& lTimePeriod =
01513         FacBom<TimePeriod>::instance().create (lTimePeriodKey);
01514     FacBomManager::addToListAndMap (lPosChannel, lTimePeriod);
01515     FacBomManager::linkWithParent (lPosChannel, lTimePeriod);
01516
01517     // Pricing -- Generate the FareRule
01518     const DayDuration_T ONE_MONTH_ADVANCE_PURCHASE = 30;
01519     // Generate the first FareFeatures for the class Q
01520     const FareFeaturesKey lFareFeaturesQKey (TRIP_TYPE_ONE_WAY,
01521                                              ONE_MONTH_ADVANCE_PURCHASE,
01522                                              SATURDAY_STAY,
01523                                              CHANGE_FEES,
01524                                              NON_REFUNDABLE,
01525                                              NO_STAY_DURATION);
01526
01527     // Create the FareFeaturesKey and link it to the TimePeriod object.
01528     FareFeatures& lFareFeaturesQ =
01529         FacBom<FareFeatures>::instance().create (lFareFeaturesQKey);
01530     FacBomManager::addToListAndMap (lTimePeriod, lFareFeaturesQ);
01531     FacBomManager::linkWithParent (lTimePeriod, lFareFeaturesQ);
01532
01533     // Generate the second FareFeatures for the class M
01534     const FareFeaturesKey lFareFeaturesMKey (TRIP_TYPE_ONE_WAY,
01535                                              NO_ADVANCE_PURCHASE,
01536                                              SATURDAY_STAY,
01537                                              CHANGE_FEES,

```

```

01537                                     NON_REFUNDABLE,
01538                                     NO_STAY_DURATION);
01539
01540 // Create the FareFeaturesKey and link it to the TimePeriod object.
01541 FareFeatures& lFareFeaturesM =
01542     FacBom<FareFeatures>::instance().create (lFareFeaturesMKey);
01543 FacBomManager::addToListAndMap (lTimePeriod, lFareFeaturesM);
01544 FacBomManager::linkWithParent (lTimePeriod, lFareFeaturesM);
01545
01546 // Generate the third FareFeatures for the class B
01547 const FareFeaturesKey lFareFeaturesBKey (TRIP_TYPE_ONE_WAY,
01548     ONE_MONTH_ADVANCE_PURCHASE,
01549     SATURDAY_STAY,
01550     NO_CHANGE_FEES,
01551     NO_NON_REFUNDABLE, //Refundable
01552     NO_STAY_DURATION);
01553
01554 // Create the FareFeaturesKey and link it to the TimePeriod object.
01555 FareFeatures& lFareFeaturesB =
01556     FacBom<FareFeatures>::instance().create (lFareFeaturesBKey);
01557 FacBomManager::addToListAndMap (lTimePeriod, lFareFeaturesB);
01558 FacBomManager::linkWithParent (lTimePeriod, lFareFeaturesB);
01559
01560 // Generate the fourth FareFeatures for the class Y
01561 const FareFeaturesKey lFareFeaturesYKey (TRIP_TYPE_ONE_WAY,
01562     NO_ADVANCE_PURCHASE,
01563     SATURDAY_STAY,
01564     NO_CHANGE_FEES,
01565     NO_NON_REFUNDABLE, //Refundable
01566     NO_STAY_DURATION);
01567
01568 // Create the FareFeaturesKey and link it to the TimePeriod object.
01569 FareFeatures& lFareFeaturesY =
01570     FacBom<FareFeatures>::instance().create (lFareFeaturesYKey);
01571 FacBomManager::addToListAndMap (lTimePeriod, lFareFeaturesY);
01572 FacBomManager::linkWithParent (lTimePeriod, lFareFeaturesY);
01573
01574 // Revenue Accounting -- Generate the YieldRule
01575 const YieldFeaturesKey lYieldFeaturesKey (TRIP_TYPE_ONE_WAY,
01576     CABIN_Y);
01577
01578 // Create the YieldFeaturesKey and link it to the TimePeriod object.
01579 YieldFeatures& lYieldFeatures =
01580     FacBom<YieldFeatures>::instance().create (lYieldFeaturesKey);
01581 FacBomManager::addToListAndMap (lTimePeriod, lYieldFeatures);
01582 FacBomManager::linkWithParent (lTimePeriod, lYieldFeatures);
01583
01584 // Generate Segment Features and link them to their respective
01585 // fare and yield rules.
01586 AirlineCodeList_T lAirlineCodeList;
01587 lAirlineCodeList.push_back ("SQ");
01588
01589 ClassList_StringList_T lClassYList;
01590 lClassYList.push_back (CLASS_CODE_Y);
01591 const AirlineClassListKey lAirlineClassYListKey (lAirlineCodeList,
01592     lClassYList);
01593
01594 // Create the AirlineClassList
01595 AirlineClassList& lAirlineClassYList =
01596     FacBom<AirlineClassList>::instance().
01597     create (lAirlineClassYListKey);
01598 // Link the AirlineClassList to the FareFeatures object
01599 FacBomManager::addToListAndMap (lFareFeaturesY, lAirlineClassYList);
01600 FacBomManager::linkWithParent (lFareFeaturesY, lAirlineClassYList);
01601 lAirlineClassYList.setFare (1200);
01602 lAirlineClassYList.setYield (1200);
01603
01604 // Link the AirlineClassList to the YieldFeatures object
01605 FacBomManager::addToListAndMap (lYieldFeatures, lAirlineClassYList);
01606 // \todo (gsabatier): the following calls overrides the parent for
01607 // lAirlineClassList. Check that it is what is actually wanted.
01608 FacBomManager::linkWithParent (lYieldFeatures, lAirlineClassYList);
01609
01610 ClassList_StringList_T lClassBList;
01611 lClassBList.push_back ("B");
01612 const AirlineClassListKey lAirlineClassBListKey (lAirlineCodeList,
01613     lClassBList);
01614
01615 // Create the AirlineClassList
01616 AirlineClassList& lAirlineClassBList =
01617     FacBom<AirlineClassList>::instance().
01618     create (lAirlineClassBListKey);
01619 // Link the AirlineClassList to the FareFeatures object
01620 FacBomManager::addToListAndMap (lFareFeaturesB, lAirlineClassBList);
01621 FacBomManager::linkWithParent (lFareFeaturesB, lAirlineClassBList);
01622 lAirlineClassBList.setFare (800);
01623 lAirlineClassBList.setYield (800);
01624

```

```

01622 // Link the AirlineClassList to the YieldFeatures object
01623 FacBomManager::addToListAndMap (lYieldFeatures, lAirlineClassBList);
01624 // \todo (gsabatier): the following calls overrides the parent for
01625 // lAirlineClassList. Check that it is what is actually wanted.
01626 FacBomManager::linkWithParent (lYieldFeatures, lAirlineClassBList);
01627
01628 ClassList_StringList_T lClassMList;
01629 lClassMList.push_back ("M");
01630 const AirlineClassListKey lAirlineClassMListKey (lAirlineCodeList,
01631 lClassMList);
01632
01633 // Create the AirlineClassList
01634 AirlineClassList& lAirlineClassMList =
01635 FacBom<AirlineClassList>::instance().
create (lAirlineClassMListKey);
01636 // Link the AirlineClassList to the FareFeatures object
01637 FacBomManager::addToListAndMap (lFareFeaturesM, lAirlineClassMList);
01638 FacBomManager::linkWithParent (lFareFeaturesM, lAirlineClassMList);
01639 lAirlineClassMList.setFare (900);
01640 lAirlineClassMList.setYield (900);
01641
01642 // Link the AirlineClassList to the YieldFeatures object
01643 FacBomManager::addToListAndMap (lYieldFeatures, lAirlineClassMList);
01644 // \todo (gsabatier): the following calls overrides the parent for
01645 // lAirlineClassList. Check that it is what is actually wanted.
01646 FacBomManager::linkWithParent (lYieldFeatures, lAirlineClassMList);
01647
01648 ClassList_StringList_T lClassQList;
01649 lClassQList.push_back ("Q");
01650 const AirlineClassListKey lAirlineClassQListKey (lAirlineCodeList,
01651 lClassQList);
01652
01653 // Create the AirlineClassList
01654 AirlineClassList& lAirlineClassQList =
01655 FacBom<AirlineClassList>::instance().
create (lAirlineClassQListKey);
01656 // Link the AirlineClassList to the FareFeatures object
01657 FacBomManager::addToListAndMap (lFareFeaturesQ, lAirlineClassQList);
01658 FacBomManager::linkWithParent (lFareFeaturesQ, lAirlineClassQList);
01659 lAirlineClassQList.setFare (600);
01660 lAirlineClassQList.setYield (600);
01661
01662 // Link the AirlineClassList to the YieldFeatures object
01663 FacBomManager::addToListAndMap (lYieldFeatures, lAirlineClassQList);
01664 // \todo (gsabatier): the following calls overrides the parent for
01665 // lAirlineClassList. Check that it is what is actually wanted.
01666 FacBomManager::linkWithParent (lYieldFeatures, lAirlineClassQList);
01667
01668 }
01669
01670 // //////////////////////////////////////
01671 void CmdBomManager::
01672 buildSampleTravelSolutionForPricing (TravelSolutionList_T& ioTravelSolutionList) {
01673
01674 // Clean the list
01675 ioTravelSolutionList.clear();
01676
01677 //
01678 const std::string lBA9_SegmentDateKey ("BA, 9, 2011-06-10, LHR, SYD, 21:45");
01679
01680 // Add the segment date key to the travel solution
01681 TravelSolutionStruct lTS;
01682 lTS.addSegment (lBA9_SegmentDateKey);
01683
01684 // Add the travel solution to the list
01685 ioTravelSolutionList.push_back (lTS);
01686 }
01687
01688 // //////////////////////////////////////
01689 void CmdBomManager::
01690 buildSampleTravelSolutions (TravelSolutionList_T& ioTravelSolutionList) {
01691
01692 // Clean the list
01693 ioTravelSolutionList.clear();
01694
01695 //
01696 const std::string lBA9_SegmentDateKey ("BA, 9, 2011-06-10, LHR, SYD, 21:45");
01697
01698 // Add the segment date key to the travel solution
01699 TravelSolutionStruct lTS1;
01700 lTS1.addSegment (lBA9_SegmentDateKey);
01701
01702 // Fare option number 1
01703 const ClassCode_T lClassPathQ (CLASS_CODE_Q);
01704 const Fare_T lFare900 (900);
01705 const ChangeFees_T lChangeFee (CHANGE_FEES);
01706 const NonRefundable_T isNonRefundable (NON_REFUNDABLE);

```

```

01707     const SaturdayStay_T lSaturdayStay (SATURDAY_STAY);
01708     const FareOptionStruct lFareOption1 (lClassPathQ, lFare900, lChangeFee,
01709                                         isNonRefundable, lSaturdayStay);
01710
01711     // Add (a copy of) the fare option
01712     lTS1.addFareOption (lFareOption1);
01713     //
01714
01715     // Map of class availabilities: set the availability for the Q
01716     // booking class (the one corresponding to the fare option) to 8.
01717     ClassAvailabilityMap_T lClassAvailabilityMap1;
01718     const Availability_T lAvl1 (8);
01719     bool hasInsertOfQBeenSuccessful = lClassAvailabilityMap1.
01720         insert (ClassAvailabilityMap_T::value_type (lClassPathQ, lAvl1)).second;
01721     assert (hasInsertOfQBeenSuccessful == true);
01722     // Add the map to the dedicated list held by the travel solution
01723     lTS1.addClassAvailabilityMap (lClassAvailabilityMap1);
01724
01725     // Add the travel solution to the list
01726     ioTravelSolutionList.push_back (lTS1);
01727
01728     //
01729     const std::string lQF12_SegmentDateKey ("QF, 12, 2011-06-10, LHR, SYD, 20:45");
01730
01731     // Add the segment date key to the travel solution
01732     TravelSolutionStruct lTS2;
01733     lTS2.addSegment (lQF12_SegmentDateKey);
01734
01735     // Fare option number 2
01736     const ClassCode_T lClassPathY (CLASS_CODE_Y);
01737     const Fare_T lFare1000 (1000);
01738     const ChangeFees_T lNoChangeFee (NO_CHANGE_FEES);
01739     const NonRefundable_T lIsRefundable (NO_NON_REFUNDABLE);
01740     const FareOptionStruct lFareOption2 (lClassPathY, lFare1000, lNoChangeFee,
01741                                         isRefundable, lSaturdayStay);
01742
01743     // Map of class availabilities: set the availability for the Y
01744     // booking class (the one corresponding to the fare option) to 9.
01745     ClassAvailabilityMap_T lClassAvailabilityMap2;
01746     const Availability_T lAvl2 (9);
01747     const bool hasInsertOfYBeenSuccessful = lClassAvailabilityMap2.
01748         insert (ClassAvailabilityMap_T::value_type (lClassPathY, lAvl2)).second;
01749     assert (hasInsertOfYBeenSuccessful == true);
01750     // Add the map to the dedicated list held by the travel solution
01751     lTS2.addClassAvailabilityMap (lClassAvailabilityMap2);
01752
01753     // Add (a copy of) the fare option
01754     lTS2.addFareOption (lFareOption2);
01755
01756     // Fare option number 3
01757     const Fare_T lFare920 (920);
01758     const FareOptionStruct lFareOption3 (lClassPathQ, lFare920, lNoChangeFee,
01759                                         isNonRefundable, lSaturdayStay);
01760
01761     // Map of class availabilities: set the availability for the Q
01762     // booking class (the one corresponding to the fare option) to 9.
01763     hasInsertOfQBeenSuccessful = lClassAvailabilityMap2.
01764         insert (ClassAvailabilityMap_T::value_type (lClassPathQ, lAvl2)).second;
01765     assert (hasInsertOfYBeenSuccessful == true);
01766     // Add the map to the dedicated list held by the travel solution
01767     lTS2.addClassAvailabilityMap (lClassAvailabilityMap2);
01768
01769     // Add (a copy of) the fare option
01770     lTS2.addFareOption (lFareOption3);
01771
01772     // Add the travel solution to the list
01773     ioTravelSolutionList.push_back (lTS2);
01774
01775 }
01776
01777 // //////////////////////////////////////
01778 BookingRequestStruct CmdBomManager::buildSampleBookingRequest() {
01779     // Origin
01780     const AirportCode_T lOrigin (AIRPORT_LHR);
01781
01782     // Destination
01783     const AirportCode_T lDestination (AIRPORT_SYD);
01784
01785     // Point of Sale (POS)
01786     const CityCode_T lPOS (POS_LHR);
01787
01788     // Preferred departure date (10-JUN-2011)
01789     const Date_T lPreferredDepartureDate (2011, boost::gregorian::Jun, 10);
01790
01791     // Preferred departure time (08:00)
01792     const Duration_T lPreferredDepartureTime (8, 0, 0);
01793

```

```

01794 // Date of the request (15-MAY-2011)
01795 const Date_T lRequestDate (2011, boost::gregorian::May, 15);
01796
01797 // Time of the request (10:00)
01798 const Duration_T lRequestTime (10, 0, 0);
01799
01800 // Date-time of the request (made of the date and time above)
01801 const DateTime_T lRequestDateTime (lRequestDate, lRequestTime);
01802
01803 // Preferred cabin (also named class of service sometimes)
01804 const CabinCode_T lPreferredCabin (CABIN_ECO);
01805
01806 // Number of persons in the party
01807 const PartySize_T lPartySize (3);
01808
01809 // Channel (direct/indirect, on-line/off-line)
01810 const ChannelLabel_T lChannel (CHANNEL_DN);
01811
01812 // Type of the trip (one-way, inbound/outbound of a return trip)
01813 const TripType_T lTripType (TRIP_TYPE_INBOUND);
01814
01815 // Duration of the stay (expressed as a number of days)
01816 const DayDuration_T lStayDuration (DEFAULT_STAY_DURATION);
01817
01818 // Frequent flyer tier (member, silver, gold, platinum, senator, etc)
01819 const FrequentFlyer_T lFrequentFlyerType (
FREQUENT_FLYER_MEMBER);
01820
01821 // Maximum willing-to-pay (WTP, expressed in monetary unit, e.g., EUR)
01822 const WTP_T lWTP (DEFAULT_WTP);
01823
01824 // Value of time, for the customer (expressed in monetary unit per
01825 // unit of time, e.g., EUR/hour)
01826 const PriceValue_T lValueOfTime (DEFAULT_VALUE_OF_TIME);
01827
01828 // Restrictions
01829 const ChangeFees_T lChangeFees = false;
01830 const Disutility_T lChangeFeeDisutility = 30;
01831 const NonRefundable_T lNonRefundable = false;
01832 const Disutility_T lNonRefundableDisutility = 50;
01833
01834 // Creation of the booking request structure
01835 BookingRequestStruct oBookingRequest (lOrigin, lDestination, lPOS,
01836                                       lPreferredDepartureDate,
01837                                       lRequestDateTime,
01838                                       lPreferredCabin,
01839                                       lPartySize, lChannel,
01840                                       lTripType, lStayDuration,
01841                                       lFrequentFlyerType,
01842                                       lPreferredDepartureTime,
01843                                       lWTP, lValueOfTime,
01844                                       lChangeFees, lChangeFeeDisutility,
01845                                       lNonRefundable,
01846                                       lNonRefundableDisutility);
01847
01848 return oBookingRequest;
01849 }
01850
01851 // //////////////////////////////////////
01852 BookingRequestStruct CmdBomManager::buildSampleBookingRequestForCRS() {
01853 // Origin
01854 const AirportCode_T lOrigin (AIRPORT_SIN);
01855
01856 // Destination
01857 const AirportCode_T lDestination (AIRPORT_BKK);
01858
01859 // Point of Sale (POS)
01860 const CityCode_T lPOS (POS_SIN);
01861
01862 // Preferred departure date (30-JAN-2010)
01863 const Date_T lPreferredDepartureDate (2010, boost::gregorian::Jan, 30);
01864
01865 // Preferred departure time (10:00)
01866 const Duration_T lPreferredDepartureTime (10, 0, 0);
01867
01868 // Date of the request (22-JAN-2010)
01869 const Date_T lRequestDate (2010, boost::gregorian::Jan, 22);
01870
01871 // Time of the request (10:00)
01872 const Duration_T lRequestTime (10, 0, 0);
01873
01874 // Date-time of the request (made of the date and time above)
01875 const DateTime_T lRequestDateTime (lRequestDate, lRequestTime);
01876
01877 // Preferred cabin (also named class of service sometimes)
01878 const CabinCode_T lPreferredCabin (CABIN_ECO);
01879

```

```

01880 // Number of persons in the party
01881 const PartySize_T lPartySize (3);
01882
01883 // Channel (direct/indirect, on-line/off-line)
01884 const ChannelLabel_T lChannel (CHANNEL_IN);
01885
01886 // Type of the trip (one-way, inbound/outbound of a return trip)
01887 const TripType_T lTripType (TRIP_TYPE_INBOUND);
01888
01889 // Duration of the stay (expressed as a number of days)
01890 const DayDuration_T lStayDuration (DEFAULT_STAY_DURATION);
01891
01892 // Frequent flyer tier (member, silver, gold, platinum, senator, etc)
01893 const FrequentFlyer_T lFrequentFlyerType (
FREQUENT_FLYER_MEMBER);
01894
01895 // Maximum willing-to-pay (WTP, expressed in monetary unit, e.g., EUR)
01896 const WTP_T lWTP (DEFAULT_WTP);
01897
01898 // Value of time, for the customer (expressed in monetary unit per
01899 // unit of time, e.g., EUR/hour)
01900 const PriceValue_T lValueOfTime (DEFAULT_VALUE_OF_TIME);
01901
01902 // Restrictions
01903 const ChangeFees_T lChangeFees = true;
01904 const Disutility_T lChangeFeeDisutility = 50;
01905 const NonRefundable_T lNonRefundable = true;
01906 const Disutility_T lNonRefundableDisutility = 50;
01907
01908 // Creation of the booking request structure
01909 BookingRequestStruct oBookingRequest (lOrigin,
01910                                     lDestination,
01911                                     lPOS,
01912                                     lPreferredDepartureDate,
01913                                     lRequestDateTime,
01914                                     lPreferredCabin,
01915                                     lPartySize, lChannel,
01916                                     lTripType, lStayDuration,
01917                                     lFrequentFlyerType,
01918                                     lPreferredDepartureTime,
01919                                     lWTP, lValueOfTime,
01920                                     lChangeFees, lChangeFeeDisutility,
01921                                     lNonRefundable,
01922                                     lNonRefundableDisutility);
01923
01924 return oBookingRequest;
01925 }
01926
01927 // //////////////////////////////////////
01928 void CmdBomManager::
01929 buildPartnershipsSampleInventoryAndRM (BomRoot& ioBomRoot) {
01930
01931 // Step 0.1: Inventory level
01932 // Create an Inventory for SQ
01933 const AirlineCode_T lAirlineCodeSQ ("SQ");
01934 const InventoryKey lSQKey (lAirlineCodeSQ);
01935 Inventory& lSQInv = FacBom<Inventory>::instance().
create (lSQKey);
01936 FacBomManager::addToListAndMap (ioBomRoot, lSQInv);
01937 FacBomManager::linkWithParent (ioBomRoot, lSQInv);
01938
01939 // Add the airline feature object to the SQ inventory
01940 const AirlineFeatureKey lAirlineFeatureSQKey (lAirlineCodeSQ);
01941 AirlineFeature& lAirlineFeatureSQ =
FacBom<AirlineFeature>::instance().create (lAirlineFeatureSQKey
);
01942 FacBomManager::setAirlineFeature (lSQInv, lAirlineFeatureSQ);
01943 FacBomManager::linkWithParent (lSQInv, lAirlineFeatureSQ);
01944 // Link the airline feature object with the top of the BOM tree
01945 FacBomManager::addToListAndMap (ioBomRoot, lAirlineFeatureSQ);
01946
01947 // Create an Inventory for CX
01948 const AirlineCode_T lAirlineCodeCX ("CX");
01949 const InventoryKey lCXKey (lAirlineCodeCX);
01950 Inventory& lCXInv = FacBom<Inventory>::instance().
create (lCXKey);
01951 FacBomManager::addToListAndMap (ioBomRoot, lCXInv);
01952 FacBomManager::linkWithParent (ioBomRoot, lCXInv);
01953
01954 // Add the airline feature object to the CX inventory
01955 const AirlineFeatureKey lAirlineFeatureCXKey (lAirlineCodeCX);
01956 AirlineFeature& lAirlineFeatureCX =
FacBom<AirlineFeature>::instance().create (lAirlineFeatureCXKey
);
01957 FacBomManager::setAirlineFeature (lCXInv, lAirlineFeatureCX);
01958 FacBomManager::linkWithParent (lCXInv, lAirlineFeatureCX);
01959 // Link the airline feature object with the top of the BOM tree
01960
01961

```

```

01962     FacBomManager::addToListAndMap (ioBomRoot, lAirlineFeatureCX);
01963
01964     // // // // SQ // // // //
01965     // Step 0.2: Flight-date level
01966     // Create a FlightDate (SQ11/08-MAR-2010) for SQ's Inventory
01967     FlightNumber_T lFlightNumber = 11;
01968     Date_T lDate (2010, 3, 8);
01969     FlightDateKey lFlightDateKey (lFlightNumber, lDate);
01970
01971     FlightDate& lSQ11_20100308_FD =
01972         FacBom<FlightDate>::instance().create (lFlightDateKey);
01973     FacBomManager::addToListAndMap (lSQInv, lSQ11_20100308_FD);
01974     FacBomManager::linkWithParent (lSQInv, lSQ11_20100308_FD);
01975
01976     // Create a (mkt) FlightDate (SQ1200/08-MAR-2010) for SQ's Inventory
01977     FlightNumber_T lMktFlightNumber = 1200;
01978     //lDate = Date_T (2010, 3, 8);
01979     FlightDateKey lMktFlightDateKey (lMktFlightNumber, lDate);
01980
01981     FlightDate& lSQ1200_20100308_FD =
01982         FacBom<FlightDate>::instance().create (lMktFlightDateKey);
01983     FacBomManager::addToListAndMap (lSQInv, lSQ1200_20100308_FD);
01984     FacBomManager::linkWithParent (lSQInv, lSQ1200_20100308_FD);
01985
01986     // Display the flight-date
01987     // STDAIR_LOG_DEBUG ("FlightDate: " << lBA9_20110610_FD.toString());
01988
01989     // Step 0.3: Segment-date level
01990     // Create a first SegmentDate (SIN-BKK) for SQ's Inventory
01991     const AirportCode_T lSIN ("SIN");
01992     const AirportCode_T lBKK ("BKK");
01993     const DateOffset_T l1Day (1);
01994     const DateOffset_T l2Days (2);
01995     const Duration_T l0820 (8, 20, 0);
01996     const Duration_T l1100 (11, 0, 0);
01997     const Duration_T l0340 (3, 40, 0);
01998     SegmentDateKey lSegmentDateKey (lSIN, lBKK);
01999
02000     SegmentDate& lSINBKKSegment =
02001         FacBom<SegmentDate>::instance().create (lSegmentDateKey);
02002     FacBomManager::addToListAndMap (lSQ11_20100308_FD, lSINBKKSegment);
02003     FacBomManager::linkWithParent (lSQ11_20100308_FD, lSINBKKSegment);
02004
02005     // Add the routing leg key to the SIN-BKK segment.
02006     const std::string lSQSINRoutingLegStr = "SQ;11;2010-Mar-8;SIN";
02007     lSINBKKSegment.addLegKey (lSQSINRoutingLegStr);
02008
02009     // Fill the SegmentDate content
02010     lSINBKKSegment.setBoardingDate (lDate);
02011     lSINBKKSegment.setOffDate (lDate);
02012     lSINBKKSegment.setBoardingTime (l0820);
02013     lSINBKKSegment.setOffTime (l1100);
02014     lSINBKKSegment.setElapsedTime (l0340);
02015
02016     // Create a second (mkt) SegmentDate (BKK-HKG) for SQ's Inventory
02017     const AirportCode_T lHKG ("HKG");
02018     const Duration_T l1200 (12, 0, 0);
02019     const Duration_T l1540 (15, 40, 0);
02020     const Duration_T l0240 (2, 40, 0);
02021     SegmentDateKey lMktSegmentDateKey (lBKK, lHKG);
02022
02023     SegmentDate& lMktBKKHKGSegment =
02024         FacBom<SegmentDate>::instance().create (lMktSegmentDateKey);
02025     FacBomManager::addToListAndMap (lSQ1200_20100308_FD, lMktBKKHKGSegment);
02026     FacBomManager::linkWithParent (lSQ1200_20100308_FD, lMktBKKHKGSegment);
02027
02028     // Add the routing leg key CX;12;2010-Mar-8;BKK to the marketing
02029     // SQ;1200;2010-Mar-8;BKK-HKG segment.
02030     const std::string lCXBKKRoutingLegStr = "CX;12;2010-Mar-8;BKK";
02031     lMktBKKHKGSegment.addLegKey (lCXBKKRoutingLegStr);
02032
02033     // Fill the (mkt) SegmentDate content
02034     lMktBKKHKGSegment.setBoardingDate (lDate);
02035     lMktBKKHKGSegment.setOffDate (lDate);
02036     lMktBKKHKGSegment.setBoardingTime (l1200);
02037     lMktBKKHKGSegment.setOffTime (l1540);
02038     lMktBKKHKGSegment.setElapsedTime (l0240);
02039
02040     // Step 0.4: Leg-date level
02041     // Create a first LegDate (SIN) for SQ's Inventory
02042     LegDateKey lLegDateKey (lSIN);
02043
02044     LegDate& lSINLeg = FacBom<LegDate>::instance().
create (lLegDateKey);
02045     FacBomManager::addToListAndMap (lSQ11_20100308_FD, lSINLeg);
02046     FacBomManager::linkWithParent (lSQ11_20100308_FD, lSINLeg);
02047

```



```

02048 // Fill the LegDate content
02049 lSINLeg.setOffPoint (lBKK);
02050 lSINLeg.setBoardingDate (lDate);
02051 lSINLeg.setOffDate (lDate);
02052 lSINLeg.setBoardingTime (l0820);
02053 lSINLeg.setOffTime (l1100);
02054 lSINLeg.setElapsedTime (l0340);
02055
02056 // Step 0.5: segment-cabin level
02057 // Create a SegmentCabin (Y) for the Segment SIN-BKK of SQ's Inventory
02058 const CabinCode_T lY ("Y");
02059 SegmentCabinKey lYSegmentCabinKey (lY);
02060
02061 SegmentCabin& lSINBKKSegmentYCabin =
02062     FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
02063 FacBomManager::addToListAndMap (lSINBKKSegment, lSINBKKSegmentYCabin);
02064 FacBomManager::linkWithParent (lSINBKKSegment, lSINBKKSegmentYCabin);
02065
02066 // Create a SegmentCabin (Y) for the (mkt) Segment BKK-HKG of SQ's Inventory
02067 SegmentCabin& lMktBKKHKGSegmentYCabin =
02068     FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
02069 FacBomManager::addToListAndMap (lMktBKKHKGSegment,
lMktBKKHKGSegmentYCabin);
02070 FacBomManager::linkWithParent (lMktBKKHKGSegment, lMktBKKHKGSegmentYCabin)
;
02071
02072
02073 // Step 0.6: leg-cabin level
02074 // Create a LegCabin (Y) for the Leg SIN-BKK on SQ's Inventory
02075 LegCabinKey lYLegCabinKey (lY);
02076
02077 LegCabin& lSINLegYCabin =
02078     FacBom<LegCabin>::instance().create (lYLegCabinKey);
02079 FacBomManager::addToListAndMap (lSINLeg, lSINLegYCabin);
02080 FacBomManager::linkWithParent (lSINLeg, lSINLegYCabin);
02081
02082 CabinCapacity_T lCapacity (100);
02083 lSINLegYCabin.setCapacities (lCapacity);
02084 lSINLegYCabin.setAvailabilityPool (lCapacity);
02085
02086
02087 // Step 0.7: fare family level
02088 // Create a FareFamily (l) for the Segment SIN-BKK, cabin Y on SQ's Inv
02089 const FamilyCode_T ll ("EcoSaver");
02090 FareFamilyKey llFareFamilyKey (ll);
02091
02092 FareFamily& lSINBKKSegmentYCabinlFamily =
02093     FacBom<FareFamily>::instance().create (llFareFamilyKey);
02094 FacBomManager::addToListAndMap (lSINBKKSegmentYCabin,
lSINBKKSegmentYCabinlFamily);
02095 FacBomManager::linkWithParent (lSINBKKSegmentYCabin,
lSINBKKSegmentYCabinlFamily);
02096
02097 // Create a FareFamily (l) for the (mkt) Segment BKK-HKG, cabin Y on SQ's Inv
02098 FareFamily& lMktBKKHKGSegmentYCabinlFamily =
02099     FacBom<FareFamily>::instance().create (llFareFamilyKey);
02100 FacBomManager::addToListAndMap (lMktBKKHKGSegmentYCabin,
lMktBKKHKGSegmentYCabinlFamily);
02101 FacBomManager::linkWithParent (lMktBKKHKGSegmentYCabin,
lMktBKKHKGSegmentYCabinlFamily);
02102
02103 // Step 0.8: booking class level
02104 // Create a BookingClass (Y) for the Segment SIN-BKK, cabin Y,
02105 // fare family 1 on SQ's Inv
02106 BookingClassKey lYBookingClassKey (lY);
02107
02108 BookingClass& lSINBKKSegmentYCabinlFamilyYClass =
02109     FacBom<BookingClass>::instance().create (lYBookingClassKey);
02110 FacBomManager::addToListAndMap (lSINBKKSegmentYCabinlFamily,
lSINBKKSegmentYCabinlFamilyYClass);
02111 FacBomManager::linkWithParent (lSINBKKSegmentYCabinlFamily,
lSINBKKSegmentYCabinlFamilyYClass);
02112
02113 FacBomManager::addToListAndMap (lSINBKKSegmentYCabin,
lSINBKKSegmentYCabinlFamilyYClass);
02114 FacBomManager::addToListAndMap (lSINBKKSegment,
lSINBKKSegmentYCabinlFamilyYClass);
02115
02116 lSINBKKSegmentYCabinlFamilyYClass.setYield(700);
02117
02118 // Create a BookingClass (Y) for the (mkt) Segment BKK-HKG, cabin Y,
02119 // fare family 1 on SQ's Inv
02120 BookingClass& lMktBKKHKGSegmentYCabinlFamilyYClass =
02121     FacBom<BookingClass>::instance().create (lYBookingClassKey);
02122 FacBomManager::addToListAndMap (lMktBKKHKGSegmentYCabinlFamily,
lMktBKKHKGSegmentYCabinlFamilyYClass);
02123 FacBomManager::linkWithParent (lMktBKKHKGSegmentYCabinlFamily,

```



```

02133                                     lMktBKKHKGSegmentYCabinlFamilyYClass);
02134
02135     FacBomManager::addToListAndMap (lMktBKKHKGSegmentYCabin,
02136                                     lMktBKKHKGSegmentYCabinlFamilyYClass);
02137     FacBomManager::addToListAndMap (lMktBKKHKGSegment,
02138                                     lMktBKKHKGSegmentYCabinlFamilyYClass);
02139
02140     lMktBKKHKGSegmentYCabinlFamilyYClass.setYield(700);
02141
02142
02143     // Create a BookingClass (M) for the Segment SIN-BKK, cabin Y,
02144     // fare family 1 on SQ's Inv
02145     const ClassCode_T lM ("M");
02146     BookingClassKey lMBookingClassKey (lM);
02147
02148     BookingClass& lSINBKKSegmentYCabinlFamilyMClass =
02149         FacBom<BookingClass>::instance().create (lMBookingClassKey);
02150     FacBomManager::addToListAndMap (lSINBKKSegmentYCabinlFamily,
02151                                     lSINBKKSegmentYCabinlFamilyMClass);
02152     FacBomManager::linkWithParent (lSINBKKSegmentYCabinlFamily,
02153                                     lSINBKKSegmentYCabinlFamilyMClass);
02154
02155     FacBomManager::addToListAndMap (lSINBKKSegmentYCabin,
02156                                     lSINBKKSegmentYCabinlFamilyMClass);
02157     FacBomManager::addToListAndMap (lSINBKKSegment,
02158                                     lSINBKKSegmentYCabinlFamilyMClass);
02159
02160     lSINBKKSegmentYCabinlFamilyMClass.setYield(500);
02161
02162     // Create a BookingClass (M) for the (mkt) Segment BKK-HKG, cabin Y,
02163     // fare family 1 on SQ's Inv
02164     BookingClass& lMktBKKHKGSegmentYCabinlFamilyMClass =
02165         FacBom<BookingClass>::instance().create (lMBookingClassKey);
02166     FacBomManager::addToListAndMap (lMktBKKHKGSegmentYCabinlFamily,
02167                                     lMktBKKHKGSegmentYCabinlFamilyMClass);
02168     FacBomManager::linkWithParent (lMktBKKHKGSegmentYCabinlFamily,
02169                                     lMktBKKHKGSegmentYCabinlFamilyMClass);
02170
02171     FacBomManager::addToListAndMap (lMktBKKHKGSegmentYCabin,
02172                                     lMktBKKHKGSegmentYCabinlFamilyMClass);
02173     FacBomManager::addToListAndMap (lMktBKKHKGSegment,
02174                                     lMktBKKHKGSegmentYCabinlFamilyMClass);
02175
02176     lMktBKKHKGSegmentYCabinlFamilyMClass.setYield(500);
02177
02178     /* ===== */
02179
02180     // Step 1.0: O&D level
02181     // Create an O&D Date (SQ11/08-MAR-2010/SIN-BKK-SQ1200/08-MAR-2010/BKK-HKG)
02182     // for SQ's Inventory
02183     OnDString_T lSQSINBKKOnDStr = "SQ;11,2010-Mar-08;SIN,BKK";
02184     OnDString_T lMktSQBKKHKGOnDStr = "SQ;1200,2010-Mar-08;BKK,HKG";
02185     OnDStringList_T lOnDStringList;
02186     lOnDStringList.push_back (lSQSINBKKOnDStr);
02187     lOnDStringList.push_back (lMktSQBKKHKGOnDStr);
02188
02189     OnDDateKey lOnDDateKey (lOnDStringList);
02190     OnDDate& lSQ_SINHKG_OnDDate =
02191         FacBom<OnDDate>::instance().create (lOnDDateKey);
02192     // Link to the inventory
02193     FacBomManager::addToListAndMap (lSQInv, lSQ_SINHKG_OnDDate);
02194     FacBomManager::linkWithParent (lSQInv, lSQ_SINHKG_OnDDate);
02195
02196     // Add the segments
02197     FacBomManager::addToListAndMap (lSQ_SINHKG_OnDDate, lSINBKKSegment);
02198     FacBomManager::addToListAndMap (lSQ_SINHKG_OnDDate, lMktBKKHKGSegment);
02199
02200     // Add total forecast info for cabin Y.
02201     const MeanStdDevPair_T lMean60StdDev6 (60.0, 6.0);
02202     const WTP_T lWTP750 = 750.0;
02203     const WTPDemandPair_T lWTP750Mean60StdDev6 (lWTP750, lMean60StdDev6);
02204     lSQ_SINHKG_OnDDate.setTotalForecast (lY, lWTP750Mean60StdDev6);
02205
02206     // Add demand info (optional).
02207     // 2 legs here, so 2 CabinClassPair to add in the list.
02208     // First leg: cabin Y, class M.
02209     CabinClassPair_T lCC_YM1 (lY,lM);
02210     // Second leg: cabin Y, class M too.
02211     CabinClassPair_T lCC_YM2 (lY,lM);
02212     CabinClassPairList_T lCabinClassPairList;
02213     lCabinClassPairList.push_back(lCC_YM1);
02214     lCabinClassPairList.push_back(lCC_YM2);
02215     const MeanStdDevPair_T lMean20StdDev2 (20.0, 2.0);
02216     const Yield_T lYield850 = 850.0;
02217     const YieldDemandPair_T lYield850Mean20StdDev2 (lYield850, lMean20StdDev2);
02218     lSQ_SINHKG_OnDDate.setDemandInformation (lCabinClassPairList, lYield850Mean20StdDev2);
02219

```

```

02220 CabinClassPair_T lCC_YY1 (lY,lY);
02221 CabinClassPair_T lCC_YY2 (lY,lY);
02222 lCabinClassPairList.clear();
02223 lCabinClassPairList.push_back(lCC_YY1);
02224 lCabinClassPairList.push_back(lCC_YY2);
02225 const MeanStdDevPair_T lMean10StdDev1 (10.0, 1.0);
02226 const Yield_T lYield1200 = 1200.0;
02227 const YieldDemandPair_T lYield1200Mean10StdDev1 (lYield1200,
02228                                                  lMean10StdDev1);
02229 lSQ_SINBKK_OnDDate.setDemandInformation (lCabinClassPairList,
02230                                         lYield1200Mean10StdDev1);
02231
02232 // Create an O&D Date (SQ11/08-MAR-2010/SIN-BKK) for SQ's Inventory
02233 lOnDStringList.clear();
02234 lOnDStringList.push_back (lSQSINBKKOnDStr);
02235
02236 lOnDDateKey = OnDDateKey(lOnDStringList);
02237 OnDDate& lSQ_SINBKK_OnDDate =
02238     FacBom<OnDDate>::instance().create (lOnDDateKey);
02239 // Link to the inventory
02240 FacBomManager::addToListAndMap (lSQInv, lSQ_SINBKK_OnDDate);
02241 FacBomManager::linkWithParent (lSQInv, lSQ_SINBKK_OnDDate);
02242
02243 // Add the segments
02244 FacBomManager::addToListAndMap (lSQ_SINBKK_OnDDate, lSINBKKSegment);
02245
02246 // Add total forecast info for cabin Y.
02247 const WTP_T lWTP400 = 400.0;
02248 const WTPDemandPair_T lWTP400Mean60StdDev6 (lWTP400, lMean60StdDev6);
02249 lSQ_SINBKK_OnDDate.setTotalForecast (lY, lWTP400Mean60StdDev6);
02250
02251 // Add demand info (optional).
02252 lCabinClassPairList.clear();
02253 lCabinClassPairList.push_back(lCC_YM1);
02254 const MeanStdDevPair_T lMean20StdDev1 (20.0, 1.0);
02255 const Yield_T lYield500 = 500.0;
02256 const YieldDemandPair_T lYield500Mean20StdDev1 (lYield500, lMean20StdDev1);
02257 lSQ_SINBKK_OnDDate.setDemandInformation (lCabinClassPairList,
02258                                         lYield500Mean20StdDev1);
02259
02260 lCabinClassPairList.clear();
02261 lCabinClassPairList.push_back(lCC_YY1);
02262 const Yield_T lYield700 = 700.0;
02263 const YieldDemandPair_T lYield700Mean20StdDev1 (lYield700, lMean10StdDev1 );
02264 lSQ_SINBKK_OnDDate.setDemandInformation (lCabinClassPairList,
02265                                         lYield700Mean20StdDev1);
02266
02267 //*****
02268 // Create an O&D Date (SQ1200/08-MAR-2010/BKK-HKG) for SQ's Inventory
02269 lFullKeyList.clear();
02270 lFullKeyList.push_back (lMktSQBKKHKGFullKeyStr);
02271
02272 lOnDDateKey = OnDDateKey(lFullKeyList);
02273 OnDDate& lMktSQ_BKKHKG_OnDDate =
02274     FacBom<OnDDate>::instance().create (lOnDDateKey);
02275 // Link to the inventory
02276 FacBomManager::addToListAndMap (lSQInv, lMktSQ_BKKHKG_OnDDate);
02277 FacBomManager::linkWithParent (lSQInv, lMktSQ_BKKHKG_OnDDate);
02278
02279 // Add the segments
02280 FacBomManager::addToListAndMap (lMktSQ_BKKHKG_OnDDate, lMktBKKHKGSegment);
02281
02282 // Demand info is not added for purely marketed O&Ds
02283 // Add demand info
02284 // lCabinClassPairList.clear();
02285 // lCabinClassPairList.push_back(lCC_YM2);
02286 // lMktSQ_BKKHKG_OnDDate.setDemandInformation (lCabinClassPairList, 500.0, 20.0, 1.0);
02287 //*****
02288
02289 // ///// CX /////
02290 // Step 0.2: Flight-date level
02291 // Create a FlightDate (CX12/08-MAR-2010) for CX's Inventory
02292 lFlightNumber = 12;
02293 //lDate = Date_T (2010, 2, 8);
02294 lFlightDateKey = FlightDateKey (lFlightNumber, lDate);
02295
02296 FlightDate& lCX12_20100308_FD =
02297     FacBom<FlightDate>::instance().create (lFlightDateKey);
02298 FacBomManager::addToListAndMap (lCXInv, lCX12_20100308_FD);
02299 FacBomManager::linkWithParent (lCXInv, lCX12_20100308_FD);
02300
02301 // Create a (mkt) FlightDate (CX1100/08-FEB-2010) for CX's Inventory
02302 lFlightNumber = 1100;
02303 //lDate = Date_T (2010, 2, 8);
02304 lMktFlightDateKey = FlightDateKey (lFlightNumber, lDate);
02305
02306

```

```

02307 FlightDate& lCX1100_20100308_FD =
02308     FacBom<FlightDate>::instance().create (lMktFlightDateKey);
02309 FacBomManager::addToListAndMap (lCXInv, lCX1100_20100308_FD);
02310 FacBomManager::linkWithParent (lCXInv, lCX1100_20100308_FD);
02311
02312 // Display the flight-date
02313 // STDAIR_LOG_DEBUG ("FlightDate: " << lAF084_20110320_FD.toString());
02314
02315 // Step 0.3: Segment-date level
02316 // Create a SegmentDate BKK-HKG for CX's Inventory
02317
02318 lSegmentDateKey = SegmentDateKey (lBKK, lHKG);
02319
02320 SegmentDate& lBKKHKGSegment =
02321     FacBom<SegmentDate>::instance().create (lSegmentDateKey);
02322 FacBomManager::addToListAndMap (lCX12_20100308_FD, lBKKHKGSegment);
02323 FacBomManager::linkWithParent (lCX12_20100308_FD, lBKKHKGSegment);
02324
02325 // Add the routing leg key to the marketing BKK-HKG segment.
02326 lBKKHKGSegment.addLegKey (lCXBKKRoutingLegStr);
02327
02328 // Fill the SegmentDate content
02329 lBKKHKGSegment.setBoardingDate (lDate);
02330 lBKKHKGSegment.setOffDate (lDate);
02331 lBKKHKGSegment.setBoardingTime (l1200);
02332 lBKKHKGSegment.setOffTime (l1540);
02333 lBKKHKGSegment.setElapsedTime (l0240);
02334
02335 // Create a second (mkt) SegmentDate (SIN-BKK) for CX's Inventory
02336 lMktSegmentDateKey = SegmentDateKey (lSIN, lBKK);
02337
02338 SegmentDate& lMktSINBKKSegment =
02339     FacBom<SegmentDate>::instance().create (lMktSegmentDateKey);
02340 FacBomManager::addToListAndMap (lCX1100_20100308_FD, lMktSINBKKSegment);
02341 FacBomManager::linkWithParent (lCX1100_20100308_FD, lMktSINBKKSegment);
02342
02343 // Add the routing leg key SQ;11;2010-Mar-8;SIN to the marketing
02344 // CX;1100;2010-Mar-8;SIN-BKK segment.
02345 lMktSINBKKSegment.addLegKey (lSQSINRoutingLegStr);
02346
02347 // Fill the (mkt) SegmentDate content
02348 lMktSINBKKSegment.setBoardingDate (lDate);
02349 lMktSINBKKSegment.setOffDate (lDate);
02350 lMktSINBKKSegment.setBoardingTime (l0820);
02351 lMktSINBKKSegment.setOffTime (l1100);
02352 lMktSINBKKSegment.setElapsedTime (l0340);
02353
02354 // Step 0.4: Leg-date level
02355 // Create a LegDate (BKK) for CX's Inventory
02356 lLegDateKey = LegDateKey (lBKK);
02357
02358 LegDate& lBKKLeg = FacBom<LegDate>::instance().
create (lLegDateKey);
02359 FacBomManager::addToListAndMap (lCX12_20100308_FD, lBKKLeg);
02360 FacBomManager::linkWithParent (lCX12_20100308_FD, lBKKLeg);
02361
02362 // Fill the LegDate content
02363 lBKKLeg.setOffPoint (lHKG);
02364 lBKKLeg.setBoardingDate (lDate);
02365 lBKKLeg.setOffDate (lDate);
02366 lBKKLeg.setBoardingTime (l1200);
02367 lBKKLeg.setOffTime (l1540);
02368 lBKKLeg.setElapsedTime (l0240);
02369
02370 // Display the leg-date
02371 // STDAIR_LOG_DEBUG ("LegDate: " << lCDGLeg.toString());
02372
02373 // Step 0.5: segment-cabin level
02374 // Create a SegmentCabin (Y) for the Segment BKK-HKG of CX's Inventory
02375 SegmentCabin& lBKKHKGSegmentYCabin =
02376     FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
02377 FacBomManager::addToListAndMap (lBKKHKGSegment, lBKKHKGSegmentYCabin);
02378 FacBomManager::linkWithParent (lBKKHKGSegment, lBKKHKGSegmentYCabin);
02379
02380 // Create a SegmentCabin (Y) for the (mkt) Segment SIN-BKK of CX's Inventory
02381 SegmentCabin& lMktSINBKKSegmentYCabin =
02382     FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
02383 FacBomManager::addToListAndMap (lMktSINBKKSegment,
lMktSINBKKSegmentYCabin);
02384 FacBomManager::linkWithParent (lMktSINBKKSegment, lMktSINBKKSegmentYCabin)
;
02385
02386 // Step 0.6: leg-cabin level
02387 // Create a LegCabin (Y) for the Leg BKK-HKG on CX's Inventory
02388 LegCabin& lBKKLegYCabin =
02389     FacBom<LegCabin>::instance().create (lYLegCabinKey);
02390 FacBomManager::addToListAndMap (lBKKLeg, lBKKLegYCabin);

```

```

02391     FacBomManager::linkWithParent (lBKKLeg, lBKKLegYCabin);
02392
02393     lCapacity = CabinCapacity_T(100);
02394     lBKKLegYCabin.setCapacities (lCapacity);
02395     lBKKLegYCabin.setAvailabilityPool (lCapacity);
02396
02397     // Step 0.7: fare family level
02398     // Create a fareFamily (1) for the Segment BKK-HKG, cabin Y on CX's Inv
02399     FareFamily& lBKKHKGSegmentYCabinlFamily =
02400         FacBom<FareFamily>::instance().create (l1FareFamilyKey);
02401     FacBomManager::addToListAndMap (lBKKHKGSegmentYCabin,
02402                                     lBKKHKGSegmentYCabinlFamily);
02403     FacBomManager::linkWithParent (lBKKHKGSegmentYCabin,
02404                                     lBKKHKGSegmentYCabinlFamily);
02405
02406     // Create a FareFamily (1) for the (mkt) Segment SIN-BKK, cabin Y on CX's Inv
02407     FareFamily& lMktSINBKKSegmentYCabinlFamily =
02408         FacBom<FareFamily>::instance().create (l1FareFamilyKey);
02409     FacBomManager::addToListAndMap (lMktSINBKKSegmentYCabin,
02410                                     lMktSINBKKSegmentYCabinlFamily);
02411     FacBomManager::linkWithParent (lMktSINBKKSegmentYCabin,
02412                                     lMktSINBKKSegmentYCabinlFamily);
02413
02414
02415     // Step 0.8: booking class level
02416     // Create a BookingClass (Y) for the
02417     // Segment BKK-HKG, cabin Y, fare family 1 on CX's Inv
02418     BookingClass& lBKKHKGSegmentYCabinlFamilyYClass =
02419         FacBom<BookingClass>::instance().create (lYBookingClassKey);
02420     FacBomManager::addToListAndMap (lBKKHKGSegmentYCabinlFamily,
02421                                     lBKKHKGSegmentYCabinlFamilyYClass);
02422     FacBomManager::linkWithParent (lBKKHKGSegmentYCabinlFamily,
02423                                     lBKKHKGSegmentYCabinlFamilyYClass);
02424
02425     FacBomManager::addToListAndMap (lBKKHKGSegmentYCabin,
02426                                     lBKKHKGSegmentYCabinlFamilyYClass);
02427     FacBomManager::addToListAndMap (lBKKHKGSegment,
02428                                     lBKKHKGSegmentYCabinlFamilyYClass);
02429
02430     lBKKHKGSegmentYCabinlFamilyYClass.setYield(700);
02431
02432     // Create a BookingClass (Y) for the (mkt) Segment SIN-BKK, cabin Y,
02433     // fare family 1 on CX's Inv
02434     BookingClass& lMktSINBKKSegmentYCabinlFamilyYClass =
02435         FacBom<BookingClass>::instance().create (lYBookingClassKey);
02436     FacBomManager::addToListAndMap (lMktSINBKKSegmentYCabinlFamily,
02437                                     lMktSINBKKSegmentYCabinlFamilyYClass);
02438     FacBomManager::linkWithParent (lMktSINBKKSegmentYCabinlFamily,
02439                                     lMktSINBKKSegmentYCabinlFamilyYClass);
02440
02441     FacBomManager::addToListAndMap (lMktSINBKKSegmentYCabin,
02442                                     lMktSINBKKSegmentYCabinlFamilyYClass);
02443     FacBomManager::addToListAndMap (lMktSINBKKSegment,
02444                                     lMktSINBKKSegmentYCabinlFamilyYClass);
02445
02446     lMktSINBKKSegmentYCabinlFamilyYClass.setYield(700);
02447
02448     //Create a BookingClass (M) for the
02449     // Segment BKK-HKG, cabin Y, fare family 1 on CX's Inv
02450     BookingClass& lBKKHKGSegmentYCabinlFamilyMClass =
02451         FacBom<BookingClass>::instance().create (lMBookingClassKey);
02452     FacBomManager::addToListAndMap (lBKKHKGSegmentYCabinlFamily,
02453                                     lBKKHKGSegmentYCabinlFamilyMClass);
02454     FacBomManager::linkWithParent (lBKKHKGSegmentYCabinlFamily,
02455                                     lBKKHKGSegmentYCabinlFamilyMClass);
02456
02457     FacBomManager::addToListAndMap (lBKKHKGSegmentYCabin,
02458                                     lBKKHKGSegmentYCabinlFamilyMClass);
02459     FacBomManager::addToListAndMap (lBKKHKGSegment,
02460                                     lBKKHKGSegmentYCabinlFamilyMClass);
02461
02462     lBKKHKGSegmentYCabinlFamilyMClass.setYield(500);
02463
02464     // Create a BookingClass (M) for the (mkt) Segment SIN-BKK, cabin Y,
02465     // fare family 1 on CX's Inv
02466     BookingClass& lMktSINBKKSegmentYCabinlFamilyMClass =
02467         FacBom<BookingClass>::instance().create (lMBookingClassKey);
02468     FacBomManager::addToListAndMap (lMktSINBKKSegmentYCabinlFamily,
02469                                     lMktSINBKKSegmentYCabinlFamilyMClass);
02470     FacBomManager::linkWithParent (lMktSINBKKSegmentYCabinlFamily,
02471                                     lMktSINBKKSegmentYCabinlFamilyMClass);
02472
02473     FacBomManager::addToListAndMap (lMktSINBKKSegmentYCabin,
02474                                     lMktSINBKKSegmentYCabinlFamilyMClass);
02475     FacBomManager::addToListAndMap (lMktSINBKKSegment,
02476                                     lMktSINBKKSegmentYCabinlFamilyMClass);
02477

```

```

02478     lMktSINBKKSegmentYCabinlFamilyMClass.setYield(500);
02479
02480     /* ===== */
02481
02482     // Step 1.0: O&D level
02483     // Create an O&D Date (CX1100/08-MAR-2010/SIN-BKK-CX12/08-MAR-2010/BKK-HKG) for CX's Inventory
02484     OnDString_T lMktCX_SINBKKOnDStr = "CX;1100,2010-Mar-08;SIN,BKK";
02485     OnDString_T lCXBKKHKGOnDStr = "CX;12,2010-Mar-08;BKK,HKG";
02486     lOnDStringList.clear();
02487     lOnDStringList.push_back (lMktCX_SINBKKOnDStr);
02488     lOnDStringList.push_back (lCXBKKHKGOnDStr);
02489
02490     lOnDDateKey = OnDDateKey(lOnDStringList);
02491     OnDDate& lCX_SINHKG_OnDDate =
02492         FacBom<OnDDate>::instance().create (lOnDDateKey);
02493     // Link to the inventory
02494     FacBomManager::addToListAndMap (lCXInv, lCX_SINHKG_OnDDate);
02495     FacBomManager::linkWithParent (lCXInv, lCX_SINHKG_OnDDate);
02496
02497     // Add the segments
02498     FacBomManager::addToListAndMap (lCX_SINHKG_OnDDate, lMktSINBKKSegment);
02499     FacBomManager::addToListAndMap (lCX_SINHKG_OnDDate, lBKKHKGSegment);
02500
02501     // Add total forecast info for cabin Y.
02502     lCX_SINHKG_OnDDate.setTotalForecast (lY, lWTP750Mean60StdDev6);
02503
02504     // Add demand info
02505     lCabinClassPairList.clear();
02506     lCabinClassPairList.push_back(lCC_YM1);
02507     lCabinClassPairList.push_back(lCC_YM2);
02508     lCX_SINHKG_OnDDate.setDemandInformation (lCabinClassPairList,
02509                                             lYield850Mean20StdDev2);
02510
02511     lCabinClassPairList.clear();
02512     lCabinClassPairList.push_back(lCC_YY1);
02513     lCabinClassPairList.push_back(lCC_YY2);
02514     lCX_SINHKG_OnDDate.setDemandInformation (lCabinClassPairList,
02515                                             lYield1200Mean10StdDev1);
02516
02517     /* ===== */
02518     // Create an O&D Date (CX1100/08-MAR-2010/SIN-BKK) for CX's Inventory
02519     lFullKeyList.clear();
02520     lFullKeyList.push_back (lMktCX_SINBKKFullKeyStr);
02521
02522     lOnDDateKey = OnDDateKey(lFullKeyList);
02523     OnDDate& lMktCX_SINBKK_OnDDate =
02524         FacBom<OnDDate>::instance().create (lOnDDateKey);
02525     // Link to the inventory
02526     FacBomManager::addToListAndMap (lCXInv, lMktCX_SINBKK_OnDDate);
02527     FacBomManager::linkWithParent (lCXInv, lMktCX_SINBKK_OnDDate);
02528
02529     // Add the segments
02530     FacBomManager::addToListAndMap (lMktCX_SINBKK_OnDDate, lMktSINBKKSegment);
02531
02532     // Demand info is not added for purely marketed O&Ds
02533     // Add demand info
02534     // lCabinClassPairList.clear();
02535     // lCabinClassPairList.push_back(lCC_YM1);
02536     // lMktCX_SINBKK_OnDDate.setDemandInformation (lCabinClassPairList, 500.0, 20.0, 1.0);
02537     /* ===== */
02538
02539     // Create an O&D Date (CX12/08-FEB-2010/BKK-HKG) for CX's Inventory
02540     lOnDStringList.clear();
02541     lOnDStringList.push_back (lCXBKKHKGOnDStr);
02542
02543     lOnDDateKey = OnDDateKey(lOnDStringList);
02544     OnDDate& lCX_BKKHKG_OnDDate =
02545         FacBom<OnDDate>::instance().create (lOnDDateKey);
02546     // Link to the inventory
02547     FacBomManager::addToListAndMap (lCXInv, lCX_BKKHKG_OnDDate);
02548     FacBomManager::linkWithParent (lCXInv, lCX_BKKHKG_OnDDate);
02549
02550     // Add the segments
02551     FacBomManager::addToListAndMap (lCX_BKKHKG_OnDDate, lBKKHKGSegment);
02552
02553     // Add total forecast info for cabin Y.
02554     lCX_BKKHKG_OnDDate.setTotalForecast (lY, lWTP400Mean60StdDev6);
02555
02556     // Add demand info
02557     lCabinClassPairList.clear();
02558     lCabinClassPairList.push_back(lCC_YM2);
02559     lCX_BKKHKG_OnDDate.setDemandInformation (lCabinClassPairList,
02560                                             lYield500Mean20StdDev1);
02561
02562     lCabinClassPairList.clear();
02563     lCabinClassPairList.push_back(lCC_YY2);
02564     const YieldDemandPair_T lYield700Mean10StdDev1 (lYield700, lMean10StdDev1 );

```

```

02565     lCX_BKKHKG_OnDDate.setDemandInformation (lCabinClassPairList,
02566                                             lYield700Mean10StdDev1);
02567
02568     /*=====
02569     =====*/
02570
02571     // Schedule:
02572     // SQ:
02573     // Step 1: flight period level
02574     // Create a flight period for SQ11:
02575     const DoWStruct lDoWSrtuct ("1111111");
02576     const Date_T lDateRangeStart (2010, boost::gregorian::Mar, 8);
02577     const Date_T lDateRangeEnd (2010, boost::gregorian::Mar, 9);
02578     const DatePeriod_T lDatePeriod (lDateRangeStart, lDateRangeEnd);
02579     const PeriodStruct lPeriodStruct (lDatePeriod, lDoWSrtuct);
02580
02581     lFlightNumber = FlightNumber_T (11);
02582
02583     FlightPeriodKey lFlightPeriodKey (lFlightNumber, lPeriodStruct);
02584
02585     FlightPeriod& lSQ11FlightPeriod =
02586         FacBom<FlightPeriod>::instance().create (lFlightPeriodKey);
02587     FacBomManager::addToListAndMap (lSQInv, lSQ11FlightPeriod);
02588     FacBomManager::linkWithParent (lSQInv, lSQ11FlightPeriod);
02589
02590     // Step 2: segment period level
02591     // Create a segment period for SIN-BKK:
02592
02593     SegmentPeriodKey lSegmentPeriodKey (lSIN, lBKK);
02594
02595     SegmentPeriod& lSINBKKSegmentPeriod =
02596         FacBom<SegmentPeriod>::instance().create (lSegmentPeriodKey);
02597     FacBomManager::addToListAndMap (lSQ11FlightPeriod, lSINBKKSegmentPeriod);
02598     FacBomManager::linkWithParent (lSQ11FlightPeriod, lSINBKKSegmentPeriod);
02599
02600     lSINBKKSegmentPeriod.setBoardingTime (10820);
02601     lSINBKKSegmentPeriod.setOffTime (11100);
02602     lSINBKKSegmentPeriod.setElapsedTime (10340);
02603     ClassList_String_T lYM ("YM");
02604     lSINBKKSegmentPeriod.addCabinBookingClassList (lY, lYM);
02605
02606     // CX:
02607     // Step 1: flight period level
02608     // Create a flight period for CX12:
02609     lFlightNumber = FlightNumber_T (12);
02610
02611     lFlightPeriodKey = FlightPeriodKey(lFlightNumber, lPeriodStruct);
02612
02613     FlightPeriod& lCX12FlightPeriod =
02614         FacBom<FlightPeriod>::instance().create (lFlightPeriodKey);
02615     FacBomManager::addToListAndMap (lCXInv, lCX12FlightPeriod);
02616     FacBomManager::linkWithParent (lCXInv, lCX12FlightPeriod);
02617
02618     // Step 2: segment period level
02619     // Create a segment period for BKK-HKG:
02620
02621     lSegmentPeriodKey = SegmentPeriodKey (lBKK, lHKG);
02622
02623     SegmentPeriod& lBKKHKGSegmentPeriod =
02624         FacBom<SegmentPeriod>::instance().create (lSegmentPeriodKey);
02625     FacBomManager::addToListAndMap (lCX12FlightPeriod, lBKKHKGSegmentPeriod);
02626     FacBomManager::linkWithParent (lCX12FlightPeriod, lBKKHKGSegmentPeriod);
02627
02628     lBKKHKGSegmentPeriod.setBoardingTime (11200);
02629     lBKKHKGSegmentPeriod.setOffTime (11540);
02630     lBKKHKGSegmentPeriod.setElapsedTime (10240);
02631     lBKKHKGSegmentPeriod.addCabinBookingClassList (lY, lYM);
02632
02633 }
02634
02635 // //////////////////////////////////////
02636 void CmdBomManager::buildPartnershipsSamplePricing (BomRoot& ioBomRoot) {
02637
02638     /*=====
02639     =====*/
02640
02641     // First airport pair SIN-BKK.
02642     // Set the airport-pair primary key.
02643     AirportPairKey lAirportPairKey ("SIN", "BKK");
02644
02645     // Create the AirportPairKey object and link it to the ioBomRoot object.
02646     AirportPair& lSINBKKAirportPair =
02647         FacBom<AirportPair>::instance().create (lAirportPairKey);
02648     FacBomManager::addToListAndMap (ioBomRoot, lSINBKKAirportPair);
02649     FacBomManager::linkWithParent (ioBomRoot, lSINBKKAirportPair);
02650
02651     // Set the fare date-period primary key.
02652     const Date_T lDateRangeStart (2010, boost::gregorian::Mar, 01);

```

```

02653     const Date_T lDateRangeEnd (2010, boost::gregorian::Mar, 31);
02654     const DatePeriod_T lDateRange (lDateRangeStart, lDateRangeEnd);
02655     const DatePeriodKey lDatePeriodKey (lDateRange);
02656
02657     // Create the DatePeriodKey object and link it to the PosChannel object.
02658     DatePeriod& lSINBKKDatePeriod =
02659         FacBom<DatePeriod>::instance().create (lDatePeriodKey);
02660     FacBomManager::addToListAndMap (lSINBKKAirportPair, lSINBKKDatePeriod);
02661     FacBomManager::linkWithParent (lSINBKKAirportPair, lSINBKKDatePeriod);
02662
02663     // Set the point-of-sale-channel primary key.
02664     PosChannelKey lPosChannelKey ("SIN", "IN");
02665
02666     // Create the PositionKey object and link it to the AirportPair object.
02667     PosChannel& lSINPosChannel =
02668         FacBom<PosChannel>::instance().create (lPosChannelKey);
02669     FacBomManager::addToListAndMap (lSINBKKDatePeriod, lSINPosChannel);
02670     FacBomManager::linkWithParent (lSINBKKDatePeriod, lSINPosChannel);
02671
02672     // Set the fare time-period primary key.
02673     const Time_T lTimeRangeStart (0, 0, 0);
02674     const Time_T lTimeRangeEnd (23, 0, 0);
02675     const TimePeriodKey lFareTimePeriodKey (lTimeRangeStart,
02676                                             lTimeRangeEnd);
02677
02678     // Create the TimePeriodKey and link it to the DatePeriod object.
02679     TimePeriod& lSINBKKFareTimePeriod =
02680         FacBom<TimePeriod>::instance().create (lFareTimePeriodKey);
02681     FacBomManager::addToListAndMap (lSINPosChannel, lSINBKKFareTimePeriod);
02682     FacBomManager::linkWithParent (lSINPosChannel, lSINBKKFareTimePeriod);
02683
02684     // Generate the FareRule
02685     const FareFeaturesKey lFareFeaturesKey (TRIP_TYPE_ONE_WAY,
02686                                             NO_ADVANCE_PURCHASE,
02687                                             SATURDAY_STAY,
02688                                             CHANGE_FEES,
02689                                             NON_REFUNDABLE,
02690                                             NO_STAY_DURATION);
02691
02692     // Create the FareFeaturesKey and link it to the TimePeriod object.
02693     FareFeatures& lSINBKKFareFeatures =
02694         FacBom<FareFeatures>::instance().create (lFareFeaturesKey);
02695     FacBomManager::addToListAndMap (lSINBKKFareTimePeriod,
02696                                     lSINBKKFareFeatures);
02697     FacBomManager::linkWithParent (lSINBKKFareTimePeriod, lSINBKKFareFeatures)
02698 ;
02699
02700     // Generate Segment Features and link them to their FareRule.
02701     AirlineCodeList_T lSQAirlineCodeList;
02702     lSQAirlineCodeList.push_back ("SQ");
02703
02704     ClassList_StringList_T lYClassCodeList;
02705     lYClassCodeList.push_back ("Y");
02706     const AirlineClassListKey lSQAirlineYClassListKey (lSQAirlineCodeList,
02707                                                         lYClassCodeList);
02708
02709     ClassList_StringList_T lMClassCodeList;
02710     lMClassCodeList.push_back ("M");
02711     const AirlineClassListKey lSQAirlineMClassListKey (lSQAirlineCodeList,
02712                                                         lMClassCodeList);
02713
02714     // Create the AirlineClassListKey and link it to the FareFeatures object.
02715     AirlineClassList& lSQAirlineYClassList =
02716         FacBom<AirlineClassList>::instance().
02717         create (lSQAirlineYClassListKey);
02718     lSQAirlineYClassList.setFare(700);
02719     FacBomManager::addToListAndMap (lSINBKKFareFeatures, lSQAirlineYClassList
02720 );
02721     FacBomManager::linkWithParent (lSINBKKFareFeatures, lSQAirlineYClassList);
02722
02723     AirlineClassList& lSQAirlineMClassList =
02724         FacBom<AirlineClassList>::instance().
02725         create (lSQAirlineMClassListKey);
02726     lSQAirlineMClassList.setFare(500);
02727     FacBomManager::addToListAndMap (lSINBKKFareFeatures, lSQAirlineMClassList
02728 );
02729     FacBomManager::linkWithParent (lSINBKKFareFeatures, lSQAirlineMClassList);
02730
02731     // Second airport pair BKK-HKG.
02732     // Set the airport-pair primary key.
02733     lAirportPairKey = AirportPairKey ("BKK", "HKG");
02734
02735     // Create the AirportPairKey object and link it to the ioBomRoot object.
02736     AirportPair& lBKKHKGAirportPair =
02737         FacBom<AirportPair>::instance().create (lAirportPairKey);
02738     FacBomManager::addToListAndMap (ioBomRoot, lBKKHKGAirportPair);

```



```

02734     FacBomManager::linkWithParent (ioBomRoot, lBKKHKGAirportPair);
02735
02736     // Set the fare date-period primary key.
02737     // Use the same as previously.
02738
02739     // Create the DatePeriodKey object and link it to the PosChannel object.
02740     DatePeriod& lBKKHKGDatePeriod =
02741         FacBom<DatePeriod>::instance().create (lDatePeriodKey);
02742     FacBomManager::addToListAndMap (lBKKHKGAirportPair, lBKKHKGDatePeriod);
02743     FacBomManager::linkWithParent (lBKKHKGAirportPair, lBKKHKGDatePeriod);
02744
02745     // Set the point-of-sale-channel primary key.
02746     lPosChannelKey = PosChannelKey("BKK","IN");
02747
02748     // Create the PositionKey object and link it to the AirportPair object.
02749     PosChannel& lBKKPosChannel =
02750         FacBom<PosChannel>::instance().create (lPosChannelKey);
02751     FacBomManager::addToListAndMap (lBKKHKGDatePeriod, lBKKPosChannel);
02752     FacBomManager::linkWithParent (lBKKHKGDatePeriod, lBKKPosChannel);
02753
02754     // Set the fare time-period primary key.
02755     // Use the same as previously.
02756
02757     // Create the TimePeriodKey and link it to the DatePeriod object.
02758     TimePeriod& lBKKHKGTimePeriod =
02759         FacBom<TimePeriod>::instance().create (lFareTimePeriodKey);
02760     FacBomManager::addToListAndMap (lBKKPosChannel, lBKKHKGTimePeriod);
02761     FacBomManager::linkWithParent (lBKKPosChannel, lBKKHKGTimePeriod);
02762
02763     // Generate the FareRule
02764     // Use the same key as previously.
02765
02766     // Create the FareFeaturesKey and link it to the TimePeriod object.
02767     FareFeatures& lBKKHKGTimePeriodFeatures =
02768         FacBom<FareFeatures>::instance().create (lFareFeaturesKey);
02769     FacBomManager::addToListAndMap (lBKKHKGTimePeriod,
lBKKHKGTimePeriodFeatures);
02770     FacBomManager::linkWithParent (lBKKHKGTimePeriod, lBKKHKGTimePeriodFeatures);
02771 ;
02772     // Generate Segment Features and link them to their FareRule.
02773     AirlineCodeList_T lCXAirlineCodeList;
02774     lCXAirlineCodeList.push_back ("CX");
02775
02776     const AirlineClassListKey lCXAirlineYClassListKey (lCXAirlineCodeList,
lYClassCodeList);
02777
02778     const AirlineClassListKey lCXAirlineMClassListKey (lCXAirlineCodeList,
lMClassCodeList);
02779
02780     // Create the AirlineClassListKey and link it to the FareFeatures object.
02781     AirlineClassList& lCXAirlineYClassList =
02782         FacBom<AirlineClassList>::instance().
create (lCXAirlineYClassListKey);
02783     lCXAirlineYClassList.setFare(700);
02784     FacBomManager::addToListAndMap (lBKKHKGTimePeriodFeatures, lCXAirlineYClassList
);
02785     FacBomManager::linkWithParent (lBKKHKGTimePeriodFeatures, lCXAirlineYClassList);
02786
02787     AirlineClassList& lCXAirlineMClassList =
02788         FacBom<AirlineClassList>::instance().
create (lCXAirlineMClassListKey);
02789     lCXAirlineMClassList.setFare(500);
02790     FacBomManager::addToListAndMap (lBKKHKGTimePeriodFeatures, lCXAirlineMClassList
);
02791     FacBomManager::linkWithParent (lBKKHKGTimePeriodFeatures, lCXAirlineMClassList);
02792
02793     //*****
02794     // Third airport pair SIN-HKG.
02795     // Set the airport-pair primary key.
02796     lAirportPairKey = AirportPairKey ("SIN", "HKG");
02797
02798     // Create the AirportPairKey object and link it to the ioBomRoot object.
02799     AirportPair& lSINHKGAirportPair =
02800         FacBom<AirportPair>::instance().create (lAirportPairKey);
02801     FacBomManager::addToListAndMap (ioBomRoot, lSINHKGAirportPair);
02802     FacBomManager::linkWithParent (ioBomRoot, lSINHKGAirportPair);
02803
02804     // Set the fare date-period primary key.
02805     // Use the same as previously.
02806
02807     // Create the DatePeriodKey object and link it to the PosChannel object.
02808     DatePeriod& lSINHKGDatePeriod =
02809         FacBom<DatePeriod>::instance().create (lDatePeriodKey);
02810     FacBomManager::addToListAndMap (lSINHKGAirportPair, lSINHKGDatePeriod);
02811     FacBomManager::linkWithParent (lSINHKGAirportPair, lSINHKGDatePeriod);
02812
02813
02814

```



```

02815 // Set the point-of-sale-channel primary key.
02816 lPosChannelKey = PosChannelKey("SIN","IN");
02817
02818 // Create the PositionKey object and link it to the AirportPair object.
02819 PosChannel& lOnDSINPosChannel =
02820     FacBom<PosChannel>::instance().create (lPosChannelKey);
02821 FacBomManager::addToListAndMap (lSINHKGDatePeriod, lOnDSINPosChannel);
02822 FacBomManager::linkWithParent (lSINHKGDatePeriod, lOnDSINPosChannel);
02823
02824 // Set the fare time-period primary key.
02825 // Use the same as previously.
02826
02827 // Create the TimePeriodKey and link it to the DatePeriod object.
02828 TimePeriod& lSINHKGFAreTimePeriod =
02829     FacBom<TimePeriod>::instance().create (lFAreTimePeriodKey);
02830 FacBomManager::addToListAndMap (lOnDSINPosChannel, lSINHKGFAreTimePeriod);
02831
02832 FacBomManager::linkWithParent (lOnDSINPosChannel, lSINHKGFAreTimePeriod);
02833
02834 // Generate the FareRule
02835 // Use the same key as previously.
02836
02837 // Create the FareFeaturesKey and link it to the TimePeriod object.
02838 FareFeatures& lSINHKGFAreFeatures =
02839     FacBom<FareFeatures>::instance().create (lFAreFeaturesKey);
02840 FacBomManager::addToListAndMap (lSINHKGFAreTimePeriod,
02841     lSINHKGFAreFeatures);
02842 FacBomManager::linkWithParent (lSINHKGFAreTimePeriod, lSINHKGFAreFeatures);
02843
02844 // Generate Segment Features and link them to their FareRule.
02845 AirlineCodeList_T lSQ_CXAirlineCodeList;
02846 lSQ_CXAirlineCodeList.push_back ("SQ");
02847 lSQ_CXAirlineCodeList.push_back ("CX");
02848
02849 ClassList_StringList_T lY_YClassCodeList;
02850 lY_YClassCodeList.push_back ("Y");
02851 lY_YClassCodeList.push_back ("Y");
02852 const AirlineClassListKey lSQ_CXAirlineYClassListKey (lSQ_CXAirlineCodeList,
02853     lY_YClassCodeList);
02854
02855 ClassList_StringList_T lM_MClassCodeList;
02856 lM_MClassCodeList.push_back ("M");
02857 lM_MClassCodeList.push_back ("M");
02858 const AirlineClassListKey lSQ_CXAirlineMClassListKey (lSQ_CXAirlineCodeList,
02859     lM_MClassList);
02860
02861 // Create the AirlineClassListKey and link it to the FareFeatures object.
02862 AirlineClassList& lSQ_CXAirlineYClassList =
02863     FacBom<AirlineClassList>::instance().
02864     create (lSQ_CXAirlineYClassListKey);
02865 lSQ_CXAirlineYClassList.setFare(1200);
02866 FacBomManager::addToListAndMap (lSINHKGFAreFeatures,
02867     lSQ_CXAirlineYClassList);
02868 FacBomManager::linkWithParent (lSINHKGFAreFeatures,
02869     lSQ_CXAirlineYClassList);
02870
02871 AirlineClassList& lSQ_CXAirlineMClassList =
02872     FacBom<AirlineClassList>::instance().
02873     create (lSQ_CXAirlineMClassListKey);
02874 lSQ_CXAirlineMClassList.setFare(850);
02875 FacBomManager::addToListAndMap (lSINHKGFAreFeatures,
02876     lSQ_CXAirlineMClassList);
02877 FacBomManager::linkWithParent (lSINHKGFAreFeatures,
02878     lSQ_CXAirlineMClassList);
02879
02880 //*****
02881 // Use the same airport pair, and date period for adding SQ SIN-BKK yields.
02882
02883 // Set the point-of-sale-channel primary key.
02884 lPosChannelKey = PosChannelKey(DEFAULT_POS, DEFAULT_CHANNEL);
02885
02886 // Create the PositionKey object and link it to the AirportPair object.
02887 PosChannel& lRAC_SINBKKPosChannel =
02888     FacBom<PosChannel>::instance().create (lPosChannelKey);
02889 FacBomManager::addToListAndMap (lSINBKKDatePeriod, lRAC_SINBKKPosChannel);
02890
02891 FacBomManager::linkWithParent (lSINBKKDatePeriod, lRAC_SINBKKPosChannel);
02892
02893 // Set the yield time-period primary key.
02894 const TimePeriodKey lYieldTimePeriodKey (lTimeRangeStart,
02895     lTimeRangeEnd);
02896
02897 // Create the TimePeriodKey and link it to the DatePeriod object.

```

```

02897     TimePeriod& lSINBKKYieldTimePeriod =
02898         FacBom<TimePeriod>::instance().create (lYieldTimePeriodKey);
02899     FacBomManager::addToListAndMap (lRAC_SINBKKPosChannel,
02900         lSINBKKYieldTimePeriod);
02901     FacBomManager::linkWithParent (lRAC_SINBKKPosChannel,
02902         lSINBKKYieldTimePeriod);
02903
02904     // Generate the YieldRule
02905     const YieldFeaturesKey lYieldFeaturesKey (TRIP_TYPE_ONE_WAY,
02906         CABIN_Y);
02907
02908     // Create the YieldFeaturesKey and link it to the TimePeriod object.
02909     YieldFeatures& lSINBKKYieldFeatures =
02910         FacBom<YieldFeatures>::instance().create (lYieldFeaturesKey);
02911     FacBomManager::addToListAndMap (lSINBKKYieldTimePeriod,
02912         lSINBKKYieldFeatures);
02913     FacBomManager::linkWithParent (lSINBKKYieldTimePeriod,
02914         lSINBKKYieldFeatures);
02915
02916     // Generate Segment Features and link them to their YieldRule.
02917     // Use the same key as previously.
02918
02919     // Create the AirlineClassListKey and link it to the YieldFeatures object.
02920     AirlineClassList& lRAC_SQAirlineYClassList =
02921         FacBom<AirlineClassList>::instance().
02922         create (lSQAirlineYClassListKey);
02923     lRAC_SQAirlineYClassList.setYield(700);
02924     FacBomManager::addToListAndMap (lSINBKKYieldFeatures,
02925         lRAC_SQAirlineYClassList);
02926     FacBomManager::linkWithParent (lSINBKKYieldFeatures,
02927         lRAC_SQAirlineYClassList);
02928
02929     AirlineClassList& lRAC_SQAirlineMClassList =
02930         FacBom<AirlineClassList>::instance().
02931         create (lSQAirlineMClassListKey);
02932     lRAC_SQAirlineMClassList.setYield(500);
02933     FacBomManager::addToListAndMap (lSINBKKYieldFeatures,
02934         lRAC_SQAirlineMClassList);
02935     FacBomManager::linkWithParent (lSINBKKYieldFeatures,
02936         lRAC_SQAirlineMClassList);
02937
02938     /*=====*/
02939     // Use the same airport pair, and date period for adding CX BKK-HKG yields.
02940
02941     // Set the point-of-sale-channel primary key.
02942     // Use the same as previously.
02943
02944     // Create the PositionKey object and link it to the AirportPair object.
02945     PosChannel& lRAC_BKKHKGPosChannel =
02946         FacBom<PosChannel>::instance().create (lPosChannelKey);
02947     FacBomManager::addToListAndMap (lBKKHKGDatePeriod, lRAC_BKKHKGPosChannel);
02948
02949     FacBomManager::linkWithParent (lBKKHKGDatePeriod, lRAC_BKKHKGPosChannel);
02950
02951     // Set the yield time-period primary key.
02952     // Use the same as previously.
02953
02954     // Create the TimePeriodKey and link it to the DatePeriod object.
02955     TimePeriod& lBKKHKGYieldTimePeriod =
02956         FacBom<TimePeriod>::instance().create (lYieldTimePeriodKey);
02957     FacBomManager::addToListAndMap (lRAC_BKKHKGPosChannel,
02958         lBKKHKGYieldTimePeriod);
02959     FacBomManager::linkWithParent (lRAC_BKKHKGPosChannel,
02960         lBKKHKGYieldTimePeriod);
02961
02962     // Generate the YieldRule
02963     // Use the same key as previously.
02964
02965     // Create the YieldFeaturesKey and link it to the TimePeriod object.
02966     YieldFeatures& lBKKHKGYieldFeatures =
02967         FacBom<YieldFeatures>::instance().create (lYieldFeaturesKey);
02968     FacBomManager::addToListAndMap (lBKKHKGYieldTimePeriod,
02969         lBKKHKGYieldFeatures);
02970     FacBomManager::linkWithParent (lBKKHKGYieldTimePeriod,
02971         lBKKHKGYieldFeatures);
02972
02973     // Generate Segment Features and link them to their YieldRule.
02974     // Use the same key as previously.
02975
02976     // Create the AirlineClassListKey and link it to the YieldFeatures object.
02977     AirlineClassList& lRAC_CXAirlineYClassList =
02978         FacBom<AirlineClassList>::instance().
02979         create (lCXAirlineYClassListKey);
02980     lRAC_CXAirlineYClassList.setYield(700);
02981     FacBomManager::addToListAndMap (lBKKHKGYieldFeatures,
02982         lRAC_CXAirlineYClassList);

```

```

02980     FacBomManager::linkWithParent (lBKKHKGyieldFeatures,
02981                                     lRAC_CXAirlineYClassList);
02982
02983     AirlineClassList& lRAC_CXAirlineMClassList =
02984         FacBom<AirlineClassList>::instance().
02985         create (lCXAirlineMClassListKey);
02986     lRAC_CXAirlineMClassList.setYield(500);
02987     FacBomManager::addToListAndMap (lBKKHKGyieldFeatures,
02988                                     lRAC_CXAirlineMClassList);
02989     FacBomManager::linkWithParent (lBKKHKGyieldFeatures,
02990                                     lRAC_CXAirlineMClassList);
02991
02992     /*=====*/
02993     // Use the same airport pair, and date period for SQ-CX SIN-HKG
02994
02995     // Set the point-of-sale-channel primary key.
02996     // Use the same as previously.
02997
02998     // Create the PositionKey object and link it to the AirportPair object.
02999     PosChannel& lRAC_SINHKGChannel =
03000         FacBom<PosChannel>::instance().create (lPosChannelKey);
03001     FacBomManager::addToListAndMap (lSINHKGDatePeriod, lRAC_SINHKGChannel);
03002     FacBomManager::linkWithParent (lSINHKGDatePeriod, lRAC_SINHKGChannel);
03003
03004     // Set the yield time-period primary key.
03005     // Use the same as previously.
03006
03007     // Create the TimePeriodKey and link it to the DatePeriod object.
03008     TimePeriod& lSINHKGyieldTimePeriod =
03009         FacBom<TimePeriod>::instance().create (lYieldTimePeriodKey);
03010     FacBomManager::addToListAndMap (lRAC_SINHKGChannel,
03011                                     lSINHKGyieldTimePeriod);
03012     FacBomManager::linkWithParent (lRAC_SINHKGChannel, lSINHKGyieldTimePeriod)
03013 ;
03014
03015     // Generate the YieldRule
03016     // Use the same key as previously.
03017
03018     // Create the YieldFeaturesKey and link it to the TimePeriod object.
03019     YieldFeatures& lSINHKGyieldFeatures =
03020         FacBom<YieldFeatures>::instance().create (lYieldFeaturesKey);
03021     FacBomManager::addToListAndMap (lSINHKGyieldTimePeriod,
03022                                     lSINHKGyieldFeatures);
03023     FacBomManager::linkWithParent (lSINHKGyieldTimePeriod,
03024                                     lSINHKGyieldFeatures);
03025
03026     // Generate Segment Features and link them to their YieldRule.
03027     // Use the same key as previously
03028
03029     // Create the AirlineClassListKey and link it to the YieldFeatures object.
03030     AirlineClassList& lRAC_SQ_CXAirlineYClassList =
03031         FacBom<AirlineClassList>::instance().
03032         create (lSQ_CXAirlineYClassListKey);
03033     lRAC_SQ_CXAirlineYClassList.setYield(1200);
03034     FacBomManager::addToListAndMap (lSINHKGyieldFeatures,
03035                                     lRAC_SQ_CXAirlineYClassList);
03036     FacBomManager::linkWithParent (lSINHKGyieldFeatures,
03037                                     lRAC_SQ_CXAirlineYClassList);
03038
03039     AirlineClassList& lRAC_SQ_CXAirlineMClassList =
03040         FacBom<AirlineClassList>::instance().
03041         create (lSQ_CXAirlineMClassListKey);
03042     lRAC_SQ_CXAirlineMClassList.setYield(850);
03043     FacBomManager::addToListAndMap (lSINHKGyieldFeatures,
03044                                     lRAC_SQ_CXAirlineMClassList);
03045     FacBomManager::linkWithParent (lSINHKGyieldFeatures,
03046                                     lRAC_SQ_CXAirlineMClassList);
03047
03048 }
03049 }
03050 }

```

### 33.539 stdair/command/CmdBomManager.hpp File Reference

```

#include <iosfwd>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/SampleType.hpp>
#include <stdair/bom/TravelSolutionTypes.hpp>
#include <stdair/command/CmdAbstract.hpp>

```

## Classes

- class [stdair::CmdBomManager](#)

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.540 CmdBomManager.hpp

```

00001 #ifndef __STDAIR_CMD_CMDBOMMANAGER_HPP
00002 #define __STDAIR_CMD_CMDBOMMANAGER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 // StdAir
00010 #include <stdair/stdair_inventory_types.hpp>
00011 #include <stdair/basic/SampleType.hpp>
00012 #include <stdair/bom/TravelSolutionTypes.hpp>
00013 #include <stdair/command/CmdAbstract.hpp>
00014
00015 namespace stdair {
00016
00017     class BomRoot;
00018     struct BookingRequestStruct;
00019
00020     class CmdBomManager : public CmdAbstract {
00021     //
00022     friend class STDAIR_Service;
00023     private:
00024
00025         // ////////////////////////////////// BOM initialisation support methods //////////////////////////////////
00026         static void buildSampleBom (BomRoot&);
00027
00028         static void buildSampleInventorySchedule (BomRoot&);
00029
00030         static void buildSampleInventoryScheduleForFareFamilies (BomRoot&);
00031
00032         static void buildCompleteDummyInventory (BomRoot&);
00033
00034         static void buildCompleteDummyInventoryForFareFamilies (BomRoot&);
00035
00036         static void buildDummyInventory (BomRoot&, const CabinCapacity_T&);
00037
00038         static void buildDummyLegSegmentAccesses (BomRoot&);
00039
00040         static void buildSamplePricing (BomRoot&);
00041
00042         static void buildSamplePricingForFareFamilies (BomRoot&);
00043
00044         static void buildSampleTravelSolutionForPricing (TravelSolutionList_T&);
00045
00046         static void buildSampleTravelSolutions (TravelSolutionList_T&);
00047
00048         static BookingRequestStruct buildSampleBookingRequest ();
00049
00050         static BookingRequestStruct buildSampleBookingRequestForCRS ();
00051
00052         static void buildPartnershipsSampleInventoryAndRM (BomRoot&);
00053
00054         static void buildPartnershipsSamplePricing (BomRoot&);
00055     };
00056 }
00057 #endif // __STDAIR_CMD_CMDBOMMANAGER_HPP

```

### 33.541 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

- [stdair](#)

*Handle on the StdAir library context.*

#### Functions

- `template<class Archive , class BOM_OBJECT1 , class BOM_OBJECT2 >`  
`void stdair::serialiseHelper (BOM_OBJECT1 &ioObject1, Archive &ioArchive, const unsigned int iFileVersion)`
- `template void stdair::BomRoot::serialize< ba::text\_oarchive > (ba::text_oarchive &, unsigned int)`
- `template void stdair::BomRoot::serialize< ba::text\_iarchive > (ba::text_iarchive &, unsigned int)`
- `template void stdair::Inventory::serialize< ba::text\_oarchive > (ba::text_oarchive &, unsigned int)`
- `template void stdair::Inventory::serialize< ba::text\_iarchive > (ba::text_iarchive &, unsigned int)`
- `template void stdair::FlightDate::serialize< ba::text\_oarchive > (ba::text_oarchive &, unsigned int)`
- `template void stdair::FlightDate::serialize< ba::text\_iarchive > (ba::text_iarchive &, unsigned int)`
- `template void stdair::SegmentDate::serialize< ba::text\_oarchive > (ba::text_oarchive &, unsigned int)`
- `template void stdair::SegmentDate::serialize< ba::text\_iarchive > (ba::text_iarchive &, unsigned int)`
- `template void stdair::SegmentCabin::serialize< ba::text\_oarchive > (ba::text_oarchive &, unsigned int)`
- `template void stdair::SegmentCabin::serialize< ba::text\_iarchive > (ba::text_iarchive &, unsigned int)`

### 33.542 CmdBomSerialiser.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/list.hpp>
00011 #include <boost/serialization/map.hpp>
```

```

00012 #include <boost/serialization/access.hpp>
00013 // StdAir
00014 #include <stdair/basic/BasConst_General.hpp>
00015 #include <stdair/basic/BasConst_Inventory.hpp>
00016 #include <stdair/bom/BomRoot.hpp>
00017 #include <stdair/bom/Inventory.hpp>
00018 #include <stdair/bom/FlightDate.hpp>
00019 #include <stdair/bom/SegmentDate.hpp>
00020 #include <stdair/bom/SegmentCabin.hpp>
00021 #include <stdair/bom/FareFamily.hpp>
00022 #include <stdair/bom/LegDate.hpp>
00023 #include <stdair/bom/LegCabin.hpp>
00024 #include <stdair/bom/Bucket.hpp>
00025 #include <stdair/factory/FacBomManager.hpp>
00026 #include <stdair/factory/FacBom.hpp>
00027 #include <stdair/command/CmdBomSerialiser.hpp>
00028 #include <stdair/service/Logger.hpp>
00029
00030 namespace stdair {
00031
00032 // //////////////////////////////////////
00033 template<class Archive, class BOM_OBJECT1, class BOM_OBJECT2>
00034 void serialiseHelper (BOM_OBJECT1& ioObject1, Archive& ioArchive,
00035                     const unsigned int iFileVersion) {
00036
00037     BomHolder<BOM_OBJECT2>* lBomHolder_ptr =
00038         FacBomManager::getBomHolderPtr<BOM_OBJECT2> (ioObject1);
00039
00040     if (lBomHolder_ptr == NULL) {
00041         lBomHolder_ptr = &FacBomManager::addBomHolder<BOM_OBJECT2> (ioObject1);
00042     }
00043     assert (lBomHolder_ptr != NULL);
00044
00045     //ioArchive.register_type (static_cast<Inventory*> (NULL));
00046     ioArchive & lBomHolder_ptr->_bomList;
00047     ioArchive & lBomHolder_ptr->_bomMap;
00048
00049     typedef typename BomHolder<BOM_OBJECT2>::BomList_T BomList_T;
00050     const BomList_T& lBomList = lBomHolder_ptr->_bomList;
00051     for (typename BomList_T::const_iterator itObject = lBomList.begin();
00052          itObject != lBomList.end(); ++itObject) {
00053         BOM_OBJECT2* lObject2_ptr = *itObject;
00054         assert (lObject2_ptr != NULL);
00055
00056         if (lObject2_ptr->getParent() == NULL) {
00057             FacBomManager::linkWithParent (ioObject1, *lObject2_ptr);
00058         }
00059     }
00060
00061     typedef typename BomHolder<BOM_OBJECT2>::BomMap_T BomMap_T;
00062     const BomMap_T& lBomMap = lBomHolder_ptr->_bomMap;
00063     if (lBomList.empty() == true && lBomMap.empty() == false) {
00064
00065         for (typename BomMap_T::const_iterator itObject = lBomMap.begin();
00066              itObject != lBomMap.end(); ++itObject) {
00067             BOM_OBJECT2* lObject2_ptr = itObject->second;
00068             assert (lObject2_ptr != NULL);
00069
00070             if (lObject2_ptr->getParent() == NULL) {
00071                 FacBomManager::linkWithParent (ioObject1, *lObject2_ptr);
00072             }
00073         }
00074     }
00075
00076 // //////////////////////////////////////
00077 void BomRoot::serialisationImplementationExport() const {
00078     std::ostringstream oStr;
00079     boost::archive::text_oarchive oa (oStr);
00080     oa << *this;
00081 }
00082
00083 // //////////////////////////////////////
00084 void BomRoot::serialisationImplementationImport() {
00085     std::istringstream iStr;
00086     boost::archive::text_iarchive ia (iStr);
00087     ia >> *this;
00088 }
00089
00090 // //////////////////////////////////////
00091 template<class Archive>
00092 void BomRoot::serialize (Archive& ioArchive,
00093                         const unsigned int iFileVersion) {
00094     // Serialise the key (by default, equal to " -- ROOT -- ")
00095     ioArchive & _key;
00096
00097     // Serialise the children of the BomRoot object, i.e., the

```

```

00139     // Inventory children
00140     stdair::serialiseHelper<Archive, BomRoot, Inventory> (*this, ioArchive,
00141                                                         iFileVersion);
00142 }
00143
00144 // //////////////////////////////////////
00145 void Inventory::serialisationImplementationExport() const {
00146     std::ostringstream oStr;
00147     boost::archive::text_oarchive oa (oStr);
00148     oa << *this;
00149 }
00150
00151 // //////////////////////////////////////
00152 void Inventory::serialisationImplementationImport() {
00153     std::istringstream iStr;
00154     boost::archive::text_iarchive ia (iStr);
00155     ia >> *this;
00156 }
00157
00158 // //////////////////////////////////////
00159 template<class Archive>
00160 void Inventory::serialize (Archive& ioArchive,
00161                             const unsigned int iFileVersion) {
00162     // Serialise the key (airline code)
00163     ioArchive & _key;
00164
00165     // Serialise the children of the Inventory object, i.e., the
00166     // FlightDate children
00167     stdair::serialiseHelper<Archive, Inventory, FlightDate> (*this, ioArchive,
00168                                                             iFileVersion);
00169 }
00170
00171 // //////////////////////////////////////
00172 void FlightDate::serialisationImplementationExport() const {
00173     std::ostringstream oStr;
00174     boost::archive::text_oarchive oa (oStr);
00175     oa << *this;
00176 }
00177
00178 // //////////////////////////////////////
00179 void FlightDate::serialisationImplementationImport() {
00180     std::istringstream iStr;
00181     boost::archive::text_iarchive ia (iStr);
00182     ia >> *this;
00183 }
00184
00185 // //////////////////////////////////////
00186 template<class Archive>
00187 void FlightDate::serialize (Archive& ioArchive,
00188                             const unsigned int iFileVersion) {
00189     ioArchive & _key;
00190 }
00191
00192 // //////////////////////////////////////
00193 void SegmentDate::serialisationImplementationExport() const {
00194     std::ostringstream oStr;
00195     boost::archive::text_oarchive oa (oStr);
00196     oa << *this;
00197 }
00198
00199 // //////////////////////////////////////
00200 void SegmentDate::serialisationImplementationImport() {
00201     std::istringstream iStr;
00202     boost::archive::text_iarchive ia (iStr);
00203     ia >> *this;
00204 }
00205
00206 // //////////////////////////////////////
00207 template<class Archive>
00208 void SegmentDate::serialize (Archive& ioArchive,
00209                             const unsigned int iFileVersion) {
00210     ioArchive & _key;
00211 }
00212
00213 // //////////////////////////////////////
00214 void SegmentCabin::serialisationImplementationExport() const {
00215     std::ostringstream oStr;
00216     boost::archive::text_oarchive oa (oStr);
00217     oa << *this;
00218 }
00219
00220 // //////////////////////////////////////
00221 void SegmentCabin::serialisationImplementationImport() {
00222     std::istringstream iStr;
00223     boost::archive::text_iarchive ia (iStr);
00224     ia >> *this;
00225 }

```

```

00226
00227 ///////////////////////////////////////////////////////////////////
00228 template<class Archive>
00229 void SegmentCabin::serialize (Archive& ioArchive,
00230                               const unsigned int iFileVersion) {
00231     ioArchive & _key;
00232 }
00233
00234 ///////////////////////////////////////////////////////////////////
00235 // Explicit template instantiations
00236 namespace ba = boost::archive;
00237 template void BomRoot::serialize<ba::text_oarchive> (ba::text_oarchive&,
00238                                                       unsigned int);
00239 template void BomRoot::serialize<ba::text_iarchive> (ba::text_iarchive&,
00240                                                       unsigned int);
00241 template void Inventory::serialize<ba::text_oarchive> (ba::text_oarchive&,
00242                                                         unsigned int);
00243 template void Inventory::serialize<ba::text_iarchive> (ba::text_iarchive&,
00244                                                         unsigned int);
00245 template void FlightDate::serialize<ba::text_oarchive> (ba::text_oarchive&,
00246                                                         unsigned int);
00247 template void FlightDate::serialize<ba::text_iarchive> (ba::text_iarchive&,
00248                                                         unsigned int);
00249 template void SegmentDate::serialize<ba::text_oarchive> (ba::text_oarchive&,
00250                                                         unsigned int);
00251 template void SegmentDate::serialize<ba::text_iarchive> (ba::text_iarchive&,
00252                                                         unsigned int);
00253 template void SegmentCabin::serialize<ba::text_oarchive> (ba::text_oarchive&,
00254                                                           unsigned int);
00255 template void SegmentCabin::serialize<ba::text_iarchive> (ba::text_iarchive&,
00256                                                           unsigned int);
00257 ///////////////////////////////////////////////////////////////////
00258
00259 }

```

### 33.543 stdair/command/CmdBomSerialiser.hpp File Reference

```

#include <iosfwd>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/TravelSolutionTypes.hpp>
#include <stdair/command/CmdAbstract.hpp>

```

#### Classes

- class [stdair::CmdBomSerialiser](#)

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.544 CmdBomSerialiser.hpp

```

00001 #ifndef __STDAIR_CMD_CMDBOMSERIALISER_HPP
00002 #define __STDAIR_CMD_CMDBOMSERIALISER_HPP
00003
00004 ///////////////////////////////////////////////////////////////////
00005 // Import section
00006 ///////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 // StdAir
00010 #include <stdair/stdair_inventory_types.hpp>
00011 #include <stdair/bom/TravelSolutionTypes.hpp>
00012 #include <stdair/command/CmdAbstract.hpp>
00013
00014 namespace stdair {
00015
00017     class BomRoot;
00018     struct BookingRequestStruct;
00019

```



```

00020
00025 class CmdBomSerialiser : public CmdAbstract {
00026 public:
00027
00028     // ////////////////////////////////// BOM serialisation support methods //////////////////////////////////
00029     //
00030 };
00031 }
00032 #endif // __STDAIR_CMD_CMDBOMSERIALISER_HPP

```

### 33.545 stdair/command/CmdCloneBomManager.cpp File Reference

#### 33.546 CmdCloneBomManager.cpp

```

00001
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <cassert>
00010 #include <sstream>
00011 // StdAir
00012 #include <stdair/factory/FacBomManager.hpp>
00013 #include <stdair/factory/FacCloneBom.hpp>
00014 #include <stdair/command/CmdCloneBomManager.hpp>
00015 #include <stdair/service/Logger.hpp>
00016 #include <stdair/bom/BomRetriever.hpp>
00017
00018 namespace stdair {
00019
00020     // //////////////////////////////////////
00021     void CmdCloneBomManager::cloneBomRoot (const BomRoot& iBomRoot,
00022                                             BomRoot& ioCloneBomRoot) {
00023
00024         // Check whether there are Inventory objects
00025         const bool hasInventoryList = BomManager::hasList<Inventory> (iBomRoot);
00026         if (hasInventoryList == true) {
00027
00028             // Browse the inventories
00029             const InventoryList_T& lInventoryList =
00030                 BomManager::getList<Inventory> (iBomRoot);
00031             for (InventoryList_T::const_iterator itInv = lInventoryList.begin();
00032                  itInv != lInventoryList.end(); ++itInv) {
00033                 const Inventory* lInv_ptr = *itInv;
00034                 assert (lInv_ptr != NULL);
00035
00036                 // Clone the current inventory
00037                 Inventory& lCloneInventory = cloneInventory (*lInv_ptr, ioCloneBomRoot);
00038                 FacBomManager::addToListAndMap (ioCloneBomRoot, lCloneInventory);
00039                 FacBomManager::linkWithParent (ioCloneBomRoot, lCloneInventory);
00040             }
00041
00042             // Check whether there are Airport Pair objects
00043             const bool hasAirportPairList =
00044                 BomManager::hasList<AirportPair> (iBomRoot);
00045             if (hasAirportPairList == true) {
00046
00047                 // Browse the airport pairs
00048                 const AirportPairList_T& lAirportPairList =
00049                     BomManager::getList<AirportPair> (iBomRoot);
00050                 for (AirportPairList_T::const_iterator itAirportPair =
00051                      lAirportPairList.begin();
00052                     itAirportPair != lAirportPairList.end(); ++itAirportPair) {
00053                     const AirportPair* lAirportPair_ptr = *itAirportPair;
00054                     assert (lAirportPair_ptr != NULL);
00055
00056                     // Clone the current airport pair
00057                     AirportPair& lCloneAirportPair = cloneAirportPair (*lAirportPair_ptr);
00058                     FacBomManager::addToListAndMap (ioCloneBomRoot, lCloneAirportPair);
00059                     FacBomManager::linkWithParent (ioCloneBomRoot, lCloneAirportPair);
00060                 }
00061             }
00062
00063             // //////////////////////////////////////
00064             Inventory& CmdCloneBomManager::cloneInventory (const Inventory& iInventory,
00065                                                             BomRoot& ioCloneBomRoot) {
00066
00067                 Inventory& lCloneInventory =
00068                     FacCloneBom<Inventory>::instance().clone (iInventory);
00069
00070                 // Check whether there are FlightDate objects
00071
00072
00073
00074
00075
00076
00077
00078
00079
00080

```

```

00081     const bool hasFlightDateList = BomManager::hasList<FlightDate> (iInventory);
00082     if (hasFlightDateList == true) {
00083         // Browse the flight-dates
00084         const FlightDateList_T& lFlightDateList =
00085             BomManager::getList<FlightDate> (iInventory);
00086         for (FlightDateList_T::const_iterator itFD = lFlightDateList.begin();
00087             itFD != lFlightDateList.end(); ++itFD) {
00088             const FlightDate* lFD_ptr = *itFD;
00089             assert (lFD_ptr != NULL);
00090
00091             // Clone the current flight-date
00092             FlightDate& lCloneFD = cloneFlightDate (*lFD_ptr);
00093             FacBomManager::addToListAndMap (lCloneInventory, lCloneFD);
00094             FacBomManager::linkWithParent (lCloneInventory, lCloneFD);
00095         }
00096     }
00097
00098     // Check if the inventory contains a list of partners
00099     const bool hasPartnerList = BomManager::hasList<Inventory> (iInventory);
00100     if (hasPartnerList == true) {
00101
00102         // Browse the partner's inventories
00103         const InventoryList_T& lPartnerInventoryList =
00104             BomManager::getList<Inventory> (iInventory);
00105
00106         for (InventoryList_T::const_iterator itInv =
00107             lPartnerInventoryList.begin();
00108             itInv != lPartnerInventoryList.end(); ++itInv) {
00109             const Inventory* lInv_ptr = *itInv;
00110             assert (lInv_ptr != NULL);
00111
00112             // Clone the current partnership inventory
00113             Inventory& lClonePartnerInventory = cloneInventory (*lInv_ptr,
00114                                                                 ioCloneBomRoot);
00115             FacBomManager::addToListAndMap (lCloneInventory,
00116                                             lClonePartnerInventory);
00117             FacBomManager::linkWithParent (lCloneInventory,
00118                                             lClonePartnerInventory);
00119         }
00120     }
00121
00122     // Check whether there are O&D date objects
00123     const bool hasOnDList = BomManager::hasList<OnDDate> (iInventory);
00124     if (hasOnDList == true){
00125
00126         //Browse the O&Ds
00127         const OnDDateList_T& lOnDDateList =
00128             BomManager::getList<OnDDate> (iInventory);
00129
00130         for (OnDDateList_T::const_iterator itOnD = lOnDDateList.begin();
00131             itOnD != lOnDDateList.end(); ++itOnD) {
00132             const OnDDate* lOnDDate_ptr = *itOnD;
00133             assert (lOnDDate_ptr != NULL);
00134
00135             // Clone the current O&D date
00136             OnDDate& lCloneOnDDate = cloneOnDDate (*lOnDDate_ptr);
00137             FacBomManager::addToListAndMap (lCloneInventory, lCloneOnDDate);
00138             FacBomManager::linkWithParent (lCloneInventory, lCloneOnDDate);
00139         }
00140     }
00141
00142     // Check whether there are Flight Period objects
00143     const bool hasFlightPeriodList =
00144         BomManager::hasList<FlightPeriod> (iInventory);
00145     if (hasFlightPeriodList == true) {
00146
00147         // Browse the flight-periods
00148         const FlightPeriodList_T& lFlightPeriodList =
00149             BomManager::getList<FlightPeriod> (iInventory);
00150         for (FlightPeriodList_T::const_iterator itFlightPeriod =
00151             lFlightPeriodList.begin();
00152             itFlightPeriod != lFlightPeriodList.end(); ++itFlightPeriod) {
00153             const FlightPeriod* lFlightPeriod_ptr = *itFlightPeriod;
00154             assert (lFlightPeriod_ptr != NULL);
00155
00156             // Clone the current flight period
00157             FlightPeriod& lCloneFlightPeriod = cloneFlightPeriod (*lFlightPeriod_ptr);
00158             FacBomManager::addToListAndMap (lCloneInventory, lCloneFlightPeriod);
00159             FacBomManager::linkWithParent (lCloneInventory, lCloneFlightPeriod);
00160         }
00161     }
00162
00163     // Check whether there is an airline feature object
00164     const AirlineFeature* lAirlineFeature_ptr =
00165         BomManager::getObjectPtr<AirlineFeature,Inventory> (iInventory,
00166                                                             iInventory.getAirlineCode());
00167     if (lAirlineFeature_ptr != NULL) {

```

```

00168         // Clone the current airline feature object
00169         AirlineFeature& lCloneAirlineFeature =
00170             cloneAirlineFeature (*lAirlineFeature_ptr);
00171         FacBomManager::setAirlineFeature (lCloneInventory,
lCloneAirlineFeature);
00172         FacBomManager::linkWithParent (lCloneInventory, lCloneAirlineFeature);
00173         // Link the airline feature object with the top of the BOM tree
00174         FacBomManager::addToListAndMap (ioCloneBomRoot, lCloneAirlineFeature);
00175     }
00176
00177     return lCloneInventory;
00178 }
00179
00180 // //////////////////////////////////////
00181 AirlineFeature& CmdCloneBomManager::
00182 cloneAirlineFeature (const AirlineFeature& iAirlineFeature) {
00183
00184     AirlineFeature& lCloneAirlineFeature =
00185         FacCloneBom<AirlineFeature>::instance().
clone (iAirlineFeature);
00189
00190     return lCloneAirlineFeature;
00191 }
00192
00193 // //////////////////////////////////////
00194 OnDDate& CmdCloneBomManager::cloneOnDDate (const OnDDate& iOnDDate) {
00195
00196     OnDDate& lCloneOnDDate =
00197         FacCloneBom<OnDDate>::instance().clone (iOnDDate);
00198
00199     return lCloneOnDDate;
00200 }
00201
00202 // //////////////////////////////////////
00203 FlightDate& CmdCloneBomManager::
cloneFlightDate (const FlightDate& iFlightDate) {
00204
00205     FlightDate& lCloneFlightDate =
00206         FacCloneBom<FlightDate>::instance().clone (iFlightDate);
00207
00208     // Check whether there are LegDate objects
00209     const bool hasLegDateList = BomManager::hasList<LegDate> (iFlightDate);
00210     if (hasLegDateList == true) {
00211
00212         // Browse the leg-dates
00213         const LegDateList_T& lLegDateList =
00214             BomManager::getList<LegDate> (iFlightDate);
00215         for (LegDateList_T::const_iterator itLD = lLegDateList.begin();
00216             itLD != lLegDateList.end(); ++itLD) {
00217             const LegDate* lLD_ptr = *itLD;
00218             assert (lLD_ptr != NULL);
00219
00220             // Clone the current leg-date
00221             LegDate& lCloneLegDate = cloneLegDate (*lLD_ptr);
00222             FacBomManager::addToListAndMap (lCloneFlightDate, lCloneLegDate);
00223             FacBomManager::linkWithParent (lCloneFlightDate, lCloneLegDate);
00224         }
00225     }
00226
00227     // Check whether there are SegmentDate objects
00228     const bool hasSegmentDateList =
00229         BomManager::hasList<SegmentDate> (iFlightDate);
00230     if (hasSegmentDateList == true) {
00231
00232         // Browse the segment-dates
00233         const SegmentDateList_T& lSegmentDateList =
00234             BomManager::getList<SegmentDate> (iFlightDate);
00235         for (SegmentDateList_T::const_iterator itSD = lSegmentDateList.begin();
00236             itSD != lSegmentDateList.end(); ++itSD) {
00237             const SegmentDate* lSD_ptr = *itSD;
00238             assert (lSD_ptr != NULL);
00239
00240             // Clone the current segment-date
00241             SegmentDate& lCloneSegmentDate = cloneSegmentDate (*lSD_ptr);
00242             FacBomManager::addToListAndMap (lCloneFlightDate, lCloneSegmentDate);
00243             FacBomManager::linkWithParent (lCloneFlightDate, lCloneSegmentDate);
00244         }
00245     }
00246
00247     return lCloneFlightDate;
00248 }
00249
00250 // //////////////////////////////////////
00251 LegDate& CmdCloneBomManager::cloneLegDate (const LegDate& iLegDate) {
00252

```

```

00265     LegDate& lCloneLegDate =
00266         FacCloneBom<LegDate>::instance().clone (iLegDate);
00267
00268     // Check whether there are LegCabin objects
00269     const bool hasLegCabinList = BomManager::hasList<LegCabin> (iLegDate);
00270     if (hasLegCabinList == true) {
00271         // Browse the leg-cabins
00272         const LegCabinList_T& lLegCabinList =
00273             BomManager::getList<LegCabin> (iLegDate);
00274         for (LegCabinList_T::const_iterator itLC = lLegCabinList.begin();
00275             itLC != lLegCabinList.end(); ++itLC) {
00276             const LegCabin* lLC_ptr = *itLC;
00277             assert (lLC_ptr != NULL);
00278
00279             // Clone the current leg-cabin
00280             LegCabin& lCloneLegCabin = cloneLegCabin (*lLC_ptr);
00281             FacBomManager::addToListAndMap (lCloneLegDate, lCloneLegCabin);
00282             FacBomManager::linkWithParent (lCloneLegDate, lCloneLegCabin);
00283         }
00284     }
00285
00286     return lCloneLegDate;
00287 }
00288
00289 // //////////////////////////////////////
00290 LegCabin& CmdCloneBomManager::cloneLegCabin (const LegCabin& iLegCabin) {
00291
00292     LegCabin& lCloneLegCabin =
00293         FacCloneBom<LegCabin>::instance().clone (iLegCabin);
00294
00295     // Check whether there are Bucket objects
00296     const bool hasBucketList = BomManager::hasList<Bucket> (iLegCabin);
00297     if (hasBucketList == true) {
00298         // Browse the buckets
00299         const BucketList_T& lBucketList =
00300             BomManager::getList<Bucket> (iLegCabin);
00301         for (BucketList_T::const_iterator itBucket = lBucketList.begin();
00302             itBucket != lBucketList.end(); ++itBucket) {
00303             const Bucket* lBucket_ptr = *itBucket;
00304             assert (lBucket_ptr != NULL);
00305
00306             // Clone the current bucket
00307             Bucket& lCloneBucket = cloneBucket (*lBucket_ptr);
00308             FacBomManager::addToListAndMap (lCloneLegCabin, lCloneBucket);
00309             FacBomManager::linkWithParent (lCloneLegCabin, lCloneBucket);
00310         }
00311     }
00312
00313     return lCloneLegCabin;
00314 }
00315
00316 // //////////////////////////////////////
00317 Bucket& CmdCloneBomManager::cloneBucket (const Bucket& iBucket) {
00318
00319     Bucket& lCloneBucket =
00320         FacCloneBom<Bucket>::instance().clone (iBucket);
00321
00322     return lCloneBucket;
00323 }
00324
00325 // //////////////////////////////////////
00326 SegmentDate& CmdCloneBomManager::
00327 cloneSegmentDate (const SegmentDate& iSegmentDate) {
00328
00329     SegmentDate& lCloneSegmentDate =
00330         FacCloneBom<SegmentDate>::instance().
00331         clone (iSegmentDate);
00332
00333     // Check whether there are SegmentCabin objects
00334     const bool hasSegmentCabinList =
00335         BomManager::hasList<SegmentCabin> (iSegmentDate);
00336     if (hasSegmentCabinList == true) {
00337         // Browse the segment-cabins
00338         const SegmentCabinList_T& lSegmentCabinList =
00339             BomManager::getList<SegmentCabin> (iSegmentDate);
00340         for (SegmentCabinList_T::const_iterator itSC = lSegmentCabinList.begin();
00341             itSC != lSegmentCabinList.end(); ++itSC) {
00342             const SegmentCabin* lSC_ptr = *itSC;
00343             assert (lSC_ptr != NULL);
00344
00345             // Clone the current segment-cabin
00346             SegmentCabin& lCloneSegmentCabin = cloneSegmentCabin (*lSC_ptr);
00347             FacBomManager::addToListAndMap (lCloneSegmentDate, lCloneSegmentCabin);
00348             FacBomManager::linkWithParent (lCloneSegmentDate, lCloneSegmentCabin);
00349         }
00350     }
00351
00352     linkBookingClassesWithSegment (lCloneSegmentDate,

```

```

00359                                     lCloneSegmentCabin);
00360
00361     }
00362 }
00363     return lCloneSegmentDate;
00364 }
00365
00366 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00367 void CmdCloneBomManager::
00368 linkBookingClassesWithSegment (SegmentDate& iCloneSegmentDate,
00369                               SegmentCabin& iCloneSegmentCabin) {
00370
00371     // Browse the fare families to link the booking-classes to the
00372     // segment-cabin and to the segment-date
00373     const bool hasFareFamilyList =
00374         BomManager::hasList<FareFamily> (iCloneSegmentCabin);
00375     if (hasFareFamilyList == true) {
00376         const FareFamilyList_T& lCloneFFList =
00377             BomManager::getList<FareFamily> (iCloneSegmentCabin);
00378         for (FareFamilyList_T::const_iterator itCloneFF = lCloneFFList.begin();
00379              itCloneFF != lCloneFFList.end(); ++itCloneFF) {
00380             const FareFamily* lCloneFF_ptr = *itCloneFF;
00381             assert (lCloneFF_ptr != NULL);
00382
00383             // Browse the list of booking classes
00384             const bool hasBookingClasslist =
00385                 BomManager::hasList<BookingClass> (*lCloneFF_ptr);
00386             if (hasBookingClasslist == true) {
00387                 const BookingClassList_T& lCloneBCList =
00388                     BomManager::getList<BookingClass> (*lCloneFF_ptr);
00389                 for (BookingClassList_T::const_iterator itCloneBC =
00390                      lCloneBCList.begin();
00391                     itCloneBC != lCloneBCList.end(); ++itCloneBC) {
00392                     const BookingClass* lCloneBC_ptr = *itCloneBC;
00393                     assert (lCloneBC_ptr != NULL);
00394
00395                     // Link the booking-class to the segment-cabin
00396                     stdair::FacBomManager::addToListAndMap (
00397                         iCloneSegmentCabin,
00398                         *lCloneBC_ptr);
00399
00400                     // Link the booking-class to the segment-date
00401                     stdair::FacBomManager::addToListAndMap (iCloneSegmentDate,
00402                                                             *lCloneBC_ptr);
00403                 }
00404             }
00405         }
00406     }
00407
00408 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00409 SegmentCabin& CmdCloneBomManager::
00410 cloneSegmentCabin (const SegmentCabin& iSegmentCabin) {
00411
00412     SegmentCabin& lCloneSegmentCabin =
00413         FacCloneBom<SegmentCabin>::instance().
00414         clone (iSegmentCabin);
00415
00416     // Check whether there are fare family objects
00417     const bool hasFareFamilyList =
00418         BomManager::hasList<FareFamily> (iSegmentCabin);
00419     if (hasFareFamilyList == true) {
00420         // Browse the fare families
00421         const FareFamilyList_T& lFareFamilyList =
00422             BomManager::getList<FareFamily> (iSegmentCabin);
00423         for (FareFamilyList_T::const_iterator itFF = lFareFamilyList.begin();
00424              itFF != lFareFamilyList.end(); ++itFF) {
00425             const FareFamily* lFF_ptr = *itFF;
00426             assert (lFF_ptr != NULL);
00427
00428             // Clone the current fare-family
00429             FareFamily& lCloneFareFamily = cloneFareFamily (*lFF_ptr);
00430             FacBomManager::addToListAndMap (lCloneSegmentCabin, lCloneFareFamily);
00431         }
00432     }
00433     FacBomManager::linkWithParent (lCloneSegmentCabin, lCloneFareFamily);
00434 }
00435
00436 return lCloneSegmentCabin;
00437 }
00438
00439 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00440 FareFamily& CmdCloneBomManager::
00441 cloneFareFamily (const FareFamily& iFareFamily) {
00442     FareFamily& lCloneFareFamily =
00443         FacCloneBom<FareFamily>::instance().clone (iFareFamily);

```

```

00448
00449 // Check whether there are booking classes objects
00450 const bool hasBookingClassList =
00451     BomManager::hasList<BookingClass> (iFareFamily);
00452 if (hasBookingClassList == true) {
00453     // Browse the list of booking classes
00454     const BookingClassList_T& lBookingClassList =
00455         BomManager::getList<BookingClass> (iFareFamily);
00456     for (BookingClassList_T::const_iterator itBookingClass =
00457         lBookingClassList.begin();
00458         itBookingClass != lBookingClassList.end(); ++itBookingClass) {
00459         const BookingClass* lBC_ptr = *itBookingClass;
00460         assert (lBC_ptr != NULL);
00461
00462         // Clone the current booking class
00463         BookingClass& lCloneBookingClass = cloneBookingClass (*lBC_ptr);
00464         FacBomManager::addToListAndMap (lCloneFareFamily, lCloneBookingClass)
00465     ;
00466         FacBomManager::linkWithParent (lCloneFareFamily, lCloneBookingClass);
00467     }
00468 }
00469 return lCloneFareFamily;
00470 }
00471
00472 // //////////////////////////////////////
00473 BookingClass& CmdCloneBomManager::
00474 cloneBookingClass (const BookingClass& iBookingClass) {
00475
00476     BookingClass& lCloneBookingClass =
00477         FacCloneBom<BookingClass>::instance().
00478         clone (iBookingClass);
00479
00480     return lCloneBookingClass;
00481 }
00482
00483 // //////////////////////////////////////
00484 AirportPair& CmdCloneBomManager::
00485 cloneAirportPair (const AirportPair& iAirportPair) {
00486
00487     AirportPair& lCloneAirportPair =
00488         FacCloneBom<AirportPair>::instance().
00489         clone (iAirportPair);
00490
00491     // Check whether there are date-period objects
00492     const bool hasDatePeriodList =
00493         BomManager::hasList<DatePeriod> (iAirportPair);
00494     if (hasDatePeriodList == true) {
00495         // Browse the date-periods
00496         const DatePeriodList_T& lDatePeriodList =
00497             BomManager::getList<DatePeriod> (iAirportPair);
00498         for (DatePeriodList_T::const_iterator itDatePeriod =
00499             lDatePeriodList.begin();
00500             itDatePeriod != lDatePeriodList.end(); ++itDatePeriod) {
00501             const DatePeriod* lDatePeriod_ptr = *itDatePeriod;
00502             assert (lDatePeriod_ptr != NULL);
00503
00504             // Clone the current date-period
00505             DatePeriod& lCloneDatePeriod = cloneDatePeriod (*lDatePeriod_ptr);
00506             FacBomManager::addToListAndMap (lCloneAirportPair, lCloneDatePeriod);
00507             FacBomManager::linkWithParent (lCloneAirportPair, lCloneDatePeriod);
00508         }
00509     }
00510     return lCloneAirportPair;
00511 }
00512
00513 // //////////////////////////////////////
00514 DatePeriod& CmdCloneBomManager::
00515 cloneDatePeriod (const DatePeriod& iDatePeriod) {
00516
00517     DatePeriod& lCloneDatePeriod =
00518         FacCloneBom<DatePeriod>::instance().clone (iDatePeriod);
00519
00520     // Check whether there are pos-channel objects
00521     const bool hasPosChannelList =
00522         BomManager::hasList<PosChannel> (iDatePeriod);
00523     if (hasPosChannelList == true) {
00524         // Browse the pos-channels
00525         const PosChannelList_T& lPosChannelList =
00526             BomManager::getList<PosChannel> (iDatePeriod);
00527         for (PosChannelList_T::const_iterator itPosChannel =
00528             lPosChannelList.begin();
00529             itPosChannel != lPosChannelList.end(); ++itPosChannel) {
00530             const PosChannel* lPosChannel_ptr = *itPosChannel;
00531             assert (lPosChannel_ptr != NULL);
00532
00533         }
00534     }
00535 }

```

```

00541         // Clone the current pos-channel
00542         PosChannel& lClonePosChannel = clonePosChannel (*lPosChannel_ptr);
00543         FacBomManager::addToListAndMap (lCloneDatePeriod, lClonePosChannel);
00544         FacBomManager::linkWithParent (lCloneDatePeriod, lClonePosChannel);
00545     }
00546 }
00547
00548     return lCloneDatePeriod;
00549 }
00550
00551 // //////////////////////////////////////
00552 PosChannel& CmdCloneBomManager::
00553 clonePosChannel (const PosChannel& iPosChannel) {
00554
00555     PosChannel& lClonePosChannel =
00556         FacCloneBom<PosChannel>::instance().clone (iPosChannel);
00557
00558     // Check whether there are time-period objects
00559     const bool hasTimePeriodList =
00560         BomManager::hasList<TimePeriod> (iPosChannel);
00561     if (hasTimePeriodList == true) {
00562         // Browse the time-periods
00563         const TimePeriodList_T& lTimePeriodList =
00564             BomManager::getList<TimePeriod> (iPosChannel);
00565         for (TimePeriodList_T::const_iterator itTimePeriod =
00566             lTimePeriodList.begin();
00567             itTimePeriod != lTimePeriodList.end(); ++itTimePeriod) {
00568             const TimePeriod* lTimePeriod_ptr = *itTimePeriod;
00569             assert (lTimePeriod_ptr != NULL);
00570
00571             // Clone the current time-period
00572             TimePeriod& lCloneTimePeriod = cloneTimePeriod (*lTimePeriod_ptr);
00573             FacBomManager::addToListAndMap (lClonePosChannel, lCloneTimePeriod);
00574             FacBomManager::linkWithParent (lClonePosChannel, lCloneTimePeriod);
00575         }
00576     }
00577
00578     return lClonePosChannel;
00579 }
00580
00581 // //////////////////////////////////////
00582 TimePeriod& CmdCloneBomManager::
00583 cloneTimePeriod (const TimePeriod& iTimePeriod) {
00584
00585     TimePeriod& lCloneTimePeriod =
00586         FacCloneBom<TimePeriod>::instance().clone (iTimePeriod);
00587
00588     // Check whether there are fare-feature objects
00589     const bool hasFareFeaturesList =
00590         BomManager::hasList<FareFeatures> (iTimePeriod);
00591     if (hasFareFeaturesList == true) {
00592         // Browse the fare-features
00593         const FareFeaturesList_T& lFareFeaturesList =
00594             BomManager::getList<FareFeatures> (iTimePeriod);
00595         for (FareFeaturesList_T::const_iterator itFF = lFareFeaturesList.begin();
00596             itFF != lFareFeaturesList.end(); ++itFF) {
00597             const FareFeatures* lFF_ptr = *itFF;
00598             assert (lFF_ptr != NULL);
00599
00600             // Clone the current fare-feature
00601             FareFeatures& lCloneFareFeatures =
00602                 cloneFeatures<FareFeatures> (*lFF_ptr);
00603             FacBomManager::addToListAndMap (lCloneTimePeriod, lCloneFareFeatures);
00604
00605             FacBomManager::linkWithParent (lCloneTimePeriod, lCloneFareFeatures);
00606         }
00607     }
00608
00609     // Check whether there are yield-feature objects
00610     const bool hasYieldFeaturesList =
00611         BomManager::hasList<YieldFeatures> (iTimePeriod);
00612     if (hasYieldFeaturesList == true) {
00613         // Browse the yield-features
00614         const YieldFeaturesList_T& lYieldFeaturesList =
00615             BomManager::getList<YieldFeatures> (iTimePeriod);
00616         for (YieldFeaturesList_T::const_iterator itYF =
00617             lYieldFeaturesList.begin();
00618             itYF != lYieldFeaturesList.end(); ++itYF) {
00619             const YieldFeatures* lYF_ptr = *itYF;
00620             assert (lYF_ptr != NULL);
00621
00622             // Clone the current yield-feature
00623             YieldFeatures& lCloneYieldFeatures =
00624                 cloneFeatures<YieldFeatures> (*lYF_ptr);
00625             FacBomManager::addToListAndMap (lCloneTimePeriod, lCloneYieldFeatures);
00626         }
00627     }
00628 }
00629
00630 };

```

```

00632         FacBomManager::linkWithParent (lCloneTimePeriod, lCloneYieldFeatures);
00633     }
00634 }
00635
00636     return lCloneTimePeriod;
00637 }
00638
00639 ///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00640 template <typename FEATURE_TYPE>
00641 FEATURE_TYPE& CmdCloneBomManager::
00642 cloneFeatures (const FEATURE_TYPE& iFeatures) {
00643
00644     FEATURE_TYPE& lCloneFeatures =
00645         FacCloneBom<FEATURE_TYPE>::instance().
00646         clone (iFeatures);
00647
00648     // Check whether there are airline-class list objects
00649     const bool hasAirlineClassList =
00650         BomManager::hasList<AirlineClassList> (iFeatures);
00651     if (hasAirlineClassList == true) {
00652         // Browse the airline-class lists
00653         const AirlineClassListList_T& lAirlineClassList =
00654             BomManager::getList<AirlineClassList> (iFeatures);
00655         for (AirlineClassListList_T::const_iterator itACList =
00656             lAirlineClassList.begin();
00657             itACList != lAirlineClassList.end(); ++itACList) {
00658             const AirlineClassList* lACList_ptr = *itACList;
00659             assert (lACList_ptr != NULL);
00660
00661             // Clone the current airline-class list
00662             AirlineClassList& lCloneAirlineClassList =
00663                 cloneAirlineClassList (*lACList_ptr);
00664             FacBomManager::addToListAndMap (lCloneFeatures,
00665                 lCloneAirlineClassList);
00666             FacBomManager::linkWithParent (lCloneFeatures,
00667                 lCloneAirlineClassList);
00668         }
00669     }
00670
00671     return lCloneFeatures;
00672 }
00673
00674 ///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00675 AirlineClassList& CmdCloneBomManager::
00676 cloneAirlineClassList (const AirlineClassList& iAirlineClassList) {
00677
00678     AirlineClassList& lCloneAirlineClassList =
00679         FacCloneBom<AirlineClassList>::instance().
00680         clone (iAirlineClassList);
00681
00682     return lCloneAirlineClassList;
00683 }
00684
00685 ///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00686 FlightPeriod& CmdCloneBomManager::
00687 cloneFlightPeriod (const FlightPeriod& iFlightPeriod) {
00688
00689     FlightPeriod& lCloneFlightPeriod =
00690         FacCloneBom<FlightPeriod>::instance().
00691         clone (iFlightPeriod);
00692
00693     // Check whether there are airline-class list objects
00694     const bool hasSegmentPeriodList =
00695         BomManager::hasList<SegmentPeriod> (iFlightPeriod);
00696     if (hasSegmentPeriodList == true) {
00697         // Browse the airline-class lists
00698         const SegmentPeriodList_T& lSegmentPeriodList =
00699             BomManager::getList<SegmentPeriod> (iFlightPeriod);
00700         for (SegmentPeriodList_T::const_iterator itSegmentPeriod =
00701             lSegmentPeriodList.begin();
00702             itSegmentPeriod != lSegmentPeriodList.end(); ++itSegmentPeriod) {
00703             const SegmentPeriod* lSegmentPeriod_ptr = *itSegmentPeriod;
00704             assert (lSegmentPeriod_ptr != NULL);
00705
00706             // Clone the current airline-class list
00707             SegmentPeriod& lCloneSegmentPeriod =
00708                 cloneSegmentPeriod (*lSegmentPeriod_ptr);
00709             FacBomManager::addToListAndMap (lCloneFlightPeriod,
00710                 lCloneSegmentPeriod);
00711             FacBomManager::linkWithParent (lCloneFlightPeriod,
00712                 lCloneSegmentPeriod);
00713         }
00714     }
00715
00716     return lCloneFlightPeriod;
00717 }
00718
00719 }
00720
00721 }
00722
00723 }
00724

```



```

00725 // //////////////////////////////////////
00726 SegmentPeriod& CmdCloneBomManager::
00727 cloneSegmentPeriod (const SegmentPeriod& iSegmentPeriod) {
00728
00732     SegmentPeriod& lCloneSegmentPeriod =
00733         FacCloneBom<SegmentPeriod>::instance().
clone (iSegmentPeriod);
00734
00735     return lCloneSegmentPeriod;
00736 }
00737
00738 }
00739

```

### 33.547 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>

```

#### Classes

- class [stdair::CmdCloneBomManager](#)

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.548 CmdCloneBomManager.hpp

```

00001 #ifndef __STDAIR_CMD_CMDCLONEBOMMANAGER_HPP
00002 #define __STDAIR_CMD_CMDCLONEBOMMANAGER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 // StdAir
00010 #include <stdair/command/CmdAbstract.hpp>

```

```

00011 // StdAir Bom
00012 #include <stdair/bom/BomRoot.hpp>
00013 #include <stdair/bom/Inventory.hpp>
00014 #include <stdair/bom/AirlineFeature.hpp>
00015 #include <stdair/bom/OnDDate.hpp>
00016 #include <stdair/bom/FlightDate.hpp>
00017 #include <stdair/bom/LegDate.hpp>
00018 #include <stdair/bom/LegCabin.hpp>
00019 #include <stdair/bom/Bucket.hpp>
00020 #include <stdair/bom/SegmentDate.hpp>
00021 #include <stdair/bom/SegmentCabin.hpp>
00022 #include <stdair/bom/FareFamily.hpp>
00023 #include <stdair/bom/BookingClass.hpp>
00024 #include <stdair/bom/AirportPair.hpp>
00025 #include <stdair/bom/PosChannel.hpp>
00026 #include <stdair/bom/DatePeriod.hpp>
00027 #include <stdair/bom/TimePeriod.hpp>
00028 #include <stdair/bom/FareFeatures.hpp>
00029 #include <stdair/bom/YieldFeatures.hpp>
00030 #include <stdair/bom/AirlineClassList.hpp>
00031 #include <stdair/bom/SegmentPeriod.hpp>
00032 #include <stdair/bom/FlightPeriod.hpp>
00033
00034 namespace stdair {
00035
00040     class CmdCloneBomManager : public CmdAbstract {
00041         //
00042         friend class STDAIR_Service;
00043     private:
00044
00051         static void cloneBomRoot (const BomRoot&, BomRoot&);
00052
00061         static Inventory& cloneInventory (const Inventory&,
00062         BomRoot&);
00070         static AirlineFeature& cloneAirlineFeature (const
00071         AirlineFeature&);
00079         static OnDDate& cloneOnDDate (const OnDDate&);
00080
00088         static FlightDate& cloneFlightDate (const FlightDate&);
00089
00097         static LegDate& cloneLegDate (const LegDate&);
00098
00106         static LegCabin& cloneLegCabin (const LegCabin&);
00107
00115         static Bucket& cloneBucket (const Bucket&);
00116
00124         static SegmentDate& cloneSegmentDate (const SegmentDate&);
00125
00133         static void linkBookingClassesWithSegment (SegmentDate&,
00134         SegmentCabin&);
00135
00143         static SegmentCabin& cloneSegmentCabin (const SegmentCabin&);
00144
00152         static FareFamily& cloneFareFamily (const FareFamily&);
00153
00161         static BookingClass& cloneBookingClass (const BookingClass&);
00162
00170         static AirportPair& cloneAirportPair (const AirportPair&);
00171
00179         static PosChannel& clonePosChannel (const PosChannel&);
00180
00188         static DatePeriod& cloneDatePeriod (const DatePeriod&);
00189
00197         static TimePeriod& cloneTimePeriod (const TimePeriod&);
00198
00206         template <typename FEATURE_TYPE>
00207         static FEATURE_TYPE& cloneFeatures (const FEATURE_TYPE&);
00208
00216         static AirlineClassList& cloneAirlineClassList (const
00217         AirlineClassList&);
00225         static FlightPeriod& cloneFlightPeriod (const FlightPeriod&);
00226
00234         static SegmentPeriod& cloneSegmentPeriod (const SegmentPeriod&);
00235
00236     };
00237 }
00238 #endif // __STDAIR_CMD_CMDCLONEBOMMANAGER_HPP

```

### 33.549 stdair/command/DBManagerForAirlines.cpp File Reference

```
#include <cassert>
```

```
#include <soci.h>
#include <mysql/soci-mysql.h>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/bom/AirlineStruct.hpp>
#include <stdair/dbadaptor/Dbairline.hpp>
#include <stdair/command/DBManagerForAirlines.hpp>
#include <stdair/service/Logger.hpp>
```

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.550 DBManagerForAirlines.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // SOCI
00007 #if defined(SOCI_HEADERS_BURIED)
00008 #include <soci/core/soci.h>
00009 #include <soci/backends/mysql/soci-mysql.h>
00010 #else // SOCI_HEADERS_BURIED
00011 #include <soci.h>
00012 #include <mysql/soci-mysql.h>
00013 #endif // SOCI_HEADERS_BURIED
00014 // StdAir
00015 #include <stdair/stdair_basic_types.hpp>
00016 #include <stdair/stdair_exceptions.hpp>
00017 #include <stdair/bom/AirlineStruct.hpp>
00018 #include <stdair/dbadaptor/Dbairline.hpp>
00019 #include <stdair/command/DBManagerForAirlines.hpp>
00020 #include <stdair/service/Logger.hpp>
00021
00022 namespace stdair {
00023
00024 // //////////////////////////////////////
00025 void DBManagerForAirlines::
00026     prepareSelectStatement (DBSession_T& ioSociSession,
00027                             DBRequestStatement_T& ioSelectStatement,
00028                             AirlineStruct& ioAirline) {
00029
00030     try {
00031
00032         // Instantiate a SQL statement (no request is performed at that stage)
00033         ioSelectStatement = (ioSociSession.prepare
00034                             << "select iata_code, name "
00035                             << "from airlines ", soci::into (ioAirline));
00036
00037         // Execute the SQL query
00038         ioSelectStatement.execute();
00039
00040     } catch (std::exception const& lException) {
00041         throw SQLDatabaseException (lException.what());
00042     }
00043
00044 // //////////////////////////////////////
00045 void DBManagerForAirlines::
00046     prepareSelectOnAirlineCodeStatement (DBSession_T& ioSociSession,
00047                                           DBRequestStatement_T& ioSelectStatement,
00048                                           const AirlineCode_T& iAirlineCode,
00049                                           AirlineStruct& ioAirline) {
00050
00051     try {
00052
00053         // Instantiate a SQL statement (no request is performed at that stage)
00054         ioSelectStatement = (ioSociSession.prepare
00055                             << "select iata_code, name "
00056                             << "from airlines "
00057                             << "where iata_code = :airline_code ",
00058                             soci::into (ioAirline), soci::use (iAirlineCode));
00059
00060     }
00061 }
```

```

00071
00072     // Execute the SQL query
00073     ioSelectStatement.execute();
00074
00075     } catch (std::exception const& lException) {
00076         throw SQLDatabaseException (lException.what());
00077     }
00078 }
00079
00080 // //////////////////////////////////////
00081 bool DBManagerForAirlines::
00082 iterateOnStatement (DBRequestStatement_T& ioStatement,
00083                     AirlineStruct& ioAirline) {
00084     bool hasStillData = false;
00085
00086     try {
00087
00088         // Retrieve the next row of Airline object
00089         hasStillData = ioStatement.fetch();
00090
00091     } catch (std::exception const& lException) {
00092         throw SQLDatabaseException (lException.what());
00093     }
00094
00095     return hasStillData;
00096 }
00097
00098 // //////////////////////////////////////
00099 void DBManagerForAirlines::updateAirlineInDB (
00100     DBSession_T& ioSociSession,
00101                                     const AirlineStruct& iAirline) {
00102     try {
00103         // Begin a transaction on the database
00104         ioSociSession.begin();
00105
00106         // Retrieve the airline code
00107         const std::string& lAirlineCode = iAirline.getAirlineCode();
00108
00109         // Retrieve the airline name
00110         const std::string& lAirlineName = iAirline.getAirlineName();
00111
00112         // Instanciate a SQL statement (no request is performed at that stage)
00113         DBRequestStatement_T lUpdateStatement =
00114             (ioSociSession.prepare
00115              << "update airlines "
00116              << "set name = :name "
00117              << "where iata_code = :iata_code",
00118              soci::use (lAirlineName), soci::use (lAirlineCode));
00119
00120         // Execute the SQL query
00121         lUpdateStatement.execute (true);
00122
00123         // Commit the transaction on the database
00124         ioSociSession.commit();
00125
00126         // Debug
00127         // STDAIR_LOG_DEBUG ("[" << lAirlineCode << "]" << iAirline);
00128
00129     } catch (std::exception const& lException) {
00130         throw SQLDatabaseException (lException.what());
00131     }
00132 }
00133
00134 // //////////////////////////////////////
00135 bool DBManagerForAirlines::retrieveAirline (
00136     DBSession_T& ioSociSession,
00137                                     const AirlineCode_T& iAirlineCode,
00138                                     AirlineStruct& ioAirline) {
00139     bool oHasRetrievedAirline = false;
00140
00141     try {
00142         // Prepare the SQL request corresponding to the select statement
00143         DBRequestStatement_T lSelectStatement (ioSociSession);
00144         prepareSelectOnAirlineCodeStatement (ioSociSession, lSelectStatement,
00145                                             iAirlineCode, ioAirline);
00146
00147         // const bool shouldDoReset = true;
00148         bool hasStillData = iterateOnStatement (lSelectStatement, ioAirline);
00149         if (hasStillData == true) {
00150             oHasRetrievedAirline = true;
00151         }
00152
00153         // Sanity check
00154         // const bool shouldNotDoReset = false;
00155         hasStillData = iterateOnStatement (lSelectStatement, ioAirline);
00156
00157         // Debug

```

```

00156         // STDAIR_LOG_DEBUG ("[" << iDocID << "]" " << ioAirline);
00157
00158     } catch (std::exception const& lException) {
00159         throw SQLDatabaseException (lException.what());
00160     }
00161
00162     return oHasRetrievedAirline;
00163 }
00164
00165 }

```

### 33.551 stdair/command/DBManagerForAirlines.hpp File Reference

```

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

```

#### Classes

- class [stdair::DBManagerForAirlines](#)

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.552 DBManagerForAirlines.hpp

```

00001 #ifndef __TVLSIM_CMD_DBMANAGERFORAIRLINES_HPP
00002 #define __TVLSIM_CMD_DBMANAGERFORAIRLINES_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_db.hpp>
00009 #include <stdair/command/CmdAbstract.hpp>
00010
00011 namespace stdair {
00012
00013     // Forward declarations
00014     struct AirlineStruct;
00015
00016     class DBManagerForAirlines : public CmdAbstract {
00017     public:
00024         static void updateAirlineInDB (DBSession_T&, const
AirlineStruct&);
00025
00032         static bool retrieveAirline (DBSession_T&, const
AirlineCode_T&,
                                AirlineStruct&);
00033
00034
00035
00036     public:
00041         static void prepareSelectStatement (DBSession_T&,
DBRequestStatement_T&,
                                AirlineStruct&);
00042
00043
00048         static bool iterateOnStatement (DBRequestStatement_T&,
AirlineStruct&);
00049
00050
00051     private:
00057         static void prepareSelectOnAirlineCodeStatement (DBSession_T&,
DBRequestStatement_T&,
                                const AirlineCode_T&,
                                AirlineStruct&);
00060
00061
00062
00063     private:
00064         // ////////////////////////////////// Constructors and Destructors //////////////////////////////////
00067         DBManagerForAirlines () {}
00068         DBManagerForAirlines (const DBManagerForAirlines&) {}

```

```

00070     ~DBManagerForAirlines () {}
00071 };
00072
00073 }
00074 #endif // __TVLSIM_CMD_DBMANAGERFORAIRLINES_HPP

```

### 33.553 stdair/dbadaptor/DbAbstract.cpp File Reference

```
#include <stdair/dbadaptor/DbAbstract.hpp>
```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.554 DbAbstract.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // StdAir
00005 #include <stdair/dbadaptor/DbAbstract.hpp>
00006
00007 namespace stdair {
00008
00009 }

```

### 33.555 stdair/dbadaptor/DbAbstract.hpp File Reference

```
#include <iosfwd>
```

#### Classes

- class [stdair::DbAbstract](#)

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

#### Functions

- `template<class charT, class traits >`  
`std::basic_ostream< charT, traits > & operator<< (std::basic_ostream< charT, traits > &ioOut, const stdair::DbAbstract &iDb)`
- `template<class charT, class traits >`  
`std::basic_istream< charT, traits > & operator>> (std::basic_istream< charT, traits > &ioIn, stdair::DbAbstract &iDb)`

#### 33.555.1 Function Documentation

**33.555.1.1** `template<class charT , class traits > std::basic_ostream<charT, traits>& operator<< ( std::basic_ostream<charT, traits > & ioOut, const stdair::DbAbstract & iDb ) [inline]`

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (p653) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 41 of file [DbAbstract.hpp](#).

**33.555.1.2** `template<class charT , class traits > std::basic_istream<charT, traits>& operator>> ( std::basic_istream<charT, traits > & ioIn, stdair::DbAbstract & ioDb ) [inline]`

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (pp655-657) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 69 of file [DbAbstract.hpp](#).

References [stdair::DbAbstract::fromStream\(\)](#).

## 33.556 DbAbstract.hpp

```

00001 #ifndef __STDAIR_DBA_DBAABSTRACT_HPP
00002 #define __STDAIR_DBA_DBAABSTRACT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009
00010 namespace stdair {
00011
00012     class DbAbstract {
00013     public:
00014
00015         virtual ~DbAbstract() {}
00016
00017         virtual void toStream (std::ostream& ioOut) const {}
00018
00019         virtual void fromStream (std::istream& ioIn) {}
00020
00021     protected:
00022         DbAbstract() {}
00023     };
00024 }
00025
00026 template <class charT, class traits>
00027 inline
00028 std::basic_ostream<charT, traits>&
00029 operator<< (std::basic_ostream<charT, traits>& ioOut,
00030             const stdair::DbAbstract & iDb) {
00031     std::basic_ostringstream<charT, traits> ostr;
00032     ostr.copyfmt (ioOut);
00033     ostr.width (0);
00034
00035     // Fill string stream
00036     iDb.toStream (ostr);
00037
00038     // Print string stream
00039     ioOut << ostr.str();
00040
00041     return ioOut;
00042 }
00043
00044 template <class charT, class traits>
00045 inline
00046 std::basic_istream<charT, traits>&
00047 operator>> (std::basic_istream<charT, traits>& ioIn,
00048             stdair::DbAbstract & ioDb) {
00049     // Fill Db object with input stream
00050     ioDb.fromStream (ioIn);
00051
00052     return ioIn;
00053 }
00054
00055 #endif // __STDAIR_DBA_DBAABSTRACT_HPP

```

**33.557 stdair/dbadaptor/Dbairline.cpp File Reference**

```
#include <exception>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/AirlineStruct.hpp>
#include <stdair/dbadaptor/Dbairline.hpp>
#include <stdair/service/Logger.hpp>
```

**Namespaces**

- [soci](#)
- [stdair](#)

*Handle on the StdAir library context.*

**33.558 Dbairline.cpp**

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <exception>
00006 #include <string>
00007 // Stdair
00008 #include <stdair/stdair_inventory_types.hpp>
00009 #include <stdair/bom/AirlineStruct.hpp>
00010 #include <stdair/dbadaptor/Dbairline.hpp>
00011 #include <stdair/service/Logger.hpp>
00012
00013 namespace soci {
00014
00015 // //////////////////////////////////////
00016 void type_conversion<stdair::AirlineStruct>::
00017 from_base (values const& iAirlineValues, indicator /* ind */,
00018           stdair::AirlineStruct& ioAirline) {
00019     /*
00020      iata_code, name
00021     */
00022     ioAirline.setAirlineCode (iAirlineValues.get<std::string> ("iata_code"));
00023     // The city code will be set to the default value (empty string)
00024     // when the column is null
00025     ioAirline.setAirlineName (iAirlineValues.get<std::string> ("name", ""));
00026 }
00027
00028 // //////////////////////////////////////
00029 void type_conversion<stdair::AirlineStruct>::
00030 to_base (const stdair::AirlineStruct& iAirline, values& ioAirlineValues,
00031         indicator& ioIndicator) {
00032     const indicator lNameIndicator =
00033         iAirline.getAirlineName().empty() ? i_null : i_ok;
00034     ioAirlineValues.set ("iata_code", iAirline.getAirlineCode());
00035     ioAirlineValues.set ("name", iAirline.getAirlineName(), lNameIndicator);
00036     ioIndicator = i_ok;
00037 }
00038
00039 }
00040
00041 namespace stdair {
00042
00043 }
```

**33.559 stdair/dbadaptor/Dbairline.hpp File Reference**

```
#include <soci/soci.h>
```

**Classes**

- struct [soci::type\\_conversion< stdair::AirlineStruct >](#)



## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

- [soci](#)

## 33.560 Dbairline.hpp

```

00001 #ifndef __STDAIR_DBA_DBAAIRLINE_HPP
00002 #define __STDAIR_DBA_DBAAIRLINE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // SOCI
00008 #if defined(SOCI_HEADERS_BURIED)
00009 #include <soci/core/soci.h>
00010 #else // SOCI_HEADERS_BURIED
00011 #include <soci/soci.h>
00012 #endif // SOCI_HEADERS_BURIED
00013
00014 // Forward declarations
00015 namespace stdair {
00016     struct AirlineStruct;
00017 }
00018
00019 namespace soci {
00020
00024     template <>
00025     struct type_conversion<stdair::AirlineStruct> {
00026
00027         typedef values base_type;
00028
00030         static void from_base (values const& iAirlineValues,
00031                               indicator /* ind */,
00032                               stdair::AirlineStruct& ioAirline);
00033
00034         static void to_base (const stdair::AirlineStruct& iAirline,
00035                             values& ioAirlineValues,
00036                             indicator& ioIndicator);
00037     };
00038 }
00039
00040 #endif // __STDAIR_DBA_DBAAIRLINE_HPP

```

## 33.561 stdair/factory/FacAbstract.cpp File Reference

```

#include <cassert>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/factory/FacAbstract.hpp>

```

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.562 FacAbstract.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // STDAIR
00007 #include <stdair/bom/BomAbstract.hpp>
00008 #include <stdair/factory/FacAbstract.hpp>
00009
00010 namespace stdair {
00011

```

```

00012 // //////////////////////////////////////
00013 FacAbstract::~FacAbstract() {
00014 }
00015
00016 }

```

### 33.563 stdair/factory/FacAbstract.hpp File Reference

#### Classes

- class [stdair::FacAbstract](#)

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.564 FacAbstract.hpp

```

00001 #ifndef __STDAIR_FAC_FACABSTRACT_HPP
00002 #define __STDAIR_FAC_FACABSTRACT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007
00008 namespace stdair {
00010     class FacAbstract {
00011     public:
00013         virtual ~FacAbstract();
00014
00015     protected:
00018         FacAbstract() {}
00019     };
00020 }
00021 #endif // __STDAIR_FAC_FACABSTRACT_HPP

```

### 33.565 stdair/factory/FacBom.hpp File Reference

```

#include <cassert>
#include <string>
#include <list>
#include <stdair/factory/FacAbstract.hpp>
#include <stdair/service/FacSupervisor.hpp>
#include <stdair/service/Logger.hpp>

```

#### Classes

- class [stdair::FacBom< BOM >](#)

*Base class for Factory layer.*

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.566 FacBom.hpp

```

00001 #ifndef __STDAIR_FAC_FACBOM_HPP
00002 #define __STDAIR_FAC_FACBOM_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <cassert>
00009 #include <string>
00010 #include <list>
00011 // StdAir
00012 #include <stdair/factory/FacAbstract.hpp>
00013 #include <stdair/service/FacSupervisor.hpp>
00014 #include <stdair/service/Logger.hpp>
00015
00016 namespace stdair {
00017
00021     template <typename BOM>
00022     class FacBom : public FacAbstract {
00023
00025         typedef std::list<BOM*> BomPool_T;
00026         typedef typename BOM::Key_T Key_T;
00027
00028     public:
00029         // ////////////////////////////////// Business methods //////////////////////////////////
00037         static FacBom& instance();
00038
00042         BOM& create ();
00043         BOM& create (const Key_T&);
00044         BOM& create (const BOM&);
00045
00046     protected:
00050         FacBom() {}
00051
00052     public:
00056         ~FacBom() {
00057             clean();
00058         }
00059
00063         void clean();
00064
00065     private:
00067         // ////////////////////////////////// Attributes //////////////////////////////////
00071         static FacBom* _instance;
00072
00076         BomPool_T _pool;
00077     };
00078
00079
00080 // //////////////////////////////////////
00081 template <typename BOM> FacBom<BOM>* FacBom<BOM>::_instance = NULL;
00082
00083 // //////////////////////////////////////
00084 template <typename BOM> FacBom<BOM>& FacBom<BOM>::instance () {
00085     if (_instance == NULL) {
00086         _instance = new FacBom ();
00087         assert (_instance != NULL);
00088
00089         FacSupervisor::instance().
00090         registerPersistentBomFactory (_instance);
00091     }
00092     return *_instance;
00093 }
00094
00095 // //////////////////////////////////////
00096 template <typename BOM> void FacBom<BOM>::clean () {
00097     // Destroy all the objects
00098     for (typename BomPool_T::iterator itBom = _pool.begin();
00099          itBom != _pool.end(); ++itBom) {
00100         BOM* currentBom_ptr = *itBom;
00101         assert (currentBom_ptr != NULL);
00102         delete currentBom_ptr; currentBom_ptr = NULL;
00103     }
00104
00105     // Empty the pool.
00106     _pool.clear();
00107
00108     // Reset the static instance.
00109     _instance = NULL;
00110 }
00111
00112 // //////////////////////////////////////
00112 template <typename BOM> BOM& FacBom<BOM>::create () {

```

```

00113     Key_T lKey;
00114     return instance().create (lKey);
00115 }
00116
00117 // //////////////////////////////////////
00118 template <typename BOM> BOM& FacBom<BOM>::create (const Key_T& iKey) {
00119     BOM* oBom_ptr = new BOM (iKey);
00120     assert (oBom_ptr != NULL);
00121     _pool.push_back (oBom_ptr);
00122     return *oBom_ptr;
00123 }
00124
00125 // //////////////////////////////////////
00126 template <typename BOM> BOM& FacBom<BOM>::create (const BOM& iBom) {
00127     BOM* oBom_ptr = new BOM (iBom);
00128     assert (oBom_ptr != NULL);
00129     _pool.push_back (oBom_ptr);
00130     return *oBom_ptr;
00131 }
00132
00133 }
00134 #endif // __STDAIR_FAC_FACBOM_HPP

```

### 33.567 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

- [stdair](#)

*Handle on the StdAir library context.*

### 33.568 FacBomManager.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // StdAir
00007 #include <stdair/basic/BasConst_General.hpp>
00008 #include <stdair/basic/BasConst_Inventory.hpp>
00009 #include <stdair/bom/BomManager.hpp>
00010 #include <stdair/bom/SegmentCabin.hpp>
00011 #include <stdair/bom/SimpleNestingStructure.hpp>
00012 #include <stdair/bom/NestingNode.hpp>
00013 #include <stdair/bom/BookingClass.hpp>
00014 #include <stdair/factory/FacBomManager.hpp>
00015 #include <stdair/service/Logger.hpp>
00016
00017 namespace stdair {
00018 // //////////////////////////////////////
00019 void FacBomManager::
00020 resetYieldBasedNestingStructure (const
SegmentCabin& iSegmentCabin) {
00021     const SimpleNestingStructure& lYieldBasedNS =
00022 BomManager::getObject<SimpleNestingStructure> (iSegmentCabin,
YIELD_BASED_NESTING_STRUCTURE_CODE);
00023
00024 // Browse the list of node and reset each one.
00025 const NestingNodeList_T& lNestingNodeList =
00026 BomManager::getList<NestingNode> (lYieldBasedNS);

```

```

00027     for (NestingNodeList_T::const_iterator itNode = lNestingNodeList.begin();
00028           itNode != lNestingNodeList.end(); ++itNode) {
00029         stdair::NestingNode* lNode_ptr = *itNode;
00030         assert (lNode_ptr != NULL);
00031
00032         lNode_ptr->setYield (-1.0);
00033
00034         // Clear the list of booking classes of the node
00035         const HolderMap_T& lHolderMap = lNode_ptr->getHolderMap();
00036         HolderMap_T::const_iterator itHolder = lHolderMap.find (&typeid (
BookingClass));
00037
00038         if (itHolder == lHolderMap.end()) {
00039             const std::string lName (typeid (BookingClass).name());
00040             throw NonInitialisedContainerException("Cannot find the holder of
type "
00041
00042
00043             + lName + " within: "
00044             + lNode_ptr->describeKey());
00045
00046         BomHolder<BookingClass>* lBomHolder_ptr =
00047             static_cast<BomHolder<BookingClass>*> (itHolder->second);
00048         assert (lBomHolder_ptr != NULL);
00049
00050         BookingClassList_T& lBCList = lBomHolder_ptr->_bomList;
00051         lBCList.clear();
00052     }
00053 }
00054 }

```

### 33.569 stdair/factory/FacBomManager.hpp File Reference

```

#include <iosfwd>
#include <list>
#include <map>
#include <boost/static_assert.hpp>
#include <boost/type_traits/is_same.hpp>
#include <stdair/bom/BomHolder.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/factory/FacAbstract.hpp>
#include <stdair/factory/FacBom.hpp>
#include <stdair/bom/SegmentDate.hpp>

```

#### Classes

- class [stdair::FacBomManager](#)  
*Utility class for linking StdAir-based objects.*

#### Namespaces

- [stdair](#)  
*Handle on the StdAir library context.*

### 33.570 FacBomManager.hpp

```

00001 #ifndef __STDAIR_FAC_FACBOMMANAGER_HPP
00002 #define __STDAIR_FAC_FACBOMMANAGER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <list>
00010 #include <map>
00011 // Boost

```

```

00012 #include <boost/static_assert.hpp>
00013 #include <boost/type_traits/is_same.hpp>
00014 // StdAir
00015 #include <stdair/bom/BomHolder.hpp>
00016 #include <stdair/bom/BomManager.hpp>
00017 #include <stdair/factory/FacAbstract.hpp>
00018 #include <stdair/factory/FacBom.hpp>
00019 // Stdair BOM Objects
00020 #include <stdair/bom/SegmentDate.hpp>
00021
00022
00023 namespace stdair {
00024     // Forward declarations.
00025     class SegmentCabin;
00026
00030     class FacBomManager : public FacAbstract {
00031     public:
00032         // ////////////////////////////////// Business methods. //////////////////////////////////
00041         template <typename OBJECT2, typename OBJECT1>
00042         static BomHolder<OBJECT2>* getBomHolderPtr (OBJECT1&);
00043
00053         template <typename OBJECT2, typename OBJECT1>
00054         static BomHolder<OBJECT2>& addBomHolder (OBJECT1&);
00055
00067         template <typename OBJECT1, typename OBJECT2>
00068         static void addToList (OBJECT1&, OBJECT2&);
00069
00082         template <typename OBJECT1, typename OBJECT2>
00083         static void addToMap (OBJECT1&, OBJECT2&, const MapKey_T&);
00084
00096         template <typename OBJECT1, typename OBJECT2>
00097         static void addToMap (OBJECT1&, OBJECT2&);
00098
00110         template <typename OBJECT1, typename OBJECT2>
00111         static void addToListAndMap (OBJECT1&, OBJECT2&);
00112
00125         template <typename OBJECT1, typename OBJECT2>
00126         static void addToListAndMap (OBJECT1&, OBJECT2&, const
MapKey_T&);
00127
00134         template <typename PARENT, typename CHILD>
00135         static void linkWithParent (PARENT&, CHILD&);
00136
00147         template <typename OBJECT2, typename OBJECT1>
00148         static void cloneHolder (OBJECT1&, const OBJECT1&);
00149
00150
00151     private:
00164         template <typename OBJECT1, typename OBJECT2>
00165         static void addToList (BomHolder<OBJECT2>&, OBJECT1&, OBJECT2&);
00166
00180         template <typename OBJECT1, typename OBJECT2>
00181         static void addToMap (BomHolder<OBJECT2>&, OBJECT1&, OBJECT2&,
const MapKey_T&);
00182
00192         template <typename OBJECT2, typename OBJECT1>
00193         static BomHolder<OBJECT2>& getBomHolder (OBJECT1&);
00194
00195     public:
00200         static void resetYieldBasedNestingStructure (const
SegmentCabin&);
00201
00205         static void setAirlineFeature (Inventory& iInventory,
AirlineFeature& iAirlineFeature) {
00206             iInventory.setAirlineFeature (iAirlineFeature);
00207         }
00208
00209
00213         static void linkWithOperating (SegmentDate& iSegmentDate,
SegmentDate& iOperatingSegmentDate) {
00214             iSegmentDate.linkWithOperating (iOperatingSegmentDate);
00215         }
00216
00217
00218
00219     protected:
00225         FacBomManager() { }
00226
00227     public:
00231         ~FacBomManager() { }
00232     };
00233
00234     // //////////////////////////////////////
00235     // Public business method.
00236     // Compile time assertion to check OBJECT1 and OBJECT2 types.
00237     template <typename OBJECT2, typename OBJECT1>
00238     BomHolder<OBJECT2>& FacBomManager::addBomHolder (OBJECT1&
ioObject1) {
00239

```

```

00240 //
00241 // Compile time assertion: this function must never be called with the
00242 // following list of couple types:
00243 // <SegmentDate, SegmentDate>
00244 // <AirlineFeature, Inventory>
00245 //
00246 BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, SegmentDate>::value == false
00247 || boost::is_same<OBJECT2, SegmentDate>::value == false));
00248 BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, Inventory>::value == false
00249 || boost::is_same<OBJECT2, AirlineFeature>::value == false));
00250
00251
00252 BomHolder<OBJECT2>* lBomHolder_ptr =
00253     &FacBom<BomHolder<OBJECT2> >::instance().create();
00254
00255 const bool hasInsertBeenSuccessful =
00256     ioObject1._holderMap.insert (typename HolderMap_T::
00257         value_type (&typeid (OBJECT2),
00258             lBomHolder_ptr)).second;
00259 assert (hasInsertBeenSuccessful == true);
00260
00261 return *lBomHolder_ptr;
00262 }
00263
00264 // //////////////////////////////////////
00265 // Public business method.
00266 // Compile time assertion to check OBJECT1 and OBJECT2 types.
00267 template <typename OBJECT2, typename OBJECT1>
00268 BomHolder<OBJECT2>* FacBomManager::getBomHolderPtr (
00269     OBJECT1& ioObject1) {
00270 //
00271 // Compile time assertion: this function must never be called with the
00272 // following list of couple types:
00273 // <SegmentDate, SegmentDate>
00274 // <AirlineFeature, Inventory>
00275 //
00276 BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, SegmentDate>::value == false
00277 || boost::is_same<OBJECT2, SegmentDate>::value == false));
00278 BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, Inventory>::value == false
00279 || boost::is_same<OBJECT2, AirlineFeature>::value == false));
00280
00281 BomHolder<OBJECT2>* lBomHolder_ptr = NULL;
00282
00283 // Find the corresponding BomHolder within the object1, if existing.
00284 HolderMap_T::const_iterator itHolder =
00285     ioObject1._holderMap.find (&typeid (OBJECT2));
00286
00287 if (itHolder != ioObject1._holderMap.end()) {
00288     lBomHolder_ptr = static_cast<BomHolder<OBJECT2>*> (itHolder->second);
00289 }
00290
00291 return lBomHolder_ptr;
00292 }
00293
00294 // //////////////////////////////////////
00295 // Private method.
00296 template <typename OBJECT2, typename OBJECT1>
00297 BomHolder<OBJECT2>& FacBomManager::getBomHolder (OBJECT1& ioObject1) {
00298 //
00299 // Compile time assertion: this function must never be called with the
00300 // following list of couple types:
00301 // <SegmentDate, SegmentDate>
00302 // <AirlineFeature, Inventory>
00303 //
00304 BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, SegmentDate>::value == false
00305 || boost::is_same<OBJECT2, SegmentDate>::value == false));
00306 BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, Inventory>::value == false
00307 || boost::is_same<OBJECT2, AirlineFeature>::value == false));
00308
00309 BomHolder<OBJECT2>* lBomHolder_ptr = NULL;
00310
00311 // Find the corresponding BomHolder within the object1. If it does
00312 // not exist, then create one.
00313 HolderMap_T::const_iterator itHolder =
00314     ioObject1._holderMap.find (&typeid (OBJECT2));
00315
00316 if (itHolder == ioObject1._holderMap.end()) {
00317     lBomHolder_ptr = &addBomHolder<OBJECT2, OBJECT1> (ioObject1);
00318 } else {
00319     lBomHolder_ptr = static_cast<BomHolder<OBJECT2>*> (itHolder->second);
00320 }
00321
00322 assert (lBomHolder_ptr != NULL);
00323
00324
00325

```

```

00326     return *lBomHolder_ptr;
00327 }
00328
00329 // //////////////////////////////////////
00330 // Private method.
00331 template <typename OBJECT1, typename OBJECT2>
00332 void FacBomManager::addToList (BomHolder<OBJECT2>& ioBomHolder,
00333                                OBJECT1& ioObject1, OBJECT2& ioObject2) {
00334
00335     //
00336     // Compile time assertion: this function must never be called with the
00337     // following list of couple types:
00338     // <SegmentDate, SegmentDate>
00339     // <AirlineFeature, Inventory>
00340     //
00341     BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, SegmentDate>::value == false
00342                           || boost::is_same<OBJECT2, SegmentDate>::value == false));
00343     BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, Inventory>::value == false
00344                           || boost::is_same<OBJECT2, AirlineFeature>::value == false));
00345
00346     ioBomHolder._bomList.push_back (&ioObject2);
00347 }
00348
00349 // //////////////////////////////////////
00350 // Public business method.
00351 // This method is specialized for the following couple types:
00352 // <SegmentDate, SegmentDate>
00353 template <typename OBJECT1, typename OBJECT2>
00354 void FacBomManager::addToList (OBJECT1& ioObject1, OBJECT2& ioObject2) {
00355
00356     //
00357     // Compile time assertion: this function must never be called with the
00358     // following list of couple types:
00359     // <SegmentDate, SegmentDate>
00360     // <AirlineFeature, Inventory>
00361     //
00362     BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, SegmentDate>::value == false
00363                           || boost::is_same<OBJECT2, SegmentDate>::value == false));
00364     BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, Inventory>::value == false
00365                           || boost::is_same<OBJECT2, AirlineFeature>::value == false));
00366
00367     BomHolder<OBJECT2>& lBomHolder = getBomHolder<OBJECT2> (ioObject1);
00368
00369     addToList<OBJECT1, OBJECT2> (lBomHolder, ioObject1, ioObject2);
00370 }
00371
00372 // //////////////////////////////////////
00373 // Private method.
00374 template <typename OBJECT1, typename OBJECT2>
00375 void FacBomManager::addToMap (BomHolder<OBJECT2>& ioBomHolder,
00376                               OBJECT1& ioObject1, OBJECT2& ioObject2,
00377                               const MapKey_T& iKey) {
00378
00379     //
00380     // Compile time assertion: this function must never be called with the
00381     // following list of couple types:
00382     // <SegmentDate, SegmentDate>
00383     // <AirlineFeature, Inventory>
00384     //
00385     BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, SegmentDate>::value == false
00386                           || boost::is_same<OBJECT2, SegmentDate>::value == false));
00387     BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, Inventory>::value == false
00388                           || boost::is_same<OBJECT2, AirlineFeature>::value == false));
00389
00390     const bool insertionSucceeded =
00391         ioBomHolder._bomMap.insert (typename std::map<const MapKey_T, OBJECT2*>::
00392                                     value_type (iKey, &ioObject2)).second;
00393
00394     if (insertionSucceeded == false) {
00395         // Build a nice message, so that the error be fully explicit
00396         std::ostringstream oStr;
00397         oStr << "The given object ('" << iKey
00398             << "') can not be added to the map of '" << ioObject1.describeKey()
00399             << "' object. That map already contains: '";
00400
00401         unsigned int idx = 0;
00402         for (typename std::map<const MapKey_T, OBJECT2*>::const_iterator iter =
00403              ioBomHolder._bomMap.begin();
00404              iter != ioBomHolder._bomMap.end(); ++iter, ++idx) {
00405             const OBJECT2* lCurrentObject_ptr = iter->second;
00406             assert (lCurrentObject_ptr != NULL);
00407
00408             if (idx != 0) {
00409                 oStr << "; ";
00410             }
00411             oStr << lCurrentObject_ptr->describeKey();
00412         }
00413     }

```



```

00413         oStr << " ";
00414
00415         STDAIR_LOG_ERROR (oStr.str());
00416         throw ObjectLinkingException (oStr.str());
00417     }
00418 }
00419
00420 // //////////////////////////////////////
00421 // Public business method.
00422 // Compile time assertion to check OBJECT1 and OBJECT2 types.
00423 template <typename OBJECT1, typename OBJECT2> void FacBomManager::
00424 addToMap (OBJECT1& ioObject1, OBJECT2& ioObject2, const MapKey_T& iKey) {
00425     //
00426     // Compile time assertion: this function must never be called with the
00427     // following list of couple types:
00428     // <SegmentDate, SegmentDate>
00429     // <AirlineFeature, Inventory>
00430     //
00431     BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, SegmentDate>::value == false
00432         || boost::is_same<OBJECT2, SegmentDate>::value == false));
00433     BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, Inventory>::value == false
00434         || boost::is_same<OBJECT2, AirlineFeature>::value == false));
00435
00436     BomHolder<OBJECT2>& lBomHolder = getBomHolder<OBJECT2> (ioObject1);
00437
00438     addToMap<OBJECT1, OBJECT2> (lBomHolder, ioObject1, ioObject2, iKey);
00439 }
00440
00441 // Public business method.
00442 // Compile time assertion to check OBJECT1 and OBJECT2 types.
00443 // //////////////////////////////////////
00444 template <typename OBJECT1, typename OBJECT2>
00445 void FacBomManager::addToMap (OBJECT1& ioObject1, OBJECT2& ioObject2) {
00446     //
00447     // Compile time assertion: this function must never be called with the
00448     // following list of couple types:
00449     // <SegmentDate, SegmentDate>
00450     // <AirlineFeature, Inventory>
00451     //
00452     BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, SegmentDate>::value == false
00453         || boost::is_same<OBJECT2, SegmentDate>::value == false));
00454     BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, Inventory>::value == false
00455         || boost::is_same<OBJECT2, AirlineFeature>::value == false));
00456
00457     const MapKey_T& lKey = ioObject2.describeKey();
00458     addToMap (ioObject1, ioObject2, lKey);
00459 }
00460
00461 // //////////////////////////////////////
00462 // Public business method.
00463 // Compile time assertion to check OBJECT1 and OBJECT2 types.
00464 template <typename OBJECT1, typename OBJECT2>
00465 void FacBomManager::addToListAndMap (OBJECT1& ioObject1, OBJECT2& ioObject2
00466     , const MapKey_T& iKey) {
00467     //
00468     // Compile time assertion: this function must never be called with the
00469     // following list of couple types:
00470     // <SegmentDate, SegmentDate>
00471     // <AirlineFeature, Inventory>
00472     //
00473     BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, SegmentDate>::value == false
00474         || boost::is_same<OBJECT2, SegmentDate>::value == false));
00475     BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, Inventory>::value == false
00476         || boost::is_same<OBJECT2, AirlineFeature>::value == false));
00477
00478     BomHolder<OBJECT2>& lBomHolder = getBomHolder<OBJECT2> (ioObject1);
00479
00480     addToList<OBJECT1, OBJECT2> (lBomHolder, ioObject1, ioObject2);
00481     addToMap<OBJECT1, OBJECT2> (lBomHolder, ioObject1, ioObject2, iKey);
00482 }
00483
00484 // //////////////////////////////////////
00485 // Public business method.
00486 // Compile time assertion to check OBJECT1 and OBJECT2 types.
00487 template <typename OBJECT1, typename OBJECT2> void FacBomManager::
00488 addToListAndMap (OBJECT1& ioObject1, OBJECT2& ioObject2) {
00489     //
00490     // Compile time assertion: this function must never be called with the
00491     // following list of couple types:
00492     // <SegmentDate, SegmentDate>
00493     // <AirlineFeature, Inventory>
00494     //
00495     BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, SegmentDate>::value == false

```

```

00499         || boost::is_same<OBJECT2, SegmentDate>::value == false));
00500     BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, Inventory>::value == false
00501         || boost::is_same<OBJECT2, AirlineFeature>::value == false));
00502
00503     const MapKey_T& lKey = ioObject2.describeKey();
00504     addToListAndMap<OBJECT1, OBJECT2> (ioObject1, ioObject2, lKey);
00505 }
00506
00507 // Public business method valid for all PARENT and CHILD types.
00508 // (No compile time assertion to check PARENT and CHILD types.)
00509 // //////////////////////////////////////
00510 template <typename PARENT, typename CHILD> void FacBomManager::
00511 linkWithParent (PARENT& ioParent, CHILD& ioChild) {
00512     ioChild._parent = &ioParent;
00513 }
00514
00515 // //////////////////////////////////////
00516 // Public business method valid for all PARENT and CHILD types.
00517 // (No compile time assertion to check PARENT and CHILD types.)
00518 template <typename OBJECT2, typename OBJECT1> void FacBomManager::
00519 cloneHolder (OBJECT1& ioDest, const OBJECT1& iOri) {
00520
00521     const BomHolder<OBJECT2>& lOriginHolder =
00522         BomManager::getBomHolder<OBJECT2> (iOri);
00523
00524     BomHolder<OBJECT2>& lDestHolder = getBomHolder<OBJECT2> (ioDest);
00525     lDestHolder._bomList = lOriginHolder._bomList;
00526     lDestHolder._bomMap = lOriginHolder._bomMap;
00527 }
00528
00529 // //////////////////////////////////////
00530 //
00531 // Specialization of the template method addToList above for the types
00532 // <SegmentDate, SegmentDate>.
00533 // Add an element to the marketing segment date list of a segment date.
00534 //
00535 // //////////////////////////////////////
00536
00537 template<>
00538 inline void FacBomManager::addToList <SegmentDate,SegmentDate>
00539 (SegmentDate& ioSegmentDate,
00540 SegmentDate& ioMarketingSegmentDate) {
00541
00542     ioSegmentDate._marketingSegmentDateList.push_back (&ioMarketingSegmentDate);
00543 }
00544
00545 // //////////////////////////////////////
00546 //
00547 // TODO:
00548 // This specialization is needed for all the objects in the current
00549 // BOM tree.
00550 // (An inventory is the parent of flight dates, a flight date is the
00551 // parent of segment dates and leg dates, ...)
00552 //
00553 // //////////////////////////////////////
00554
00555 }
00556 }
00557
00558 // //////////////////////////////////////
00559
00560 #endif // __STDAIR_FAC_FACBOMMANAGER_HPP

```

### 33.571 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>

```

#### Classes

- class [stdair::FacCloneBom< BOM >](#)  
Base class for Factory layer.

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## 33.572 FacCloneBom.hpp

```

00001 #ifndef __STDAIR_FAC_FACCLONEBOM_HPP
00002 #define __STDAIR_FAC_FACCLONEBOM_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <cassert>
00009 #include <string>
00010 #include <list>
00011 // StdAir
00012 #include <stdair/factory/FacAbstract.hpp>
00013 #include <stdair/service/FacSupervisor.hpp>
00014 #include <stdair/service/Logger.hpp>
00015
00016 namespace stdair {
00017
00021     template <typename BOM>
00022     class FacCloneBom : public FacAbstract {
00023
00025         typedef std::list<BOM*> BomPool_T;
00026         typedef typename BOM::Key_T Key_T;
00027
00028
00029     public:
00030         // ////////////////////////////////// Business methods //////////////////////////////////
00037         static FacCloneBom& instance();
00038
00042         BOM& clone (const BOM&);
00043
00044     protected:
00048         FacCloneBom() {}
00049
00050     public:
00054         ~FacCloneBom() {
00055             clean();
00056         }
00057
00061         void clean();
00062
00063     private:
00065         // ////////////////////////////////// Attributes //////////////////////////////////
00069         static FacCloneBom* _instance;
00070
00074         BomPool_T _pool;
00075     };
00076
00077
00078 // //////////////////////////////////////
00079 template <typename BOM> FacCloneBom<BOM>* FacCloneBom<BOM>::_instance = NULL;
00080
00081 // //////////////////////////////////////
00082 template <typename BOM> FacCloneBom<BOM>&
FacCloneBom<BOM>::_instance () {
00083     if (_instance == NULL) {
00084         _instance = new FacCloneBom ();
00085         assert (_instance != NULL);
00086
00087         FacSupervisor::instance().registerCloneBomFactory (
00088             _instance);
00089         return *_instance;
00090     }
00091
00092 // //////////////////////////////////////
00093 template <typename BOM> void FacCloneBom<BOM>::clean () {
00094     // Destroy all the objects
00095     for (typename BomPool_T::iterator itBom = _pool.begin();
00096          itBom != _pool.end(); ++itBom) {
00097         BOM* currentBom_ptr = *itBom;
00098         assert (currentBom_ptr != NULL);
00099         delete currentBom_ptr; currentBom_ptr = NULL;
00100     }
00101
00102     // Empty the pool.

```

```

00103     _pool.clear();
00104
00105     // Reset the static instance.
00106     _instance = NULL;
00107 }
00108
00109 // //////////////////////////////////////
00110 template <typename BOM> BOM& FacCloneBom<BOM>::clone (const BOM& iBom) {
00111     BOM* oBom_ptr = new BOM (iBom);
00112     assert (oBom_ptr != NULL);
00113     _pool.push_back (oBom_ptr);
00114     return *oBom_ptr;
00115 }
00116
00117 }
00118 #endif // __STDAIR_FAC_FACCLONEBOM_HPP

```

### 33.573 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

- [stdair](#)

*Handle on the StdAir library context.*

### 33.574 DBSessionManager.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <string>
00007 #include <sstream>
00008 // SOCI
00009 #if defined(SOCI_HEADERS_BURIED)
00010 #include <soci/core/soci.h>
00011 #include <soci/backends/mysql/soci-mysql.h>
00012 #else // SOCI_HEADERS_BURIED
00013 #include <soci.h>
00014 #include <mysql/soci-mysql.h>
00015 #endif // SOCI_HEADERS_BURIED
00016 // StdAir
00017 #include <stdair/stdair_exceptions.hpp>
00018 #include <stdair/basic/BasDBParams.hpp>
00019 #include <stdair/service/DBSessionManager.hpp>
00020 #include <stdair/service/Logger.hpp>
00021
00022 namespace stdair {
00023
00024     // //////////////////////////////////////
00025     DBSessionManager::DBSessionManager () : _dbSession (NULL) {
00026     }
00027
00028     // //////////////////////////////////////
00029     DBSessionManager::DBSessionManager (const DBSessionManager&)
00030     : _dbSession (NULL) {
00031         assert (false);
00032     }
00033
00034     // //////////////////////////////////////
00035     DBSessionManager::~DBSessionManager () {

```

```

00036     // std::cout << "In DBSessionManager destructor" << std::endl;
00037     dbFinalise();
00038 }
00039
00040 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00041 void DBSessionManager::dbInit (const BasDBParams& iDBParams) {
00042
00043     // Database parameters
00044     std::ostringstream oStr;
00045     oStr << "db=" << iDBParams.getDBName() << " user=" << iDBParams.getUser()
00046         << " password=" << iDBParams.getPassword()
00047         << " port=" << iDBParams.getPort() << " host=" << iDBParams.getHost();
00048     const std::string lDBSessionConnectionString (oStr.str());
00049
00050     // Instantiate the database session: nothing else is performed at that stage
00051     _dbSession = new DBSession_T;
00052
00053     try {
00054         // Open the connection to the database
00055         _dbSession->open (soci::mysql, lDBSessionConnectionString);
00056
00057     } catch (std::exception const& lException) {
00058         std::ostringstream oMessage;
00059         oMessage <<"Error while opening a connection to database: "
00060             << lException.what() << std::endl
00061             << "Database parameters used:"
00062             << " db=" << iDBParams.getDBName()
00063             << " user=" << iDBParams.getUser()
00064             << " port=" << iDBParams.getPort()
00065             << " host=" << iDBParams.getHost();
00066         throw SQLDatabaseConnectionImpossibleException (oMessage.str());
00067     }
00068 }
00069
00070 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00071 void DBSessionManager::dbFinalise () {
00072     delete _dbSession; _dbSession = NULL;
00073 }
00074
00075 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00076 void DBSessionManager::init (const BasDBParams& iDBParams) {
00077     DBSessionManager& lInstance = instance();
00078     lInstance.dbInit (iDBParams);
00079 }
00080
00081 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00082 DBSessionManager& DBSessionManager::instance() {
00083     static DBSessionManager _instance;
00084     return _instance;
00085 }
00086
00087 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00088 void DBSessionManager::clean() {
00089 }
00090
00091 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00092 DBSession_T& DBSessionManager::getDBSession() const {
00093     if (_dbSession == NULL) {
00094         throw NonInitialisedDBSessionManagerException ("");
00095     }
00096     assert (_dbSession != NULL);
00097     return *_dbSession;
00098 }
00099
00100 }

```

### 33.575 stdair/service/DBSessionManager.hpp File Reference

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

#### Classes

- class [stdair::DBSessionManager](#)

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.576 DBSessionManager.hpp

```

00001 #ifndef __STDAIR_SVC_DBSESSIONMANAGER_HPP
00002 #define __STDAIR_SVC_DBSESSIONMANAGER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_db.hpp>
00009
00010 namespace stdair {
00011
00012 // Forward declarations
00013 struct BasDBParams;
00014
00015 class DBSessionManager {
00016     // Friend classes
00017     friend class FacSupervisor;
00018     friend class STDAIR_Service;
00019
00020 public:
00021     static DBSessionManager& instance();
00022
00023     DBSession_T& getDBSession() const;
00024
00025 private:
00026     DBSessionManager ();
00027     DBSessionManager (const DBSessionManager&);
00028     ~DBSessionManager ();
00029
00030     void dbInit (const BasDBParams&);
00031
00032     void dbFinalise ();
00033
00034 private:
00035     static void init (const BasDBParams&);
00036
00037     static void clean();
00038
00039 private:
00040     DBSession_T* _dbSession;
00041 };
00042
00043 #endif // __STDAIR_SVC_DBSESSIONMANAGER_HPP

```

### 33.577 stdair/service/FacServiceAbstract.cpp File Reference

```

#include <cassert>
#include <stdair/service/ServiceAbstract.hpp>
#include <stdair/service/FacServiceAbstract.hpp>

```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.578 FacServiceAbstract.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // StdAir

```

```

00007 #include <stdair/service/ServiceAbstract.hpp>
00008 #include <stdair/service/FacServiceAbstract.hpp>
00009
00010 namespace stdair {
00011
00012 // //////////////////////////////////////
00013 FacServiceAbstract::~FacServiceAbstract() {
00014     clean();
00015 }
00016
00017 // //////////////////////////////////////
00018 void FacServiceAbstract::clean() {
00019     for (ServicePool_T::iterator itService = _pool.begin();
00020          itService != _pool.end(); itService++) {
00021         ServiceAbstract* currentService_ptr = *itService;
00022         assert (currentService_ptr != NULL);
00023
00024         delete (currentService_ptr); currentService_ptr = NULL;
00025     }
00026
00027     // Empty the pool of Service Factories
00028     _pool.clear();
00029 }
00030
00031 }

```

### 33.579 stdair/service/FacServiceAbstract.hpp File Reference

```
#include <vector>
```

#### Classes

- class [stdair::FacServiceAbstract](#)

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.580 FacServiceAbstract.hpp

```

00001 #ifndef __STDAIR_SVC_FACSERVICEABSTRACT_HPP
00002 #define __STDAIR_SVC_FACSERVICEABSTRACT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <vector>
00009
00010 namespace stdair {
00011
00012 // Forward declarations
00013 class ServiceAbstract;
00014
00015 class FacServiceAbstract {
00016 public:
00017
00018     typedef std::vector<ServiceAbstract*> ServicePool_T;
00019
00020     virtual ~FacServiceAbstract();
00021
00022     void clean();
00023
00024 protected:
00025     FacServiceAbstract() {}
00026
00027     ServicePool_T _pool;
00028 };
00029
00030 #endif // __STDAIR_SVC_FACSERVICEABSTRACT_HPP

```

**33.581 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**

- [stdair](#)

*Handle on the StdAir library context.*

**33.582 FacSTDAIRServiceContext.cpp**

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // StdAir
00007 #include <stdair/service/FacSupervisor.hpp>
00008 #include <stdair/service/FacSTDAIRServiceContext.hpp>
00009 #include <stdair/service/STDAIR_ServiceContext.hpp>
00010
00011 namespace stdair {
00012
00013     FacSTDAIRServiceContext* FacSTDAIRServiceContext::_instance = NULL;
00014
00015     // //////////////////////////////////////
00016     FacSTDAIRServiceContext::~FacSTDAIRServiceContext() {
00017         _instance = NULL;
00018     }
00019
00020     // //////////////////////////////////////
00021     FacSTDAIRServiceContext&
00022     FacSTDAIRServiceContext::instance() {
00023         if (_instance == NULL) {
00024             _instance = new FacSTDAIRServiceContext();
00025             assert (_instance != NULL);
00026
00027             FacSupervisor::instance().registerServiceFactory (
00028                 _instance);
00029             return *_instance;
00030         }
00031
00032         // //////////////////////////////////////
00033         STDAIR_ServiceContext& FacSTDAIRServiceContext::create
00034         () {
00035             STDAIR_ServiceContext* aServiceContext_ptr = NULL;
00036
00037             aServiceContext_ptr = new STDAIR_ServiceContext ();
00038             assert (aServiceContext_ptr != NULL);
00039
00040             // The new object is added to the Bom pool
00041             _pool.push_back (aServiceContext_ptr);
00042
00043             return *aServiceContext_ptr;
00044         }
00045     }
```

**33.583 stdair/service/FacSTDAIRServiceContext.hpp File Reference**

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



## Classes

- class [stdair::FacSTDAIRServiceContext](#)  
*Factory for [Bucket](#).*

## Namespaces

- [stdair](#)  
*Handle on the StdAir library context.*

## 33.584 FacSTDAIRServiceContext.hpp

```

00001 #ifndef __STDAIR_SVC_FACSTDAIRSERVICECONTEXT_HPP
00002 #define __STDAIR_SVC_FACSTDAIRSERVICECONTEXT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/service/FacServiceAbstract.hpp>
00009
00010 namespace stdair {
00011
00012     class STDAIR_ServiceContext;
00013
00014     class FacSTDAIRServiceContext : public
00015     FacServiceAbstract {
00016     public:
00017
00018         static FacSTDAIRServiceContext& instance();
00019
00020         ~FacSTDAIRServiceContext();
00021
00022         STDAIR_ServiceContext& create();
00023
00024     protected:
00025         FacSTDAIRServiceContext() {}
00026
00027     private:
00028         static FacSTDAIRServiceContext* _instance;
00029     };
00030 }
00031 #endif // __STDAIR_SVC_FACSTDAIRSERVICECONTEXT_HPP

```

## 33.585 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

- [stdair](#)  
*Handle on the StdAir library context.*

## 33.586 FacSupervisor.cpp

```

00001 // //////////////////////////////////////
00002 // Import section

```

```

00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // StdAir
00007 #include <stdair/factory/FacAbstract.hpp>
00008 #include <stdair/service/FacServiceAbstract.hpp>
00009 #include <stdair/service/FacSupervisor.hpp>
00010 #include <stdair/service/Logger.hpp>
00011 #include <stdair/service/DBSessionManager.hpp>
00012
00013 namespace stdair {
00014
00015     FacSupervisor* FacSupervisor::_instance = NULL;
00016
00017     // //////////////////////////////////////
00018     FacSupervisor& FacSupervisor::instance() {
00019         if (_instance == NULL) {
00020             _instance = new FacSupervisor();
00021         }
00022
00023         return *_instance;
00024     }
00025
00026     // //////////////////////////////////////
00027     FacSupervisor::~FacSupervisor() {
00028         cleanPersistentBomLayer();
00029         cleanCloneBomLayer();
00030         cleanServiceLayer();
00031     }
00032
00033     // //////////////////////////////////////
00034     void FacSupervisor::registerPersistentBomFactory (
00035         FacAbstract* ioFac_ptr) {
00036         _persistentBomPool.push_back (ioFac_ptr);
00037     }
00038
00039     // //////////////////////////////////////
00040     void FacSupervisor::registerCloneBomFactory (
00041         FacAbstract* ioFac_ptr) {
00042         _cloneBomPool.push_back (ioFac_ptr);
00043     }
00044
00045     // //////////////////////////////////////
00046     void FacSupervisor::registerServiceFactory (
00047         FacServiceAbstract* ioFac_ptr) {
00048         _svcPool.push_back (ioFac_ptr);
00049     }
00050
00051     // //////////////////////////////////////
00052     void FacSupervisor::cleanPersistentBomLayer() {
00053         for (PersistentBomFactoryPool_T::const_iterator itFactory = _persistentBomPool.begin();
00054             itFactory != _persistentBomPool.end(); itFactory++) {
00055             const FacAbstract* currentFactory_ptr = *itFactory;
00056             assert (currentFactory_ptr != NULL);
00057
00058             delete (currentFactory_ptr); currentFactory_ptr = NULL;
00059         }
00060         // Empty the pool of BOM factories
00061         _persistentBomPool.clear();
00062     }
00063
00064     // //////////////////////////////////////
00065     void FacSupervisor::cleanCloneBomLayer() {
00066         for (CloneBomFactoryPool_T::const_iterator itFactory = _cloneBomPool.begin();
00067             itFactory != _cloneBomPool.end(); itFactory++) {
00068             const FacAbstract* currentFactory_ptr = *itFactory;
00069             assert (currentFactory_ptr != NULL);
00070
00071             delete (currentFactory_ptr); currentFactory_ptr = NULL;
00072         }
00073         // Empty the pool of BOM factories
00074         _cloneBomPool.clear();
00075     }
00076
00077     // //////////////////////////////////////
00078     void FacSupervisor::cleanServiceLayer() {
00079         for (ServiceFactoryPool_T::const_iterator itFactory = _svcPool.begin();
00080             itFactory != _svcPool.end(); itFactory++) {
00081             const FacServiceAbstract* currentFactory_ptr = *itFactory;
00082             assert (currentFactory_ptr != NULL);
00083
00084             delete (currentFactory_ptr); currentFactory_ptr = NULL;
00085         }
00086         // Empty the pool of Service Factories
00087         _svcPool.clear();

```

```

00087     }
00088
00089     // //////////////////////////////////////
00090 void FacSupervisor::cleanLoggerService() {
00091     // Clean the static instance of the log service
00092     Logger::clean();
00093 }
00094
00095     // //////////////////////////////////////
00096 void FacSupervisor::cleanDBSessionManager() {
00097     // Clean the static instance of the database service
00098     DBSessionManager::clean();
00099 }
00100
00101     // //////////////////////////////////////
00102 void FacSupervisor::cleanAll() {
00103
00104     // Clean the static instance of the database session manager
00105     cleanDBSessionManager();
00106
00107     // Clean the static instance of the log service
00108     cleanLoggerService();
00109
00110     // Clean the static instance of the FacSupervisor.
00111     // This in turn will invoke the destructor (~FacSupervisor() method)
00112     // of the static instance, thus cleaning both the BOM and service layers.
00113     delete _instance; _instance = NULL;
00114 }
00115
00116 }

```

### 33.587 stdair/service/FacSupervisor.hpp File Reference

```

#include <iosfwd>
#include <list>

```

#### Classes

- class [stdair::FacSupervisor](#)

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.588 FacSupervisor.hpp

```

00001 #ifndef __STDAIR_SVC_FACSUPERVISOR_HPP
00002 #define __STDAIR_SVC_FACSUPERVISOR_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <list>
00010
00011 namespace stdair {
00012
00013     class FacAbstract;
00014     class FacServiceAbstract;
00015
00016     class FacSupervisor {
00017     public:
00018         typedef std::list<FacAbstract*> PersistentBomFactoryPool_T;
00019         typedef std::list<FacAbstract*> CloneBomFactoryPool_T;
00020         typedef std::list<FacServiceAbstract*> ServiceFactoryPool_T;
00021
00022         static FacSupervisor& instance();
00023
00024         void registerPersistentBomFactory (FacAbstract*);
00025
00026     };
00027
00028 }
00029
00030 #endif

```

```

00053     void registerCloneBomFactory (FacAbstract*);
00054
00062     void registerServiceFactory (FacServiceAbstract*);
00063
00070     void cleanPersistentBomLayer();
00071
00078     void cleanCloneBomLayer();
00079
00086     void cleanServiceLayer();
00087
00091     static void cleanLoggerService();
00092
00096     static void cleanDBSessionManager();
00097
00103     static void cleanAll();
00104
00111     ~FacSupervisor();
00112
00113
00114 protected:
00120     FacSupervisor() {}
00121     FacSupervisor (const FacSupervisor&) {}
00122
00123 private:
00127     static FacSupervisor* _instance;
00128
00132     PersistentBomFactoryPool_T _persistentBomPool;
00133
00137     CloneBomFactoryPool_T _cloneBomPool;
00138
00142     ServiceFactoryPool_T _svcPool;
00143
00144 };
00145 }
00146 #endif // __STDAIR_SVC_FACSUPERVISOR_HPP

```

### 33.589 stdair/service/Logger.cpp File Reference

```

#include <iostream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/service/Logger.hpp>

```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.590 Logger.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <iostream>
00006 // StdAir Logger
00007 #include <stdair/stdair_exceptions.hpp>
00008 #include <stdair/service/Logger.hpp>
00009
00010 namespace stdair {
00011
00012 // //////////////////////////////////////
00013 Logger::Logger()
00014 : _level (LOG::DEBUG), _logStream (&std::cout),
00015   _hasBeenInitialised (false) {
00016 }
00017
00018 // //////////////////////////////////////
00019 Logger::Logger (const Logger&)
00020 : _level (LOG::DEBUG), _logStream (&std::cout),
00021   _hasBeenInitialised (false) {
00022     assert (false);
00023 }
00024
00025 // //////////////////////////////////////

```

```

00026   Logger::~Logger() {
00027       // std::cout << "In Logger destructor" << std::endl;
00028   }
00029
00030   // //////////////////////////////////////
00031   void Logger::init (const BasLogParams& iLogParams) {
00032       //
00033       Logger& lInstance = instance();
00034       const bool hasBeenInitialised = lInstance.getStatus();
00035       if (hasBeenInitialised == true
00036           && iLogParams.getForcedInitialisationFlag() == false) {
00037           STDAIR_LOG_ERROR ("Error: the log stream has already been initialised");
00038           assert (false);
00039       }
00040
00041       lInstance.setLevel (iLogParams._logLevel);
00042       lInstance.setStream (iLogParams._logStream);
00043       lInstance.setStatus (true);
00044   }
00045
00046   // //////////////////////////////////////
00047   Logger& Logger::instance() {
00048       static Logger _instance;
00049       return _instance;
00050   }
00051
00052   // //////////////////////////////////////
00053   BasLogParams Logger::getLogParams() {
00054       std::ostream* oStream_ptr = instance()._logStream;
00055       assert (oStream_ptr != NULL);
00056       return BasLogParams (instance()._level, *oStream_ptr);
00057   }
00058
00059   // //////////////////////////////////////
00060   void Logger::clean() {
00061       Logger& lInstance = instance();
00062       lInstance.setStatus (false);
00063   }
00064
00065
00066 }

```

### 33.591 stdair/service/Logger.hpp File Reference

```

#include <cassert>
#include <sstream>
#include <string>
#include <stdair/stdair_log.hpp>
#include <stdair/basic/BasLogParams.hpp>

```

#### Classes

- class [stdair::Logger](#)

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

#### Macros

- #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)`

### 33.591.1 Macro Definition Documentation

#### 33.591.1.1 `#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](#).

#### 33.591.1.2 `#define STDAIR_LOG_CRITICAL( iToBeLogged ) STDAIR_LOG_CORE (stdair::LOG::CRITICAL, iToBeLogged)`

Definition at line 20 of file [Logger.hpp](#).

#### 33.591.1.3 `#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\(\)](#).

#### 33.591.1.4 `#define STDAIR_LOG_NOTIFICATION( iToBeLogged ) STDAIR_LOG_CORE (stdair::LOG::NOTIFICATION, iToBeLogged)`

Definition at line 26 of file [Logger.hpp](#).

#### 33.591.1.5 `#define STDAIR_LOG_WARNING( iToBeLogged ) STDAIR_LOG_CORE (stdair::LOG::WARNING, iToBeLogged)`

Definition at line 29 of file [Logger.hpp](#).

#### 33.591.1.6 `#define STDAIR_LOG_DEBUG( iToBeLogged ) STDAIR_LOG_CORE (stdair::LOG::DEBUG, iToBeLogged)`

Definition at line 32 of file [Logger.hpp](#).

Referenced by [stdair::FFDisutilityCurveHolderStruct::addCurve\(\)](#), [stdair::FRAT5CurveHolderStruct::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\(\)](#).

#### 33.591.1.7 `#define STDAIR_LOG_VERBOSE( iToBeLogged ) STDAIR_LOG_CORE (stdair::LOG::VERBOSE, iToBeLogged)`

Definition at line 35 of file [Logger.hpp](#).

## 33.592 Logger.hpp

```
00001 #ifndef __STDAIR_SVC_LOGGER_HPP
```

```

00002 #define __STDAIR_SVC_LOGGER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <cassert>
00009 #include <sstream>
00010 #include <string>
00011 // StdAir
00012 #include <stdair/stdair_log.hpp>
00013 #include <stdair/basic/BasLogParams.hpp>
00014
00015 // ////////////////////////////////// LOG MACROS //////////////////////////////////
00016 #define STDAIR_LOG_CORE(iLevel, iToBeLogged) \
00017     { std::ostringstream ostr; ostr << iToBeLogged; \
00018       stdair::Logger::instance().log (iLevel, __LINE__, __FILE__, ostr.str()); }
00019
00020 #define STDAIR_LOG_CRITICAL(iToBeLogged) \
00021     STDAIR_LOG_CORE (stdair::LOG::CRITICAL, iToBeLogged)
00022
00023 #define STDAIR_LOG_ERROR(iToBeLogged) \
00024     STDAIR_LOG_CORE (stdair::LOG::ERROR, iToBeLogged)
00025
00026 #define STDAIR_LOG_NOTIFICATION(iToBeLogged) \
00027     STDAIR_LOG_CORE (stdair::LOG::NOTIFICATION, iToBeLogged)
00028
00029 #define STDAIR_LOG_WARNING(iToBeLogged) \
00030     STDAIR_LOG_CORE (stdair::LOG::WARNING, iToBeLogged)
00031
00032 #define STDAIR_LOG_DEBUG(iToBeLogged) \
00033     STDAIR_LOG_CORE (stdair::LOG::DEBUG, iToBeLogged)
00034
00035 #define STDAIR_LOG_VERBOSE(iToBeLogged) \
00036     STDAIR_LOG_CORE (stdair::LOG::VERBOSE, iToBeLogged)
00037 // ////////////////////////////////// (END OF) LOG MACROS //////////////////////////////////
00038
00039 namespace stdair {
00040
00041     class Logger {
00042     friend class FacSupervisor;
00043     friend class STDAIR_Service;
00044
00045     public:
00046
00047         template <typename T>
00048         void log (const LOG::EN_LogLevel iLevel, const int iLineNumber,
00049                 const std::string& iFileName, const T& iToBeLogged) {
00050             assert (_logStream != NULL);
00051             if (iLevel <= _level) {
00052                 *_logStream << "[" << LOG::_logLevels[iLevel] << "]" << iFileName << ":"
00053                     << iLineNumber << ": " << iToBeLogged << std::endl;
00054             }
00055         }
00056
00057         static Logger& instance();
00058
00059     private:
00060         // ////////////////////////////////// Initialisation and finalisation //////////////////////////////////
00061         bool getStatus() const {
00062             return _hasBeenInitialised;
00063         }
00064
00065         void setLevel (const LOG::EN_LogLevel& iLevel) {
00066             _level = iLevel;
00067         }
00068
00069         void setStream (std::ostream& ioStream) {
00070             _logStream = &ioStream;
00071         }
00072
00073         void setStatus (const bool iStatus) {
00074             _hasBeenInitialised = iStatus;
00075         }
00076
00077         Logger();
00078         Logger (const Logger&);
00079         ~Logger();
00080
00081         static void init (const BasLogParams&);
00082
00083         static BasLogParams getLogParams();
00084
00085         static void clean();
00086
00087     };
00088
00089 }
00090
00091
00092
00093
00094
00095
00096
00097
00098
00099
00100
00101
00102
00103
00104
00105
00106
00107
00108
00109
00110
00111
00112
00113
00114

```

```

00135
00136 private:
00137     // ////////////////////////////////// Attributes //////////////////////////////////
00141     LOG::EN_LogLevel _level;
00142
00146     std::ostream* _logStream;
00147
00151     bool _hasBeenInitialised;
00152 };
00153
00154 }
00155 #endif // __STDAIR_SVC_LOGGER_HPP
00156

```

### 33.593 stdair/service/ServiceAbstract.cpp File Reference

```
#include <stdair/service/ServiceAbstract.hpp>
```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.594 ServiceAbstract.cpp

```

00001 // //////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////
00004 // StdAir
00005 #include <stdair/service/ServiceAbstract.hpp>
00006
00007 namespace stdair {
00008
00009 }

```

### 33.595 stdair/service/ServiceAbstract.hpp File Reference

```
#include <iosfwd>
```

#### Classes

- class [stdair::ServiceAbstract](#)

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

#### Functions

- `template<class charT, class traits >`  
`std::basic_ostream< charT, traits > & operator<< (std::basic_ostream< charT, traits > &ioOut, const`  
`stdair::ServiceAbstract &iService)`
- `template<class charT, class traits >`  
`std::basic_istream< charT, traits > & operator>> (std::basic_istream< charT, traits > &ioIn, stdair::ServiceAbstract &ioService)`



### 33.595.1 Function Documentation

**33.595.1.1** `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](#).

**33.595.1.2** `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\(\)](#).

### 33.596 ServiceAbstract.hpp

```

00001 #ifndef __STDAIR_SVC_SERVICEABSTRACT_HPP
00002 #define __STDAIR_SVC_SERVICEABSTRACT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009
00010 namespace stdair {
00011
00012     class ServiceAbstract {
00013     public:
00014         virtual ~ServiceAbstract() {}
00015
00016         virtual void toStream (std::ostream& ioOut) const {}
00017
00018         virtual void fromStream (std::istream& ioIn) {}
00019
00020         // virtual const std::string describe() const = 0;
00021
00022     protected:
00023         ServiceAbstract() {}
00024     };
00025
00026 template <class charT, class traits>
00027 inline
00028 std::basic_ostream<charT, traits>&
00029 operator<< (std::basic_ostream<charT, traits>& ioOut,
00030            const stdair::ServiceAbstract& iService) {
00031     std::basic_ostringstream<charT, traits> ostr;
00032     ostr.copyfmt (ioOut);
00033     ostr.width (0);
00034
00035     // Fill string stream
00036     iService.toStream (ostr);
00037
00038     // Print string stream
00039     ioOut << ostr.str();
00040
00041     return ioOut;
00042 }
00043
00044 template <class charT, class traits>
00045 inline
00046 std::basic_istream<charT, traits>&
00047 operator>> (std::basic_istream<charT, traits>& ioIn,
00048            stdair::ServiceAbstract& ioService) {
00049     // Fill Service object with input stream
00050     ioService.fromStream (ioIn);
00051     return ioIn;
00052 }
00053
00054 #endif // __STDAIR_SVC_SERVICEABSTRACT_HPP

```

**33.597 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**

- [bpt](#)
- [stdair](#)

*Handle on the StdAir library context.*

**33.598 STDAIR\_Service.cpp**

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 #if BOOST_VERSION >= 104100
00008 // Boost Property Tree
00009 #include <boost/property_tree/ptree.hpp>
00010 #include <boost/property_tree/json_parser.hpp>
00011 #endif // BOOST_VERSION >= 104100
00012 // StdAir
00013 #include <stdair/stdair_types.hpp>
00014 #include <stdair/stdair_json.hpp>
00015 #include <stdair/basic/BasChronometer.hpp>
00016 #include <stdair/bom/BomManager.hpp>
00017 #include <stdair/bom/BomRetriever.hpp>
00018 #include <stdair/bom/BomJSONExport.hpp>
00019 #include <stdair/bom/BomJSONImport.hpp>
00020 #include <stdair/bom/BomINIImport.hpp>
00021 #include <stdair/bom/BomDisplay.hpp>
00022 #include <stdair/bom/BomRoot.hpp>
00023 #include <stdair/bom/EventStruct.hpp>
00024 #include <stdair/bom/BookingRequestStruct.hpp>
00025 #include <stdair/bom/DatePeriod.hpp>
00026 #include <stdair/command/CmdBomManager.hpp>
00027 #include <stdair/command/CmdCloneBomManager.hpp>
00028 #include <stdair/service/FacSupervisor.hpp>
00029 #include <stdair/service/FacSTDAIRServiceContext.hpp>
00030 #include <stdair/service/Logger.hpp>
00031 #include <stdair/service/DBSessionManager.hpp>
00032 #include <stdair/STDAIR_Service.hpp>
00033
00034 #if BOOST_VERSION >= 104100

```

```

00035 namespace bpt = boost::property_tree;
00036 #else // BOOST_VERSION >= 104100
00037 namespace bpt {
00038     typedef char ptree;
00039 }
00040 #endif // BOOST_VERSION >= 104100
00041
00042 namespace stdair {
00043
00044     // //////////////////////////////////////
00045     STDAIR_Service::STDAIR_Service() : _stdairServiceContext (NULL) {
00046
00047         // Initialise the service context
00048         initServiceContext();
00049
00050         // Initialise the (remaining of the) context
00051         init();
00052     }
00053
00054     // //////////////////////////////////////
00055     STDAIR_Service::STDAIR_Service (const
STDAIR_Service& iService)
00056         : _stdairServiceContext (NULL) {
00057         assert (false);
00058     }
00059
00060     // //////////////////////////////////////
00061     STDAIR_Service::STDAIR_Service (const
BasLogParams& iLogParams)
00062         : _stdairServiceContext (NULL) {
00063
00064         // Initialise the service context
00065         initServiceContext();
00066
00067         // Set the log file
00068         logInit (iLogParams);
00069
00070         // Initialise the (remaining of the) context
00071         init();
00072     }
00073
00074     // //////////////////////////////////////
00075     STDAIR_Service::STDAIR_Service (const
BasLogParams& iLogParams,
00076                                     const BasDBParams& iDBParams)
00077         : _stdairServiceContext (NULL) {
00078
00079         // Initialise the service context
00080         initServiceContext();
00081
00082         // Set the log file
00083         logInit (iLogParams);
00084
00085         // Create a database session
00086         dbInit (iDBParams);
00087
00088         // Initialise the (remaining of the) context
00089         init();
00090     }
00091
00092     // //////////////////////////////////////
00093     STDAIR_Service::~STDAIR_Service() {
00094         // Delete/Clean all the objects from memory
00095         finalise();
00096     }
00097
00098     // //////////////////////////////////////
00099     void STDAIR_Service::initServiceContext() {
00100         // Initialise the service context
00101         STDAIR_ServiceContext& lSTDAIR_ServiceContext =
FacSTDAIRServiceContext::instance().
create();
00102
00103         // Store the stdair service context
00104         _stdairServiceContext = &lSTDAIR_ServiceContext;
00105     }
00106
00107     // //////////////////////////////////////
00108     void STDAIR_Service::logInit (const BasLogParams& iLogParams) {
00109         Logger::init (iLogParams);
00110     }
00111
00112     // //////////////////////////////////////
00113     void STDAIR_Service::dbInit (const BasDBParams& iDBParams) {
00114         DBSessionManager::init (iDBParams);
00115     }
00116
00117     // Store the database parameters into the StdAir service context

```

```

00118     assert (_stdairServiceContext != NULL);
00119     STDAIR_ServiceContext& lSTDAIR_ServiceContext = *_stdairServiceContext;
00120     lSTDAIR_ServiceContext.setDBParams (iDBParams);
00121 }
00122
00123 // //////////////////////////////////////
00124 void STDAIR_Service::init() {
00125 }
00126
00127 // //////////////////////////////////////
00128 BomRoot& STDAIR_Service::getBomRoot() const {
00129     // Retrieve the StdAir service context
00130     assert (_stdairServiceContext != NULL);
00131     const STDAIR_ServiceContext& lSTDAIR_ServiceContext =
00132         *_stdairServiceContext;
00133     // Return the clone built-in Bom root
00134     return lSTDAIR_ServiceContext.getCloneBomRoot();
00135 }
00136
00137 // //////////////////////////////////////
00138 BomRoot& STDAIR_Service::getPersistentBomRoot() const {
00139     // Retrieve the StdAir service context
00140     assert (_stdairServiceContext != NULL);
00141     const STDAIR_ServiceContext& lSTDAIR_ServiceContext =
00142         *_stdairServiceContext;
00143     // Return the persistent built-in Bom root
00144     return lSTDAIR_ServiceContext.getPersistentBomRoot();
00145 }
00146
00147 // //////////////////////////////////////
00148 BasLogParams STDAIR_Service::getLogParams() const {
00149     return Logger::getLogParams();
00150 }
00151
00152 // //////////////////////////////////////
00153 const BasDBParams& STDAIR_Service::getDBParams() const {
00154     // Retrieve the StdAir service context
00155     assert (_stdairServiceContext != NULL);
00156     const STDAIR_ServiceContext& lSTDAIR_ServiceContext =
00157         *_stdairServiceContext;
00158     return lSTDAIR_ServiceContext.getDBParams();
00159 }
00160
00161 // //////////////////////////////////////
00162 const ServiceInitialisationType& STDAIR_Service::
00163 getServiceInitialisationType() const {
00164     // Retrieve the StdAir service context
00165     assert (_stdairServiceContext != NULL);
00166     const STDAIR_ServiceContext& lSTDAIR_ServiceContext =
00167         *_stdairServiceContext;
00168     return lSTDAIR_ServiceContext.getServiceInitialisationType();
00169 }
00170
00171 // //////////////////////////////////////
00172 void STDAIR_Service::buildSampleBom() {
00173     // Retrieve the StdAir service context
00174     assert (_stdairServiceContext != NULL);
00175     const STDAIR_ServiceContext& lSTDAIR_ServiceContext =
00176         *_stdairServiceContext;
00177
00178     // Retrieve the BOM tree root
00179     BomRoot& lPersistentBomRoot = lSTDAIR_ServiceContext.getPersistentBomRoot();
00180
00181     // Delegate the building process to the dedicated command
00182     CmdBomManager::buildSampleBom (lPersistentBomRoot);
00183 }
00184
00185 // //////////////////////////////////////
00186 void STDAIR_Service::
00187 buildDummyInventory (const CabinCapacity_T& iCabinCapacity) {
00188     // Retrieve the StdAir service context
00189     assert (_stdairServiceContext != NULL);
00190     const STDAIR_ServiceContext& lSTDAIR_ServiceContext =
00191         *_stdairServiceContext;
00192
00193     // Retrieve the BOM tree root
00194     BomRoot& lPersistentBomRoot = lSTDAIR_ServiceContext.getPersistentBomRoot();
00195
00196     // Delegate the building process to the dedicated command
00197     CmdBomManager::buildDummyInventory (lPersistentBomRoot, iCabinCapacity);
00198     CmdBomManager::buildCompleteDummyInventoryForFareFamilies (lPersistentBomRoot);
00199 }
00200
00201 // //////////////////////////////////////
00202 void STDAIR_Service::
00203 buildDummyLegSegmentAccesses (BomRoot& iBomRoot) {

```

```

00205 // Retrieve the StdAir service context
00206 assert (_stdairServiceContext != NULL);
00207
00208 // Delegate the building process to the dedicated command
00209 CmdBomManager::buildDummyLegSegmentAccesses (iBomRoot);
00210
00211 }
00212
00213 // //////////////////////////////////////
00214 void STDAIR_Service::
00215 buildSampleTravelSolutionForPricing (
00216     TravelSolutionList_T& ioTravelSolutionList) {
00217     // Build a sample list of travel solution structures
00218     CmdBomManager::buildSampleTravelSolutionForPricing (ioTravelSolutionList);
00219 }
00220 // //////////////////////////////////////
00221 void STDAIR_Service::
00222 buildSampleTravelSolutions (TravelSolutionList_T&
00223     ioTravelSolutionList) {
00224     // Build a sample list of travel solution structures
00225     CmdBomManager::buildSampleTravelSolutions (ioTravelSolutionList);
00226 }
00227 // //////////////////////////////////////
00228 BookingRequestStruct STDAIR_Service::
00229 buildSampleBookingRequest (const bool isForCRS) {
00230
00231     // Build a sample booking request structure
00232     if (isForCRS == true) {
00233         return CmdBomManager::buildSampleBookingRequestForCRS();
00234     }
00235
00236     return CmdBomManager::buildSampleBookingRequest();
00237 }
00238
00239 // //////////////////////////////////////
00240 std::string STDAIR_Service::
00241 jsonExportFlightDateList (const AirlineCode_T& iAirlineCode,
00242     const FlightNumber_T& iFlightNumber) const {
00243     std::ostringstream oStr;
00244
00245     // Retrieve the StdAir service context
00246     assert (_stdairServiceContext != NULL);
00247     const STDAIR_ServiceContext& lSTDAIR_ServiceContext =
00248         *_stdairServiceContext;
00249
00250     // Retrieve the BOM tree root
00251     const BomRoot& lCloneBomRoot = lSTDAIR_ServiceContext.getCloneBomRoot();
00252
00253     BomJSONExport::jsonExportFlightDateList (oStr, lCloneBomRoot,
00254         iAirlineCode, iFlightNumber);
00255
00256     return oStr.str();
00257 }
00258
00259 // //////////////////////////////////////
00260 std::string STDAIR_Service::
00261 jsonExportFlightDateObjects (const
00262     stdair::AirlineCode_T& iAirlineCode,
00263     const stdair::FlightNumber_T& iFlightNumber,
00264     const stdair::Date_T& iDepartureDate) const {
00265     std::ostringstream oStr;
00266
00267     // Retrieve the StdAir service context
00268     assert (_stdairServiceContext != NULL);
00269     const STDAIR_ServiceContext& lSTDAIR_ServiceContext =
00270         *_stdairServiceContext;
00271
00272     // Retrieve the BOM tree root
00273     const BomRoot& lCloneBomRoot = lSTDAIR_ServiceContext.getCloneBomRoot();
00274
00275     // Retrieve the flight-date object corresponding to the key
00276     FlightDate* lFlightDate_ptr =
00277         BomRetriever::retrieveFlightDateFromKeySet (lCloneBomRoot,
00278             iAirlineCode,
00279             iFlightNumber,
00280             iDepartureDate);
00281
00282     // Dump the content of the whole BOM tree into the string
00283     if (lFlightDate_ptr != NULL) {
00284         BomJSONExport::jsonExportFlightDateObjects (oStr, *
00285             lFlightDate_ptr);
00286     } else {
00287         #if BOOST_VERSION >= 104100

```

```

00287     //
00288     bpt::ptree lPropertyTree;
00289
00290     // Build the appropriate message, so that the client may know that
00291     // no flight-date can be found for that given key.
00292     std::ostringstream oNoFlightDateStream;
00293     oNoFlightDateStream << "No flight-date found for the given key: '"
00294         << iAirlineCode << iFlightNumber
00295         << " - " << iDepartureDate << "'";
00296     const std::string oNoFlightDateString (oNoFlightDateStream.str());
00297
00298     // Put in the property tree the fact that no flight-date has been found.
00299     // \note That is not (necessary) an error.
00300     lPropertyTree.put ("error", oNoFlightDateString.c_str());
00301
00302     // Write the property tree into the JSON stream.
00303     write_json (oStr, lPropertyTree);
00304 #endif // BOOST_VERSION >= 104100
00305 }
00306
00307     return oStr.str();
00308 }
00309
00310 // //////////////////////////////////////
00311 std::string STDAIR_Service::
00312 jsonExportEventObject (const EventStruct& iEventStruct) const {
00313
00314     std::ostringstream oStr;
00315
00316     const EventType::EN_EventType& lEventType =
00317         iEventStruct.getEventType();
00318
00319     switch (lEventType) {
00320     case EventType::BKG_REQ: {
00321         BomJSONExport::jsonExportBookingRequestObject (oStr,
00322 iEventStruct);
00323         break;
00324     }
00325     case EventType::CX:
00326     case EventType::OPT_NOT_4_FD:
00327     case EventType::OPT_NOT_4_NET:
00328     case EventType::SKD_CHG:
00329     case EventType::SNAPSHOT:
00330     case EventType::RM:
00331         break;
00332     case EventType::BRK_PT:
00333         BomJSONExport::jsonExportBreakPointObject (oStr,
00334 iEventStruct);
00335         break;
00336     default:
00337         break;
00338     }
00339     return oStr.str();
00340 }
00341 // //////////////////////////////////////
00342 bool STDAIR_Service::
00343 jsonImportConfiguration (const JSONString& iJSONString) const {
00344
00345     // Retrieve the StdAir service context
00346     assert (_stdairServiceContext != NULL);
00347     const STDAIR_ServiceContext& lSTDAIR_ServiceContext =
00348         *_stdairServiceContext;
00349
00350     // Retrieve the config holder
00351     ConfigHolderStruct& lConfigHolder =
00352         lSTDAIR_ServiceContext.getConfigHolder();
00353
00354     // Import the JSON string in the configuration holder
00355     return BomJSONImport::jsonImportConfig (iJSONString, lConfigHolder);
00356 }
00357 // //////////////////////////////////////
00358 std::string STDAIR_Service::
00359 jsonExportConfiguration () const {
00360
00361     // Retrieve the StdAir service context
00362     assert (_stdairServiceContext != NULL);
00363     const STDAIR_ServiceContext& lSTDAIR_ServiceContext =
00364         *_stdairServiceContext;
00365
00366     // Retrieve the config holder
00367     ConfigHolderStruct& lConfigHolder =
00368         lSTDAIR_ServiceContext.getConfigHolder();
00369
00370     // Export the configuration tree in a JSon format
00371     return lConfigHolder.jsonExport();

```

```

00372     }
00373
00374     // //////////////////////////////////////
00375     void STDair_Service::importINIConfig (const
ConfigINIFile& iConfigINIFile) {
00376
00377         // Retrieve the StdAir service context
00378         assert (_stdairServiceContext != NULL);
00379         const STDair_ServiceContext& lSTDair_ServiceContext =
00380             *_stdairServiceContext;
00381
00382         // Retrieve the config holder
00383         ConfigHolderStruct& lConfigHolder =
00384             lSTDair_ServiceContext.getConfigHolder();
00385
00386         // Try to import the configuration
00387         stdair::BomINIImport::importINIConfig (lConfigHolder,
iConfigINIFile);
00388     }
00389
00390     // //////////////////////////////////////
00391     void STDair_Service::importConfigValue (const std::string& iValue,
00392                                             const std::string& iPath) {
00393
00394         // Retrieve the StdAir service context
00395         assert (_stdairServiceContext != NULL);
00396         const STDair_ServiceContext& lSTDair_ServiceContext =
00397             *_stdairServiceContext;
00398
00399         // Retrieve the config holder
00400         ConfigHolderStruct& lConfigHolder =
00401             lSTDair_ServiceContext.getConfigHolder();
00402
00403         // Add the given value to the configuration
00404         lConfigHolder.addValue (iValue, iPath);
00405     }
00406
00407     // //////////////////////////////////////
00408     void STDair_Service::updateAirlineFeatures () {
00409
00410         // Retrieve the StdAir service context
00411         assert (_stdairServiceContext != NULL);
00412         const STDair_ServiceContext& lSTDair_ServiceContext =
00413             *_stdairServiceContext;
00414
00415         // Retrieve the config holder
00416         ConfigHolderStruct& lConfigHolder =
00417             lSTDair_ServiceContext.getConfigHolder();
00418
00419         // Retrieve the persistent BOM tree root
00420         BomRoot& lPersistentBomRoot =
00421             lSTDair_ServiceContext.getPersistentBomRoot();
00422
00423         // Add the given value to the configuration
00424         lConfigHolder.updateAirlineFeatures (lPersistentBomRoot);
00425     }
00426
00427     // //////////////////////////////////////
00428     std::string STDair_Service::list (const AirlineCode_T& iAirlineCode,
00429                                     const FlightNumber_T& iFlightNumber) const {
00430         std::ostringstream oStr;
00431
00432         // Retrieve the StdAir service context
00433         assert (_stdairServiceContext != NULL);
00434         const STDair_ServiceContext& lSTDair_ServiceContext = *_stdairServiceContext;
00435
00436         // Retrieve the BOM tree root
00437         const BomRoot& lCloneBomRoot = lSTDair_ServiceContext.getCloneBomRoot();
00438
00439         // Dump the content of the whole BOM tree into the string
00440         BomDisplay::list (oStr, lCloneBomRoot, iAirlineCode, iFlightNumber);
00441
00442         return oStr.str();
00443     }
00444
00445     // //////////////////////////////////////
00446     std::string STDair_Service::listAirportPairDateRange () const {
00447         std::ostringstream oStr;
00448
00449         // Retrieve the StdAir service context
00450         assert (_stdairServiceContext != NULL);
00451         const STDair_ServiceContext& lSTDair_ServiceContext = *_stdairServiceContext;
00452
00453         // Retrieve the BOM tree root
00454         const BomRoot& lCloneBomRoot = lSTDair_ServiceContext.getCloneBomRoot();
00455
00456         // Dump the content of the whole BOM tree into the string

```

```

00457     BomDisplay::listAirportPairDateRange (oStr, lCloneBomRoot);
00458
00459     return oStr.str();
00460 }
00461
00462 // //////////////////////////////////////
00463 bool STDAIR_Service::check (const AirlineCode_T& iAirlineCode,
00464                             const FlightNumber_T& iFlightNumber,
00465                             const stdair::Date_T& iDepartureDate) const {
00466     std::ostringstream oStr;
00467
00468     // Retrieve the StdAir service context
00469     assert (_stdairServiceContext != NULL);
00470     const STDAIR_ServiceContext& lSTDAIR_ServiceContext = *_stdairServiceContext;
00471
00472     // Retrieve the BOM tree root
00473     const BomRoot& lCloneBomRoot = lSTDAIR_ServiceContext.getCloneBomRoot();
00474
00475     // Dump the content of the whole BOM tree into the string
00476     const FlightDate* lFlightDate_ptr =
00477         BomRetriever::retrieveFlightDateFromKeySet (lCloneBomRoot,
00478             iAirlineCode,
00479             iFlightNumber,
00480             iDepartureDate);
00481     return (lFlightDate_ptr != NULL);
00482 }
00483
00484 // //////////////////////////////////////
00485 bool STDAIR_Service::check (const stdair::AirportCode_T&
00486                             ioOrigin,
00487                             const stdair::AirportCode_T& ioDestination,
00488                             const stdair::Date_T& ioDepartureDate) const {
00489     std::ostringstream oStr;
00490
00491     // Retrieve the StdAir service context
00492     assert (_stdairServiceContext != NULL);
00493     const STDAIR_ServiceContext& lSTDAIR_ServiceContext = *_stdairServiceContext;
00494
00495     // Retrieve the BOM tree root
00496     const BomRoot& lCloneBomRoot = lSTDAIR_ServiceContext.getCloneBomRoot();
00497
00498     // Dump the content of the whole BOM tree into the string
00499     stdair::DatePeriodList_T lDatePeriodList;
00500     BomRetriever::retrieveDatePeriodListFromKeySet (
00501         lCloneBomRoot, ioOrigin,
00502         ioDestination,
00503         ioDepartureDate,
00504         lDatePeriodList);
00505     return (lDatePeriodList.size() != 0);
00506 }
00507
00508 // //////////////////////////////////////
00509 std::string STDAIR_Service::configDisplay () const {
00510     // Retrieve the StdAir service context
00511     assert (_stdairServiceContext != NULL);
00512     const STDAIR_ServiceContext& lSTDAIR_ServiceContext =
00513         *_stdairServiceContext;
00514
00515     // Retrieve the config holder
00516     ConfigHolderStruct& lConfigHolder =
00517         lSTDAIR_ServiceContext.getConfigHolder();
00518
00519     // Display (dump in the returned string) the configuration.
00520     return lConfigHolder.describe();
00521 }
00522
00523 // //////////////////////////////////////
00524 std::string STDAIR_Service::csvDisplay () const {
00525     // Retrieve the StdAir service context
00526     assert (_stdairServiceContext != NULL);
00527     const STDAIR_ServiceContext& lSTDAIR_ServiceContext =
00528         *_stdairServiceContext;
00529
00530     // Retrieve the persistent BOM tree root
00531     const BomRoot& lPersistentBomRoot =
00532         lSTDAIR_ServiceContext.getPersistentBomRoot();
00533
00534     // Call the dedicated service
00535     return csvDisplay (lPersistentBomRoot);
00536 }
00537
00538 // //////////////////////////////////////
00539
00540

```



```

00541     std::string STDAIR_Service::csvDisplay (const
BomRoot& iBomRoot) const {
00542         std::ostringstream oStr;
00543
00544         // Retrieve the StdAir service context
00545         assert (_stdairServiceContext != NULL);
00546
00547         // Dump the content of the whole BOM tree into the string
00548         BomDisplay::csvDisplay (oStr, iBomRoot);
00549
00550         return oStr.str();
00551     }
00552
00553     ///////////////////////////////////////////////////////////////////
00554     std::string STDAIR_Service::
00555     csvDisplay (const stdair::AirlineCode_T& iAirlineCode,
00556                 const stdair::FlightNumber_T& iFlightNumber,
00557                 const stdair::Date_T& iDepartureDate) const {
00558         std::ostringstream oStr;
00559
00560         // Retrieve the StdAir service context
00561         assert (_stdairServiceContext != NULL);
00562         const STDAIR_ServiceContext& lSTDAIR_ServiceContext = *_stdairServiceContext;
00563
00564         // Retrieve the BOM tree root
00565         const BomRoot& lCloneBomRoot = lSTDAIR_ServiceContext.getCloneBomRoot();
00566
00567         // Retrieve the flight-date object corresponding to the key
00568         FlightDate* lFlightDate_ptr =
00569             BomRetriever::retrieveFlightDateFromKeySet (lCloneBomRoot,
iAirlineCode,
00570                                                         iFlightNumber,
00571                                                         iDepartureDate);
00572
00573         // Dump the content of the whole BOM tree into the string
00574         if (lFlightDate_ptr != NULL) {
00575             BomDisplay::csvDisplay (oStr, *lFlightDate_ptr);
00576         } else {
00577             oStr << "    No flight-date found for the given key: '"
00578                 << iAirlineCode << iFlightNumber << " - " << iDepartureDate << "'";
00579         }
00580
00581         return oStr.str();
00582     }
00583
00584     ///////////////////////////////////////////////////////////////////
00585     std::string STDAIR_Service::
00586     csvDisplay (const TravelSolutionList_T& iTravelSolutionList) const {
00587
00588         // Dump the content of the whole list of travel solutions into the string
00589         std::ostringstream oStr;
00590         BomDisplay::csvDisplay (oStr, iTravelSolutionList);
00591
00592         return oStr.str();
00593     }
00594
00595     ///////////////////////////////////////////////////////////////////
00596     std::string STDAIR_Service::
00597     csvDisplay (const stdair::AirportCode_T& iOrigin,
00598                 const stdair::AirportCode_T& iDestination,
00599                 const stdair::Date_T& iDepartureDate) const {
00600         std::ostringstream oStr;
00601
00602         // Retrieve the StdAir service context
00603         assert (_stdairServiceContext != NULL);
00604         const STDAIR_ServiceContext& lSTDAIR_ServiceContext = *_stdairServiceContext;
00605
00606         // Retrieve the BOM tree root
00607         const BomRoot& lCloneBomRoot = lSTDAIR_ServiceContext.getCloneBomRoot();
00608
00609         // Retrieve the flight-date object corresponding to the key
00610         DatePeriodList_T lDatePeriodList;
00611         BomRetriever::retrieveDatePeriodListFromKeySet (
00612             lCloneBomRoot, iOrigin,
00613                                                         iDestination,
00614                                                         iDepartureDate,
00615                                                         lDatePeriodList);
00616
00617         // Dump the content of the whole BOM tree into the string
00618         if (lDatePeriodList.empty()) {
00619             oStr << "    No fare-rule found for the given key: '"
00620                 << iOrigin << "-" << iDestination << " - " << iDepartureDate << "'";
00621         } else {
00622             BomDisplay::csvDisplay (oStr, lDatePeriodList);
00623         }
00624

```

```

00625     return oStr.str();
00626 }
00627
00628 // //////////////////////////////////////
00629 void STDAIR_Service::finalise() {
00630     // Clean all the objects
00631     FacSupervisor::cleanAll();
00632 }
00633
00634 // //////////////////////////////////////
00635 void STDAIR_Service::clonePersistentBom () {
00636
00637     // Retrieve the StdAir service context
00638     assert (_stdairServiceContext != NULL);
00639     STDAIR_ServiceContext& lSTDAIR_ServiceContext = *_stdairServiceContext;
00640
00641     // Clean all the cloned objects
00642     FacSupervisor::instance().cleanCloneBomLayer();
00643
00644     // Init the root of the clone BOM tree
00645     lSTDAIR_ServiceContext.initCloneBomRoot();
00646
00647     // Retrieve the persistent BOM tree root and the clone BOM tree root
00648     const BomRoot& lPersistentBomRoot =
00649         lSTDAIR_ServiceContext.getPersistentBomRoot();
00650     BomRoot& lCloneBomRoot = lSTDAIR_ServiceContext.getCloneBomRoot();
00651
00652     // Call the dedicated service to clone the whole BOM
00653     CmdCloneBomManager::cloneBomRoot (lPersistentBomRoot, lCloneBomRoot);
00654
00655 }
00656
00657 }

```

### 33.599 stdair/service/STDAIR\_ServiceContext.cpp File Reference

#### 33.600 STDAIR\_ServiceContext.cpp

```

00001
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <cassert>
00010 #include <sstream>
00011 // Boost
00012 #if BOOST_VERSION >= 103900
00013 #include <boost/make_shared.hpp>
00014 #else // BOOST_VERSION >= 103900
00015 #include <boost/shared_ptr.hpp>
00016 #endif // BOOST_VERSION >= 103900
00017 // StdAir
00018 #include <stdair/basic/BasConst_General.hpp>
00019 #include <stdair/bom/BomRoot.hpp>
00020 #include <stdair/factory/FacBom.hpp>
00021 #include <stdair/factory/FacCloneBom.hpp>
00022 #include <stdair/service/STDAIR_ServiceContext.hpp>
00023
00024 namespace stdair {
00025
00026 // //////////////////////////////////////
00027 STDAIR_ServiceContext::STDAIR_ServiceContext ()
00028 : _cloneBomRoot (NULL),
00029   _persistentBomRoot (NULL),
00030   _initType (ServiceInitialisationType::NOT_YET_INITIALISED) {
00031     // Build the BomRoot object
00032     init();
00033 }
00034
00035 // //////////////////////////////////////
00036 STDAIR_ServiceContext::
00037 STDAIR_ServiceContext (const STDAIR_ServiceContext& iServiceContext)
00038 : _cloneBomRoot (iServiceContext._cloneBomRoot),
00039   _persistentBomRoot (iServiceContext._persistentBomRoot),
00040   _initType (ServiceInitialisationType::NOT_YET_INITIALISED) {
00041     assert (false);
00042 }
00043
00044 // //////////////////////////////////////
00045 STDAIR_ServiceContext::~STDAIR_ServiceContext () {
00046 }
00047
00048 // //////////////////////////////////////

```

```

00049 void STDAIR_ServiceContext::init() {
00050     //
00051     initBomRoot();
00052     initConfigHolder();
00053 }
00054
00055 // //////////////////////////////////////
00056 void STDAIR_ServiceContext::initBomRoot() {
00057     _persistentBomRoot = &FacBom<BomRoot>::instance().
create();
00058     initCloneBomRoot();
00059 }
00060
00061 // //////////////////////////////////////
00062 void STDAIR_ServiceContext::initCloneBomRoot() {
00063     _cloneBomRoot =
00064         &FacCloneBom<BomRoot>::instance().clone(*_persistentBomRoot);
00065 }
00066
00067 // //////////////////////////////////////
00068 void STDAIR_ServiceContext::initConfigHolder() {
00069     _configHolderPtr = boost::make_shared<ConfigHolderStruct> ();
00070 }
00071
00072 // //////////////////////////////////////
00073 const std::string STDAIR_ServiceContext::shortDisplay() const {
00074     std::ostringstream oStr;
00075     oStr << "STDAIR_ServiceContext -- " << _initType
00076         << " -- DB: " << _dbParams;
00077     return oStr.str();
00078 }
00079
00080 // //////////////////////////////////////
00081 const std::string STDAIR_ServiceContext::display() const {
00082     std::ostringstream oStr;
00083     oStr << shortDisplay();
00084     return oStr.str();
00085 }
00086
00087 // //////////////////////////////////////
00088 const std::string STDAIR_ServiceContext::describe() const {
00089     return shortDisplay();
00090 }
00091
00092 // //////////////////////////////////////
00093 BomRoot& STDAIR_ServiceContext::getPersistentBomRoot() const {
00094     assert (_persistentBomRoot != NULL);
00095     return *_persistentBomRoot;
00096 }
00097
00098 // //////////////////////////////////////
00099 BomRoot& STDAIR_ServiceContext::getCloneBomRoot() const {
00100     assert (_cloneBomRoot != NULL);
00101     return *_cloneBomRoot;
00102 }
00103
00104 // //////////////////////////////////////
00105 ConfigHolderStruct& STDAIR_ServiceContext::getConfigHolder() const {
00106     assert (_configHolderPtr != NULL);
00107     return *_configHolderPtr;
00108 }
00109 }
00110

```

### 33.601 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>

```

## Classes

- class `stdair::STDAIR_ServiceContext`  
Class holding the context of the Stdair services.

## Namespaces

- `stdair`  
Handle on the StdAir library context.

## 33.602 STDAIR\_ServiceContext.hpp

```

00001 #ifndef __STDAIR_SVC_STDAIRSERVICECONTEXT_HPP
00002 #define __STDAIR_SVC_STDAIRSERVICECONTEXT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/stdair_basic_types.hpp>
00011 #include <stdair/basic/BasLogParams.hpp>
00012 #include <stdair/basic/BasDBParams.hpp>
00013 #include <stdair/bom/ConfigHolderStruct.hpp>
00014 #include <stdair/basic/ServiceInitialisationType.hpp>
00015 #include <stdair/service/ServiceAbstract.hpp>
00016
00017 namespace stdair {
00018
00020     class BomRoot;
00021
00025     class STDAIR_ServiceContext : public ServiceAbstract {
00029         friend class STDAIR_Service;
00030         friend class FacSTDAIRServiceContext;
00031
00032     private:
00033         // ////////// Getters //////////
00037         BomRoot& getPersistentBomRoot() const;
00038
00042         BomRoot& getCloneBomRoot() const;
00043
00047         ConfigHolderStruct& getConfigHolder() const;
00048
00052         const BasDBParams& getDBParams() const {
00053             return _dbParams;
00054         }
00055
00059         const ServiceInitialisationType& getServiceInitialisationType() const {
00060             return _initType;
00061         }
00062
00063     private:
00065         // ////////// Setters //////////
00069         void setDBParams (const BasDBParams& iDBParams) {
00070             _dbParams = iDBParams;
00071         }
00072
00076         void setServiceInitialisationType (const ServiceInitialisationType& iSIT) {
00077             _initType = iSIT;
00078         }
00079
00080     private:
00082         // ////////// Display Methods //////////
00086         const std::string shortDisplay() const;
00087
00091         const std::string display() const;
00092
00096         const std::string describe() const;
00097
00098     private:
00099         // ////////// Construction / initialisation //////////
00100         STDAIR_ServiceContext();
00104
00105         STDAIR_ServiceContext (const STDAIR_ServiceContext&);
00112
00113

```

```

00117     ~STDAIR_ServiceContext();
00118
00126     void init();
00127
00134     void initBomRoot();
00135
00142     void initCloneBomRoot();
00143
00149     void initConfigHolder();
00150
00151     private:
00152         // ////////// Children //////////
00156         BomRoot* _cloneBomRoot;
00157
00161         BomRoot* _persistentBomRoot;
00162
00166         ConfigHolderPtr_T _configHolderPtr;
00167
00171         BasDBParams _dbParams;
00172
00186         ServiceInitialisationType _initType;
00187     };
00188
00189 }
00190 #endif // __STDAIR_SVC_STDAIRSERVICECONTEXT_HPP

```

### 33.603 stdair/stdair\_basic\_types.hpp File Reference

```

#include <string>
#include <list>

```

#### Namespaces

- [stdair](#)  
*Handle on the StdAir library context.*

#### Typedefs

- typedef std::string [stdair::LocationCode\\_T](#)
- typedef unsigned long int [stdair::Distance\\_T](#)
- typedef LocationCode\_T [stdair::AirportCode\\_T](#)
- typedef LocationCode\_T [stdair::CityCode\\_T](#)
- typedef std::string [stdair::KeyDescription\\_T](#)
- typedef std::string [stdair::AirlineCode\\_T](#)
- typedef unsigned short [stdair::FlightNumber\\_T](#)
- typedef unsigned short [stdair::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](#)

### 33.604 stdair\_basic\_types.hpp

```

00001 #ifndef __STDAIR_STDAIR_BASIC_TYPES_HPP
00002 #define __STDAIR_STDAIR_BASIC_TYPES_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <list>
00010
00011 namespace stdair {
00012
00013 // //////////////////////////////////////
00014 // Basic types
00016 typedef std::string LocationCode_T;
00017
00019 typedef unsigned long int Distance_T;
00020
00022 typedef LocationCode_T AirportCode_T;
00023
00025 typedef LocationCode_T CityCode_T;
00026
00028 typedef std::string KeyDescription_T;
00029
00031 typedef std::string AirlineCode_T;
00032
00034 typedef unsigned short FlightNumber_T;
00035
00037 typedef unsigned short TableID_T;
00038
00041 typedef std::string CabinCode_T;
00042
00044 typedef std::string FamilyCode_T;
00045
00047 typedef std::string PolicyCode_T;
00048
00050 typedef std::string NestingStructureCode_T;
00051
00053 typedef std::string NestingNodeCode_T;
00054
00057 typedef std::string ClassCode_T;
00058
00060 typedef unsigned long Identity_T;
00061
00064 typedef std::string TripType_T;
00065
00067 typedef double MonetaryValue_T;
00068
00070 typedef double RealNumber_T;
00071

```

```

00073     typedef double Percentage_T;
00074
00076     typedef double PriceValue_T;
00077
00079     typedef double YieldValue_T;
00080
00082     typedef std::string PriceCurrency_T;
00083
00085     typedef double Revenue_T;
00086
00088     typedef double Multiplier_T;
00089
00092     typedef double NbOfSeats_T;
00093
00095     typedef unsigned int Count_T;
00096
00098     typedef short PartySize_T;
00099
00101     typedef double NbOfRequests_T;
00102
00104     typedef NbOfRequests_T NbOfBookings_T;
00105
00107     typedef NbOfRequests_T NbOfCancellations_T;
00108
00111     typedef unsigned short NbOfTravelSolutions_T;
00112
00114     typedef std::string ClassList_String_T;
00115
00117     typedef unsigned short NbOfSegments_T;
00118
00120     typedef unsigned short NbOfAirlines_T;
00121
00123     typedef double Availability_T;
00124
00126     typedef double Fare_T;
00127
00129     typedef bool Flag_T;
00130
00132     typedef unsigned int UnsignedIndex_T;
00133
00135     typedef unsigned int NbOfClasses_T;
00136
00138     typedef unsigned int NbOfFareFamilies_T;
00139
00140     // ////////// Technical //////////
00144     typedef std::string Filename_T;
00145
00148     typedef std::string FileAddress_T;
00149
00152     typedef float ProgressPercentage_T;
00153
00154 }
00155 #endif // __STDAIR_STDAIR_BASIC_TYPES_HPP

```

### 33.605 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

- [stdair](#)

*Handle on the StdAir library context.*

#### Typedefs

- typedef boost::posix\_time::time\_duration [stdair::Duration\\_T](#)
- typedef boost::gregorian::date [stdair::Date\\_T](#)
- typedef boost::posix\_time::time\_duration [stdair::Time\\_T](#)
- typedef boost::posix\_time::ptime [stdair::DateTime\\_T](#)
- typedef boost::gregorian::date\_period [stdair::DatePeriod\\_T](#)

- typedef std::string [stdair::DOW\\_String\\_T](#)
- typedef boost::gregorian::date\_duration [stdair::DateOffset\\_T](#)
- typedef 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](#)

### 33.606 stdair\_date\_time\_types.hpp

```

00001 #ifndef __STDAIR_STDAIR_DATE_TIME_TYPES_HPP
00002 #define __STDAIR_STDAIR_DATE_TIME_TYPES_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // Boost (Extended STL)
00010 #include <boost/date_time/gregorian/gregorian.hpp>
00011 #include <boost/date_time/posix_time/posix_time.hpp>
00012
00013 namespace stdair {
00014
00015     // ////////// Type definitions //////////
00017     typedef boost::posix_time::time_duration Duration_T;
00018
00020     typedef boost::gregorian::date Date_T;
00021
00023     typedef boost::posix_time::time_duration Time_T;
00024
00026     typedef boost::posix_time::ptime DateTime_T;
00027
00029     typedef boost::gregorian::date_period DatePeriod_T;
00030
00032     typedef std::string DOW_String_T;
00033
00035     typedef boost::gregorian::date_duration DateOffset_T;
00036
00038     typedef int DayDuration_T;
00039
00041     typedef bool SaturdayStay_T;
00042
00044     typedef long int IntDuration_T;
00045
00047     typedef long long int LongDuration_T;
00048
00050     typedef float FloatDuration_T;
00051
00052 }
00053 #endif // __STDAIR_STDAIR_DATE_TIME_TYPES_HPP

```

### 33.607 stdair/stdair\_db.hpp File Reference

```
#include <string>
```

#### Namespaces

- [soci](#)
- [stdair](#)

*Handle on the StdAir library context.*

#### Typedefs

- typedef soci::session [stdair::DBSession\\_T](#)
- typedef soci::statement [stdair::DBRequestStatement\\_T](#)
- typedef std::string [stdair::DBConnectionName\\_T](#)



### 33.608 stdair\_db.hpp

```

00001 #ifndef __STDAIR_STDAIR_DB_HPP
00002 #define __STDAIR_STDAIR_DB_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009
00010 // Forward declarations
00011 namespace soci {
00012     class session;
00013     class statement;
00014 }
00015
00016 namespace stdair {
00017
00018     // ////////// Type definitions //////////
00020     typedef soci::session DBSession_T;
00021
00023     typedef soci::statement DBRequestStatement_T;
00024
00026     typedef std::string DBConnectionName_T;
00027
00028 }
00029 #endif // __STDAIR_STDAIR_DB_HPP

```

### 33.609 stdair/stdair\_demand\_types.hpp File Reference

```

#include <string>
#include <vector>
#include <map>
#include <boost/random/linear_congruential.hpp>
#include <boost/random/uniform_real.hpp>
#include <boost/random/variante_generator.hpp>
#include <boost/date_time/gregorian/gregorian.hpp>
#include <boost/date_time/posix_time/posix_time.hpp>
#include <boost/tuple/tuple.hpp>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/stdair_inventory_types.hpp>

```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

#### Typedefs

- typedef bool [stdair::ChangeFees\\_T](#)
- typedef bool [stdair::NonRefundable\\_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::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

### 33.610 stdair\_demand\_types.hpp

```

00001 #ifndef __STDAIR_STDAIR_DEMAND_TYPES_HPP
00002 #define __STDAIR_STDAIR_DEMAND_TYPES_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010 #include <map>
00011 // Boost Random
00012 #include <boost/random/linear_congruential.hpp>
00013 #include <boost/random/uniform_real.hpp>
00014 #include <boost/random/variante_generator.hpp>
00015 // Boost (Extended STL)
00016 #include <boost/date_time/gregorian/gregorian.hpp>
00017 #include <boost/date_time/posix_time/posix_time.hpp>
00018 #include <boost/tuple/tuple.hpp>
00019 // StdAir
00020 #include <stdair/stdair_basic_types.hpp>
00021 #include <stdair/stdair_maths_types.hpp>
00022 #include <stdair/stdair_inventory_types.hpp>
00023
00024 namespace stdair {
00025
00026 // ////////////////////////////////// Type definitions //////////////////////////////////
00027
00029 typedef bool ChangeFees_T;
00030
00032 typedef bool NonRefundable_T;
00033
00035 typedef bool SaturdayStay_T;
00036
00039 typedef double SaturdayStayRatio_T;
00040
00043 typedef double ChangeFeesRatio_T;
00044
00047 typedef double NonRefundableRatio_T;
00048
00050 typedef double Disutility_T;
00051
00054 typedef std::string PassengerType_T;
00055
00057 typedef std::string DistributionPatternId_T;
00058
00060 typedef std::string CancellationRateCurveId_T;
00061
00063 typedef std::string AirlinePreferenceId_T;
00064
00066 typedef std::pair<Percentage_T, Percentage_T> CancellationNoShowRatePair_T;
00067
00070 typedef std::string CharacteristicsPatternId_T;
00071
00073 typedef std::string CharacteristicsIndex_T;

```

```

00074
00076     typedef double WTP_T;
00077
00079     typedef boost::tuples::tuple<double, WTP_T> CharacteristicsWTP_tuple_T;
00080
00082     typedef std::pair<WTP_T, MeanStdDevPair_T> WTPDemandPair_T;
00083
00085     typedef NbOfRequests_T NbOfCancellations_T;
00086
00088     typedef NbOfRequests_T NbOfNoShows_T;
00089
00091     typedef double MatchingIndicator_T;
00092
00094     typedef std::string DemandStreamKeyStr_T;
00095
00097     typedef std::string ChannelLabel_T;
00098
00100     typedef std::string FrequentFlyer_T;
00101
00104     typedef std::string RequestStatus_T;
00105
00107     typedef std::map<Identity_T, Identity_T> BookingTSIDMap_T;
00108
00110     typedef std::pair<CabinCode_T, ClassCode_T> CabinClassPair_T;
00111
00113     typedef std::list<CabinClassPair_T> CabinClassPairList_T;
00114
00116     typedef double ProportionFactor_T;
00117
00119     typedef std::list<ProportionFactor_T> ProportionFactorList_T;
00120
00122     typedef std::string OnDString_T;
00123
00125     typedef std::list<OnDString_T> OnDStringList_T;
00126
00127 }
00128 #endif // __STDAIR_STDAIR_DEMAND_TYPES_HPP

```

### 33.611 stdair/stdair\_event\_types.hpp File Reference

```
#include <string>
```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

#### Typedefs

- typedef std::string [stdair::EventName\\_T](#)
- typedef double [stdair::NbOfEvents\\_T](#)
- typedef std::string [stdair::EventGeneratorKey\\_T](#)

### 33.612 stdair\_event\_types.hpp

```

00001 #ifndef __STDAIR_STDAIR_EVENT_TYPES_HPP
00002 #define __STDAIR_STDAIR_EVENT_TYPES_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009
00010 namespace stdair {
00011
00012     // ////////////////////////////////// Type definitions //////////////////////////////////
00014     typedef std::string EventName_T;
00015
00017     typedef double NbOfEvents_T;
00018

```

```

00020     typedef std::string EventGeneratorKey_T;
00021
00022 }
00023 #endif // __STDAIR_STDAIR_EVENT_TYPES_HPP

```

### 33.613 stdair/stdair\_exceptions.hpp File Reference

```
#include <string>
```

#### Classes

- class [stdair::RootException](#)  
*Root of the stdair exceptions.*
- class [stdair::FileNotFoundException](#)
- class [stdair::NonInitialisedLogServiceException](#)
- class [stdair::NonInitialisedServiceException](#)
- class [stdair::NonInitialisedContainerException](#)
- class [stdair::NonInitialisedRelationShipException](#)
- class [stdair::MemoryAllocationException](#)
- class [stdair::ObjectLinkingException](#)
- class [stdair::DocumentNotFoundException](#)
- class [stdair::ParserException](#)
- class [stdair::SerialisationException](#)
- class [stdair::KeyNotFoundException](#)
- class [stdair::CodeConversionException](#)
- class [stdair::CodeDuplicationException](#)
- class [stdair::KeyDuplicationException](#)
- class [stdair::ObjectCreationDuplicationException](#)
- class [stdair::ObjectNotFoundException](#)
- class [stdair::ParsingFileFailedException](#)
- class [stdair::SQLDatabaseException](#)
- class [stdair::NonInitialisedDBSessionManagerException](#)
- class [stdair::SQLDatabaseConnectionImpossibleException](#)
- class [stdair::EventException](#)
- class [stdair::SimpleNestingStructException](#)
- class [stdair::BookingClassListEmptyInNestingStructException](#)

#### Namespaces

- [stdair](#)  
*Handle on the StdAir library context.*

### 33.614 stdair\_exceptions.hpp

```

00001 #ifndef __STDAIR_STDAIR_EXCEPTIONS_HPP
00002 #define __STDAIR_STDAIR_EXCEPTIONS_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009
00010 namespace stdair {
00011
00019     class RootException : public std::exception {
00020     public:
00024         RootException (const std::string& iWhat) : _what (iWhat) {}

```

```

00028     RootException() : _what ("No further details") {}
00029
00033     virtual ~RootException() throw() {}
00034
00038     const char* what() const throw() {
00039         return _what.c_str();
00040     }
00041
00042     protected:
00046         std::string _what;
00047     };
00048
00050     class FileNotFoundException : public RootException {
00051     public:
00053         FileNotFoundException (const std::string& iWhat) :
RootException (iWhat) {}
00054     };
00055
00057     class NonInitialisedLogServiceException : public
RootException {
00058     public:
00060         NonInitialisedLogServiceException (const std::string& iWhat)
00061             : RootException (iWhat) {}
00062     };
00063
00065     class NonInitialisedServiceException : public
RootException {
00066     public:
00068         NonInitialisedServiceException (const std::string& iWhat)
00069             : RootException (iWhat) {}
00070     };
00071
00073     class NonInitialisedContainerException : public
RootException {
00074     public:
00076         NonInitialisedContainerException (const std::string& iWhat)
00077             : RootException (iWhat) {}
00078     };
00079
00081     class NonInitialisedRelationshipException : public
RootException {
00082     public:
00084         NonInitialisedRelationshipException (const std::string& iWhat)
00085             : RootException (iWhat) {}
00086     };
00087
00089     class MemoryAllocationException : public
RootException {
00090     public:
00092         MemoryAllocationException (const std::string& iWhat)
00093             : RootException (iWhat) {}
00094     };
00095
00097     class ObjectLinkingException : public RootException {
00098     public:
00100         ObjectLinkingException (const std::string& iWhat) :
RootException (iWhat) {}
00101     };
00102
00104     class DocumentNotFoundException : public
RootException {
00105     public:
00107         DocumentNotFoundException (const std::string& iWhat)
00108             : RootException (iWhat) {}
00109     };
00110
00112     class ParserException : public RootException {
00113     public:
00115         ParserException (const std::string& iWhat) : RootException (iWhat) {}
00116     };
00117
00119     class SerialisationException : public RootException {
00120     public:
00122         SerialisationException (const std::string& iWhat) :
RootException (iWhat) {}
00123     };
00124
00126     class KeyNotFoundException : public RootException {
00127     public:
00129         KeyNotFoundException (const std::string& iWhat) :
RootException (iWhat) {}
00130     };
00131
00133     class CodeConversionException : public ParserException {
00134     public:
00136         CodeConversionException (const std::string& iWhat)
00137             : ParserException (iWhat) {}

```

```

00138     };
00139
00141     class CodeDuplicationException : public
ParserException {
00142     public:
00144         CodeDuplicationException (const std::string& iWhat)
00145             : ParserException(iWhat) {}
00146     };
00147
00149     class KeyDuplicationException : public ParserException {
00150     public:
00152         KeyDuplicationException (const std::string& iWhat)
00153             : ParserException(iWhat) {}
00154     };
00155
00157     class ObjectCreationDuplicationException : public
ParserException {
00158     public:
00160         ObjectCreationDuplicationException (const std::string& iWhat)
00161             : ParserException (iWhat) {}
00162     };
00163
00165     class ObjectNotFoundException : public RootException {
00166     public:
00168         ObjectNotFoundException (const std::string& iWhat)
00169             : RootException (iWhat) {}
00170     };
00171
00173     class ParsingFileFailedException : public
ParserException {
00174     public:
00176         ParsingFileFailedException (const std::string& iWhat)
00177             : ParserException (iWhat) {}
00178     };
00179
00181     class SQLiteDatabaseException : public RootException {
00182     public:
00184         SQLiteDatabaseException (const std::string& iWhat) :
RootException (iWhat) {}
00185     };
00186
00188     class NonInitialisedDBSessionManagerException : public
RootException {
00189     public:
00191         NonInitialisedDBSessionManagerException (const std::string&
iWhat)
00192             : RootException (iWhat) {}
00193     };
00194
00196     class SQLiteDatabaseConnectionImpossibleException : public
SQLDatabaseException {
00197     public:
00199         SQLiteDatabaseConnectionImpossibleException (const std::string&
iWhat)
00200             : SQLiteDatabaseException (iWhat) {}
00201     };
00202
00204     class EventException : public RootException {
00205     public:
00207         EventException (const std::string& iWhat) : RootException (iWhat) {}
00208     };
00209
00211     class SimpleNestingStructException : public
RootException {
00212     public:
00214         SimpleNestingStructException (const std::string& iWhat)
00215             : RootException (iWhat) {}
00216     };
00217
00219     class BookingClassListEmptyInNestingStructException :
public SimpleNestingStructException {
00220     public:
00223         BookingClassListEmptyInNestingStructException (const
std::string& iWhat)
00224             : SimpleNestingStructException (iWhat) {}
00225     };
00226
00227 }
00228 #endif // __STDAIR_STDAIR_EXCEPTIONS_HPP

```

## 33.615 stdair/stdair\_fare\_types.hpp File Reference

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

## Typedefs

- typedef double [stdair::NbOfFareRules\\_T](#)

### 33.616 stdair\_fare\_types.hpp

```
00001 #ifndef __STDAIR_STDAIR_FARE_TYPES_HPP
00002 #define __STDAIR_STDAIR_FARE_TYPES_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007
00008 namespace stdair {
00009
00010     // ////////// Type definitions //////////
00012     typedef double NbOfFareRules_T;
00013
00014 }
00015 #endif // __STDAIR_STDAIR_FARE_TYPES_HPP
```

### 33.617 stdair/stdair\_file.hpp File Reference

```
#include <string>
#include <boost/utility.hpp>
#include <stdair/stdair_basic_types.hpp>
```

## Classes

- class [stdair::RootFilePath](#)  
*Root of the input and output files.*
- class [stdair::InputFilePath](#)
- class [stdair::ScheduleFilePath](#)
- class [stdair::ODFilePath](#)
- class [stdair::FRAT5FilePath](#)
- class [stdair::FFDisutilityFilePath](#)
- class [stdair::ConfigINFile](#)

## Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

### 33.618 stdair\_file.hpp

```
00001 #ifndef __STDAIR_STDAIR_FILE_HPP
00002 #define __STDAIR_STDAIR_FILE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // Boost
```

```

00010 #include <boost/utility.hpp>
00011 // StdAir
00012 #include <stdair/stdair_basic_types.hpp>
00013
00014 namespace stdair {
00015
00022     class RootFilePath {
00023     public:
00027         RootFilePath (const Filename_T& iFilename) :
00028             _filename (iFilename) {}
00032         RootFilePath () : _filename ("MyFilename") {}
00033
00037         virtual ~RootFilePath() {}
00038
00042         const char * name() const {
00043             return _filename.c_str();
00044         }
00045
00046     protected:
00050         const Filename_T _filename;
00051     };
00052
00054     class InputFilePath : public RootFilePath {
00055     public:
00057         InputFilePath (const Filename_T& iFilename) :
00058             RootFilePath (iFilename) {}
00059     };
00060
00064     class ScheduleFilePath : public InputFilePath {
00065     public:
00069         explicit ScheduleFilePath (const Filename_T& iFilename)
00070             : InputFilePath (iFilename) {}
00071     };
00072
00076     class ODFilePath : public InputFilePath {
00077     public:
00081         explicit ODFilePath (const Filename_T& iFilename)
00082             : InputFilePath (iFilename) {}
00083     };
00084
00088     class FRAT5FilePath : public InputFilePath {
00089     public:
00093         explicit FRAT5FilePath (const Filename_T& iFilename)
00094             : InputFilePath (iFilename) {}
00095     };
00096
00100     class FFDisutilityFilePath : public InputFilePath {
00101     public:
00105         explicit FFDisutilityFilePath (const Filename_T& iFilename)
00106             : InputFilePath (iFilename) {}
00107     };
00108
00112     class ConfigINIFile : public InputFilePath {
00113     public:
00117         explicit ConfigINIFile (const Filename_T& iFilename)
00118             : InputFilePath (iFilename) {}
00119     };
00120 };
00121
00122 }
00123 #endif // __STDAIR_STDAIR_FILE_HPP

```

### 33.619 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

- [stdair](#)



*Handle on the StdAir library context.*

## Typedefs

- typedef std::string [stdair::NetworkID\\_T](#)
- typedef std::vector< AirlineCode\_T > [stdair::AirlineCodeList\\_T](#)
- typedef std::vector< ClassList\_String\_T > [stdair::ClassList\\_StringList\\_T](#)
- typedef std::vector< ClassCode\_T > [stdair::ClassCodeList\\_T](#)
- typedef unsigned short [stdair::SubclassCode\\_T](#)
- typedef std::string [stdair::FlightPathCode\\_T](#)
- typedef std::map< CabinCode\_T, ClassList\_String\_T > [stdair::CabinBookingClassMap\\_T](#)
- typedef 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::DTDFrattMap\\_T](#)
- typedef std::map< FloatDuration\_T, float > [stdair::DTDProbMap\\_T](#)
- typedef std::vector< CensorshipFlag\_T > [stdair::CensorshipFlagList\\_T](#)
- typedef double [stdair::BookingRatio\\_T](#)
- typedef double [stdair::Yield\\_T](#)
- typedef unsigned int [stdair::YieldLevel\\_T](#)
- typedef std::map< YieldLevel\_T, MeanStdDevPair\_T > [stdair::YieldLevelDemandMap\\_T](#)
- typedef std::pair< Yield\_T, MeanStdDevPair\_T > [stdair::YieldDemandPair\\_T](#)
- typedef double [stdair::BidPrice\\_T](#)
- typedef std::vector< BidPrice\_T > [stdair::BidPriceVector\\_T](#)
- typedef unsigned int [stdair::SeatIndex\\_T](#)
- typedef std::string [stdair::ControlMode\\_T](#)
- typedef double [stdair::OverbookingRate\\_T](#)
- typedef double [stdair::ProtectionLevel\\_T](#)
- typedef std::vector< double > [stdair::EmsrValueList\\_T](#)
- typedef std::vector< double > [stdair::BookingLimitVector\\_T](#)
- typedef std::vector< double > [stdair::ProtectionLevelVector\\_T](#)
- typedef boost::multi\_array< double, 2 > [stdair::SnapshotBlock\\_T](#)
- typedef SnapshotBlock\_T::index\_range [stdair::SnapshotBlockRange\\_T](#)
- typedef SnapshotBlock\_T::array\_view< 1 >::type [stdair::SegmentCabinDTDSnapshotView\\_T](#)
- typedef SnapshotBlock\_T::array\_view< 2 >::type [stdair::SegmentCabinDTRangeSnapshotView\\_T](#)
- typedef SnapshotBlock\_T::const\_array\_view< 1 >::type [stdair::ConstSegmentCabinDTDSnapshotView\\_T](#)
- typedef SnapshotBlock\_T::const\_array\_view< 2 >::type [stdair::ConstSegmentCabinDTRangeSnapshotView\\_T](#)
- typedef unsigned short [stdair::SegmentDataID\\_T](#)
- typedef unsigned short [stdair::LegDataID\\_T](#)
- typedef unsigned short [stdair::ClassIndex\\_T](#)

## 33.620 stdair\_inventory\_types.hpp

```

00001 #ifndef __STDAIR_STDAIR_INVENTORY_TYPES_HPP
00002 #define __STDAIR_STDAIR_INVENTORY_TYPES_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010 #include <map>
00011 #include <list>
00012 // BOOST
00013 #include <boost/multi_array.hpp>
00014 // StdAir
00015 #include <stdair/stdair_basic_types.hpp>
00016 #include <stdair/stdair_maths_types.hpp>
00017 #include <stdair/stdair_date_time_types.hpp>
00018
00019 namespace stdair {
00020
00021 // ////////// Type definitions //////////
00023 typedef std::string NetworkID_T;
00024
00026 typedef std::vector<AirlineCode_T> AirlineCodeList_T;
00027
00029 typedef std::vector<ClassList_String_T> ClassList_StringList_T;
00030
00032 typedef std::vector<ClassCode_T> ClassCodeList_T;
00033
00037 typedef unsigned short SubclassCode_T;
00038
00040 typedef std::string FlightPathCode_T;
00041
00044 typedef std::map<CabinCode_T, ClassList_String_T> CabinBookingClassMap_T;
00045
00047 typedef std::string CurveKey_T;
00048
00051 typedef double CabinCapacity_T;
00052
00054 typedef double NbofFlightDates_T;
00055
00057 typedef double CommittedSpace_T;
00058
00060 typedef double UPR_T;
00061
00063 typedef double BookingLimit_T;
00064
00066 typedef double AuthorizationLevel_T;
00067
00069 typedef double CapacityAdjustment_T;
00070
00072 typedef double BlockSpace_T;
00073
00075 typedef bool AvailabilityStatus_T;
00076
00078 typedef std::vector<Availability_T> BucketAvailabilities_T;
00079
00081 typedef double NbofYields_T;
00082
00084 typedef double NbofInventoryControlRules_T;
00085
00087 typedef bool CensorshipFlag_T;
00088
00090 typedef short DTD_T;
00091
00093 typedef short DCP_T;
00094
00096 typedef std::list<DCP_T> DCPList_T;
00097
00099 typedef std::map<DTD_T, RealNumber_T> DTDFractMap_T;
00100
00102 typedef std::map<FloatDuration_T, float> DTDProbMap_T;
00103
00106 typedef std::vector<CensorshipFlag_T> CensorshipFlagList_T;
00107
00110 typedef double BookingRatio_T;
00111
00113 typedef double Yield_T;
00114
00116 typedef unsigned int YieldLevel_T;
00117
00119 typedef std::map<YieldLevel_T, MeanStdDevPair_T> YieldLevelDemandMap_T;
00120
00122 typedef std::pair<Yield_T, MeanStdDevPair_T> YieldDemandPair_T;
00123

```

```

00125     typedef double BidPrice_T;
00126
00128     typedef std::vector<BidPrice_T> BidPriceVector_T;
00129
00131     typedef unsigned int SeatIndex_T;
00132
00134     typedef std::string ControlMode_T;
00135
00137     typedef double OverbookingRate_T;
00138
00141     typedef double BookingLimit_T;
00142
00145     typedef double ProtectionLevel_T;
00146
00148     typedef std::vector<double> EmsrValueList_T;
00149
00152     typedef std::vector<double> BookingLimitVector_T;
00153
00156     typedef std::vector<double> ProtectionLevelVector_T;
00157
00159     typedef boost::multi_array<double, 2> SnapshotBlock_T;
00160
00162     typedef SnapshotBlock_T::index_range SnapshotBlockRange_T;
00163
00165     typedef SnapshotBlock_T::array_view<1>::type SegmentCabinDTDSnapshotView_T;
00166
00168     typedef SnapshotBlock_T::array_view<2>::type
SegmentCabinDTRangeSnapshotView_T;
00169
00171     typedef SnapshotBlock_T::const_array_view<1>::type
ConstSegmentCabinDTDSnapshotView_T;
00172
00174     typedef SnapshotBlock_T::const_array_view<2>::type
ConstSegmentCabinDTRangeSnapshotView_T;
00175
00177     typedef unsigned short SegmentDataID_T;
00178
00180     typedef unsigned short LegDataID_T;
00181
00184     typedef unsigned short ClassIndex_T;
00185
00186 }
00187 #endif // __STDAIR_STDAIR_INVENTORY_TYPES_HPP

```

### 33.621 stdair/stdair\_json.hpp File Reference

```
#include <string>
```

#### Classes

- class [stdair::JSONString](#)  
*JSON-formatted string.*

#### Namespaces

- [stdair](#)  
*Handle on the StdAir library context.*

### 33.622 stdair\_json.hpp

```

00001 #ifndef __STDAIR_STDAIR_JSON_HPP
00002 #define __STDAIR_STDAIR_JSON_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009
00010 namespace stdair {
00011
00016     class JSONString {

```

```

00017 public:
00021     explicit JSONString (const std::string& iJsonString) :
00022         _jsonString (iJsonString) {}
00026     explicit JSONString () : _jsonString ("" ) {}
00027
00031     virtual ~JSONString() {}
00032
00036     const std::string& getString() const {
00037         return _jsonString;
00038     }
00039
00040 protected:
00044     std::string _jsonString;
00045 };
00046
00047 }
00048 #endif // __STDAIR_STDAIR_JSON_HPP

```

### 33.623 stdair/stdair\_log.hpp File Reference

```
#include <string>
```

#### Namespaces

- [stdair](#)  
*Handle on the StdAir library context.*
- [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]

### 33.624 stdair\_log.hpp

```

00001 #ifndef __STDAIR_STDAIR_LOG_HPP
00002 #define __STDAIR_STDAIR_LOG_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009
00010 namespace stdair {
00011
00012     // Forward declarations
00013     class STDAIR_Service;
00014
00015     // ////////////////////////////////// Log //////////////////////////////////
00017     namespace LOG {
00018         typedef enum {
00019             CRITICAL = 0,
00020             ERROR,
00021             NOTIFICATION,
00022             WARNING,
00023             DEBUG,
00024             VERBOSE,
00025             LAST_VALUE
00026         } EN_LogLevel;
00027
00028         static const std::string _logLevels[LAST_VALUE] =
00029             {"C", "E", "N", "W", "D", "V"};

```

```

00030     }
00031
00032 }
00033 #endif // __STDAIR_STDAIR_LOG_HPP

```

### 33.625 stdair/stdair\_maths\_types.hpp File Reference

```

#include <string>
#include <vector>
#include <map>
#include <boost/random/linear_congruential.hpp>
#include <boost/random/uniform_real.hpp>
#include <boost/random/normal_distribution.hpp>
#include <boost/random/exponential_distribution.hpp>
#include <boost/random/variante_generator.hpp>

```

#### Namespaces

- [stdair](#)

*Handle on the StdAir library context.*

#### Typedefs

- typedef unsigned int [stdair::ReplicationNumber\\_T](#)
- typedef unsigned long int [stdair::ExponentialSeed\\_T](#)
- typedef unsigned long int [stdair::UniformSeed\\_T](#)
- typedef unsigned long int [stdair::RandomSeed\\_T](#)
- typedef boost::minstd\_rand [stdair::BaseGenerator\\_T](#)
- typedef boost::uniform\_real [stdair::UniformDistribution\\_T](#)
- typedef boost::variante\_generator< BaseGenerator\_T &, UniformDistribution\_T > [stdair::UniformGenerator\\_T](#)
- typedef boost::normal\_distribution [stdair::NormalDistribution\\_T](#)
- typedef boost::variante\_generator< BaseGenerator\_T &, NormalDistribution\_T > [stdair::NormalGenerator\\_T](#)
- typedef boost::exponential\_distribution [stdair::ExponentialDistribution\\_T](#)
- typedef boost::variante\_generator< BaseGenerator\_T &, ExponentialDistribution\_T > [stdair::ExponentialGenerator\\_T](#)
- typedef double [stdair::MeanValue\\_T](#)
- typedef double [stdair::StdDevValue\\_T](#)
- typedef std::pair< MeanValue\_T, StdDevValue\_T > [stdair::MeanStdDevPair\\_T](#)
- typedef std::vector< MeanStdDevPair\_T > [stdair::MeanStdDevPairVector\\_T](#)
- typedef float [stdair::Probability\\_T](#)

### 33.626 stdair\_maths\_types.hpp

```

00001 #ifndef __STDAIR_STDAIR_MATHS_TYPES_HPP
00002 #define __STDAIR_STDAIR_MATHS_TYPES_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010 #include <map>
00011 // Boost Random
00012 #include <boost/random/linear_congruential.hpp>
00013 #include <boost/random/uniform_real.hpp>
00014 #include <boost/random/normal_distribution.hpp>
00015 #include <boost/random/exponential_distribution.hpp>

```

```

00016 #include <boost/random/variante_generator.hpp>
00017
00018 namespace stdair {
00019
00020     // ////////// Type definitions //////////
00024     typedef unsigned int ReplicationNumber_T;
00025
00029     typedef unsigned long int ExponentialSeed_T;
00030
00034     typedef unsigned long int UniformSeed_T;
00035
00039     typedef unsigned long int RandomSeed_T;
00040
00044     typedef boost::minstd_rand BaseGenerator_T;
00045
00049     typedef boost::uniform_real<> UniformDistribution_T;
00050
00054     typedef boost::variate_generator<BaseGenerator_T&,
00055                                     UniformDistribution_T> UniformGenerator_T;
00056
00060     typedef boost::normal_distribution<> NormalDistribution_T;
00061
00065     typedef boost::variate_generator<BaseGenerator_T&,
00066                                     NormalDistribution_T> NormalGenerator_T;
00067
00069     typedef boost::exponential_distribution<> ExponentialDistribution_T;
00070
00071
00073     typedef boost::variate_generator<BaseGenerator_T&,
00074                                     ExponentialDistribution_T>
ExponentialGenerator_T;
00075
00079     typedef double MeanValue_T;
00080
00084     typedef double StdDevValue_T;
00085
00089     typedef std::pair<MeanValue_T, StdDevValue_T> MeanStdDevPair_T;
00090
00094     typedef std::vector<MeanStdDevPair_T> MeanStdDevPairVector_T;
00095
00099     typedef float Probability_T;
00100
00101 }
00102 #endif // __STDAIR_STDAIR_MATHS_TYPES_HPP

```

### 33.627 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

- [stdair](#)

*Handle on the StdAir library context.*

#### Typedefs

- typedef std::string [stdair::ForecasterMode\\_T](#)
- typedef short [stdair::HistoricalDataLimit\\_T](#)
- typedef std::string [stdair::OptimizerMode\\_T](#)
- typedef NbOfBookings\_T [stdair::PolicyDemand\\_T](#)
- typedef std::vector< double > [stdair::GeneratedDemandVector\\_T](#)
- typedef std::vector< GeneratedDemandVector\_T > [stdair::GeneratedDemandVectorHolder\\_T](#)
- typedef double [stdair::SellupProbability\\_T](#)
- typedef std::vector< 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 unsigned int [stdair::NbOfSamples\\_T](#)

### 33.628 stdair\_rm\_types.hpp

```

00001 #ifndef __STDAIR_STDAIR_RM_TYPES_HPP
00002 #define __STDAIR_STDAIR_RM_TYPES_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010 #include <map>
00011 // StdAir
00012 #include <stdair/stdair_basic_types.hpp>
00013 #include <stdair/stdair_inventory_types.hpp>
00014
00015 namespace stdair {
00016     // Forward declarations.
00017     class BookingClass;
00018
00019     // ////////// Type definitions //////////
00021     typedef std::string ForecasterMode_T;
00022
00024     typedef short HistoricalDataLimit_T;
00025
00027     typedef std::string OptimizerMode_T;
00028
00030     typedef NbOfBookings_T PolicyDemand_T;
00031
00034     typedef std::vector<double> GeneratedDemandVector_T;
00035
00037     typedef std::vector<GeneratedDemandVector_T> GeneratedDemandVectorHolder_T;
00038
00040     typedef double SellupProbability_T;
00041
00043     typedef std::vector<NbOfRequests_T> UncDemVector_T;
00044
00046     typedef std::vector<NbOfBookings_T> BookingVector_T;
00047
00049     typedef double FRAT5_T;
00050
00052     typedef std::map<const DTD_T, FRAT5_T> FRAT5Curve_T;
00053
00055     typedef std::map<const DTD_T, double> FFDisutilityCurve_T;
00056
00058     typedef std::map<const DTD_T, double> SellUpCurve_T;
00059
00061     typedef std::map<const DTD_T, double> DispatchingCurve_T;
00062
00064     typedef std::map<BookingClass*, SellUpCurve_T> BookingClassSellUpCurveMap_T;
00065
00067     typedef std::map<BookingClass*, DispatchingCurve_T>
BookingClassDispatchingCurveMap_T;
00068
00071     typedef std::map<const Yield_T, double> YieldDemandMap_T;
00072
00074     typedef double Revenue_T;
00075
00077     typedef unsigned int NbOfSamples_T;
00078
00079 }
00080 #endif // __STDAIR_STDAIR_RM_TYPES_HPP

```

### 33.629 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>
```

## Classes

- class [stdair::STDAIR\\_Service](#)  
*Interface for the STDAIR Services.*

## Namespaces

- [stdair](#)  
*Handle on the StdAir library context.*

## 33.630 STDAIR\_Service.hpp

```
00001
00012 #ifndef __STDAIR_STDAIR_HPP
00013 #define __STDAIR_STDAIR_HPP
00014
00015 // //////////////////////////////////////
00016 // Import section
00017 // //////////////////////////////////////
00018 // STL
00019 #include <string>
00020 // StdAir
00021 #include <stdair/stdair_inventory_types.hpp>
00022 #include <stdair/stdair_service_types.hpp>
00023 #include <stdair/stdair_file.hpp>
00024 #include <stdair/basic/BasLogParams.hpp>
00025 #include <stdair/basic/BasDBParams.hpp>
00026 #include <stdair/basic/ServiceInitialisationType.hpp>
00027 #include <stdair/bom/TravelSolutionTypes.hpp>
00028 #include <stdair/bom/ConfigHolderStruct.hpp>
00029 #include <stdair/service/STDAIR_ServiceContext.hpp>
00030
00031 namespace stdair {
00032
00033     class BomRoot;
00034     struct EventStruct;
00035     struct ProgressStatusSet;
00036     struct BookingRequestStruct;
00037     class JSONString;
00038
00039
00040
00041     class STDAIR_Service {
00042     public:
00043         // ////////////////////////////////// Constructors and destructors //////////////////////////////////
00044         STDAIR_Service();
00045
00046         STDAIR_Service (const BasLogParams&);
00047
00048         STDAIR_Service (const BasLogParams&, const
00049 BasDBParams&);
00050
00051         ~STDAIR_Service();
00052
00053     public:
00054         // ////////////////////////////////// Business support methods //////////////////////////////////
00055         void buildSampleBom();
00056
00057         void buildDummyInventory (const CabinCapacity_T& iCabinCapacity);
00058
00059         void buildDummyLegSegmentAccesses (BomRoot&);
00060
00061
00062
00063
00064
00065
00066
00067
00068
00069
00070
00071
00072
00073
00074
00075
00076
00077
00078
00079
00080
00081
00082
00083
00084
00085
00086
00087
00088
00089
00090
00091
00092
00093
00094
00095
00096
00097
00098
00099
00100
00101
00102
00103
00104
00105
00106
00107
00108
00109
00110
00111
00112
00113
00114
00115
00116
00117
00118
00119
00120
00121
00122
00123
00124
00125
00126
00127
00128
00129
00130
00131
00132
00133
00134
00135
00136
00137
00138
00139
00140
00141
00142
00143
00144
00145
00146
```



```

00147
00162     void buildSampleTravelSolutionForPricing (
00163         TravelSolutionList_T&);
00163
00181     void buildSampleTravelSolutions (
00182         TravelSolutionList_T&);
00182
00211     BookingRequestStruct buildSampleBookingRequest (const bool
00212         isForCRS = false);
00212
00216     void clonePersistentBom ();
00217
00218     public:
00219
00220         // //////////// Export support methods ////////////
00233         std::string jsonExportFlightDateList (const
00234             AirlineCode_T& iAirlineCode = "all",
00235                                     const FlightNumber_T& iFlightNumber = 0) const;
00235
00246         std::string jsonExportFlightDateObjects (const
00247             AirlineCode_T&,
00248                                     const FlightNumber_T&,
00249                                     const Date_T& iDepartureDate) const;
00249
00256         std::string jsonExportEventObject (const EventStruct&) const;
00257
00264         std::string jsonExportConfiguration () const;
00265
00266     public:
00267
00268         // //////////// Import support methods ////////////
00276         bool jsonImportConfiguration (const JSONString&) const;
00277
00278     public:
00279         // //////////// Display support methods ////////////
00293         std::string list (const AirlineCode_T& iAirlineCode = "all",
00294                             const FlightNumber_T& iFlightNumber = 0) const;
00295
00302         std::string listAirportPairDateRange () const;
00303
00313         bool check (const AirlineCode_T&, const FlightNumber_T&,
00314                     const Date_T& iDepartureDate) const;
00315
00328         bool check (const AirportCode_T&, const AirportCode_T&,
00329                     const Date_T& iDepartureDate) const;
00330
00337         std::string configDisplay () const;
00338
00346         std::string csvDisplay () const;
00347
00356         std::string csvDisplay (const BomRoot&) const;
00357
00367         std::string csvDisplay (const AirlineCode_T&, const
00368             FlightNumber_T&,
00369                                     const Date_T& iDepartureDate) const;
00377         std::string csvDisplay (const TravelSolutionList_T&) const;
00378
00389         std::string csvDisplay (const AirportCode_T&, const
00390             AirportCode_T&,
00391                                     const Date_T& iDepartureDate) const;
00392
00393     public:
00394         // //////////// Getters ////////////
00403         BomRoot& getBomRoot () const;
00404
00413         BomRoot& getPersistentBomRoot () const;
00414
00420         BasLogParams getLogParams () const;
00421
00428         const BasDBParams& getDBParams () const;
00429
00438         const ServiceInitialisationType&
00439         getServiceInitialisationType () const;
00440
00441     private:
00442         // ////////// Construction and Destruction helper methods //////////
00449         STDAIR_Service (const STDAIR_Service&);
00450
00455         void initServiceContext();
00456
00474         void logInit (const BasLogParams&);
00475
00481         void dbInit (const BasDBParams&);
00482

```

```

00498     void init();
00499
00503     void finalise();
00504
00505     public:
00506
00512     void importINIConfig (const ConfigINIFile&);
00513
00522     void importConfigValue (const std::string& iValue,
00523                             const std::string& iPath);
00524
00533     template <typename ValueType>
00534     bool exportConfigValue (ValueType& ioValue, const std::string& iPath);
00535
00540     void updateAirlineFeatures ();
00541
00542     private:
00543         // ////////// Service Context //////////
00547         STDAIR_ServiceContext* _stdairServiceContext;
00548     };
00549
00550     // ////////////////////////////////////////////
00551     template <typename ValueType>
00552     bool STDAIR_Service::exportConfigValue (ValueType& ioValue,
00553                                             const std::string& iPath) {
00554
00555         // Retrieve the StdAir service context
00556         assert (_stdairServiceContext != NULL);
00557         const STDAIR_ServiceContext& lSTDAIR_ServiceContext =
00558             *_stdairServiceContext;
00559
00560         // Retrieve the BOM tree root
00561         const ConfigHolderStruct& lConfigHolder =
00562             lSTDAIR_ServiceContext.getConfigHolder();
00563
00564         // Call the dedicated configuration holder method.
00565         return lConfigHolder.exportValue <ValueType> (ioValue, iPath);
00566     }
00567     // ////////////////////////////////////////////
00568
00569 }
00570 #endif // __STDAIR_STDAIR_HPP

```

### 33.631 stdair/stdair\_service\_types.hpp File Reference

```
#include <boost/shared_ptr.hpp>
```

#### Namespaces

- [stdair](#)  
Handle on the StdAir library context.

#### Typedefs

- typedef boost::shared\_ptr< STDAIR\_Service > [stdair::STDAIR\\_ServicePtr\\_T](#)

### 33.632 stdair\_service\_types.hpp

```

00001 #ifndef __STDAIR_STDAIR_SERVICE_HPP
00002 #define __STDAIR_STDAIR_SERVICE_HPP
00003
00004 // ////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////
00007 // Boost (Extended STL)
00008 #include <boost/shared_ptr.hpp>
00009
00010 namespace stdair {
00011
00012     // Forward declarations
00013     class STDAIR_Service;
00014

```

```

00016     typedef boost::shared_ptr<STDAIR_Service> STDAIR_ServicePtr_T;
00017
00018 }
00019 #endif // __STDAIR_STDAIR_SERVICE_HPP

```

### 33.633 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>

```

### 33.634 stdair\_types.hpp

```

00001 #ifndef __STDAIR_STDAIR_TYPES_HPP
00002 #define __STDAIR_STDAIR_TYPES_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_exceptions.hpp>
00009 #include <stdair/stdair_log.hpp>
00010 #include <stdair/stdair_db.hpp>
00011 #include <stdair/stdair_basic_types.hpp>
00012 #include <stdair/stdair_demand_types.hpp>
00013 #include <stdair/stdair_maths_types.hpp>
00014 #include <stdair/stdair_fare_types.hpp>
00015 #include <stdair/stdair_inventory_types.hpp>
00016 #include <stdair/stdair_rm_types.hpp>
00017 #include <stdair/stdair_date_time_types.hpp>
00018 #include <stdair/stdair_service_types.hpp>
00019
00020 #endif // __STDAIR_STDAIR_TYPES_HPP

```

### 33.635 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]

### 33.635.1 Typedef Documentation

#### 33.635.1.1 typedef int(\* [pt2Func](#)) (char \*)

Definition at line 35 of file [readline\\_autocomp.hpp](#).

### 33.635.2 Function Documentation

#### 33.635.2.1 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\(\)](#).

#### 33.635.2.2 char\* [xmalloc](#) ( size\_t )

Referenced by [dupstr\(\)](#).

#### 33.635.2.3 void [com\\_list](#) ( char \* *arg* )

List the file(s) named in arg.

Definition at line 264 of file [readline\\_autocomp.hpp](#).

#### 33.635.2.4 `int com_view ( char * arg )`

Definition at line 274 of file [readline\\_autocomp.hpp](#).

References [valid\\_argument\(\)](#).

#### 33.635.2.5 `int com_rename ( char * arg )`

Definition at line 284 of file [readline\\_autocomp.hpp](#).

References [too\\_dangerous\(\)](#).

#### 33.635.2.6 `int com_stat ( char * arg )`

Definition at line 289 of file [readline\\_autocomp.hpp](#).

References [valid\\_argument\(\)](#).

#### 33.635.2.7 `int com_pwd ( char * ignore )`

Definition at line 367 of file [readline\\_autocomp.hpp](#).

References [getwd\(\)](#).

Referenced by [com\\_cd\(\)](#).

#### 33.635.2.8 `int com_delete ( char * arg )`

Definition at line 315 of file [readline\\_autocomp.hpp](#).

References [too\\_dangerous\(\)](#).

#### 33.635.2.9 `int com_help ( char * arg )`

Print out help for ARG, or for all of the commands if ARG is not present.

Definition at line 324 of file [readline\\_autocomp.hpp](#).

References [COMMAND::name](#).

#### 33.635.2.10 `int com_cd ( char * arg )`

Definition at line 356 of file [readline\\_autocomp.hpp](#).

References [com\\_pwd\(\)](#).

#### 33.635.2.11 `int com_quit ( char * arg )`

Definition at line 381 of file [readline\\_autocomp.hpp](#).

#### 33.635.2.12 `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](#).

#### 33.635.2.13 `COMMAND * find_command ( char * name )`

Look up NAME as the name of a command, and return a pointer to that command. Return a NULL pointer if NAME isn't a command name.

Definition at line 136 of file [readline\\_autocomp.hpp](#).

References [COMMAND::name](#).

Referenced by [execute\\_line\(\)](#).

#### 33.635.2.14 char\* dupstr ( char \* *iString* )

Duplicate a string

Definition at line 85 of file [readline\\_autocomp.hpp](#).

References [xmalloc\(\)](#).

Referenced by [command\\_generator\(\)](#).

#### 33.635.2.15 int execute\_line ( char \* *line* )

Execute a command line.

Definition at line 94 of file [readline\\_autocomp.hpp](#).

References [find\\_command\(\)](#), and [COMMAND::func](#).

#### 33.635.2.16 char \* command\_generator ( char \* *text*, int *state* )

Generator function for command completion. STATE lets us know whether to start from scratch; without any state (i.e. STATE == 0), then we start at the top of the list.

Definition at line 222 of file [readline\\_autocomp.hpp](#).

References [dupstr\(\)](#).

Referenced by [fileman\\_completion\(\)](#).

#### 33.635.2.17 char \*\* fileman\_completion ( char \* *text*, int *start*, int *end* )

Attempt to complete on the contents of TEXT. START and END bound the region of rl\_line\_buffer that contains the word to complete. TEXT is the word to complete. We can use the entire contents of rl\_line\_buffer in case we want to do some simple parsing. Return the array of matches, or NULL if there aren't any.

Definition at line 200 of file [readline\\_autocomp.hpp](#).

References [command\\_generator\(\)](#).

Referenced by [initialize\\_readline\(\)](#).

#### 33.635.2.18 void initialize\_readline ( )

Tell the GNU Readline library how to complete. We want to try to complete on command names if this is the first word in the line, or on filenames if not.

Definition at line 185 of file [readline\\_autocomp.hpp](#).

References [fileman\\_completion\(\)](#).

#### 33.635.2.19 void too\_dangerous ( char \* *caller* )

Definition at line 387 of file [readline\\_autocomp.hpp](#).

Referenced by [com\\_delete\(\)](#), and [com\\_rename\(\)](#).

#### 33.635.2.20 int valid\_argument ( char \* *caller*, char \* *arg* )

Definition at line 395 of file [readline\\_autocomp.hpp](#).

Referenced by [com\\_stat\(\)](#), and [com\\_view\(\)](#).

### 33.635.3 Variable Documentation

#### 33.635.3.1 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](#).

### 33.635.3.2 int done

When non-zero, this global means the user is done using this program.

Definition at line 80 of file [readline\\_autocomp.hpp](#).

### 33.635.3.3 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](#).

## 33.636 readline\_autocomp.hpp

```
00001
00006 #ifndef __AIRINV_READLINE_AUTOCOMP_HPP
00007 #define __AIRINV_READLINE_AUTOCOMP_HPP
00008
00009 // STL
00010 #include <string>
00011 #include <iosfwd>
00012 #include <cstdio>
00013 #include <sys/types.h>
00014 #include <sys/file.h>
00015 #include <sys/stat.h>
00016 #include <sys/errno.h>
00017
00018 #include <readline/readline.h>
00019 #include <readline/history.h>
00020
00021 extern char* getwd();
00022 extern char* xmalloc (size_t);
00023
00024 /* The names of functions that actually do the manipulation. */
00025 int com_list (char*);
00026 int com_view (char*);
00027 int com_rename (char*);
00028 int com_stat (char*);
00029 int com_pwd (char*);
00030 int com_delete (char*);
00031 int com_help (char*);
00032 int com_cd (char*);
00033 int com_quit (char*);
00034
00035 typedef int (*pt2Func) (char*);
00036
00041 typedef struct {
00045     char const* name;
00046
00050     pt2Func *func;
00051
00055     char *doc;
00056 } COMMAND;
00057
00058 COMMAND commands[] = {
00059     { "cd", (*com_cd)(), "Change to directory DIR" },
00060     { "delete", com_delete, "Delete FILE" },
00061     { "help", com_help, "Display this text" },
00062     { "?", com_help, "Synonym for 'help'" },
00063     { "list", com_list, "List files in DIR" },
00064     { "ls", com_list, "Synonym for 'list'" },
00065     { "pwd", com_pwd, "Print the current working directory" },
```

```

00066 { "quit", com_quit, "Quit using airinv" },
00067 { "rename", com_rename, "Rename FILE to NEWNAME" },
00068 { "stat", com_stat, "Print out statistics on FILE" },
00069 { "view", com_view, "View the contents of FILE" },
00070 { (char*) NULL, (pt2Func) NULL, (char*) NULL }
00071 };
00072
00073 // Forward declarations
00074 char* stripwhite (char* iString);
00075 COMMAND* find_command (char* iString);
00076
00080 int done;
00081
00085 char* dupstr (char* iString) {
00086     char* r = xmalloc (std::strlen (iString) + 1);
00087     strcpy (r, iString);
00088     return r;
00089 }
00090
00094 int execute_line (char* line) {
00095     register int i;
00096     COMMAND* command;
00097     char* word;
00098
00099     /* Isolate the command word. */
00100     i = 0;
00101     while (line[i] && whitespace (line[i])) {
00102         i++;
00103     }
00104     word = line + i;
00105
00106     while (line[i] && !whitespace (line[i])) {
00107         i++;
00108     }
00109
00110     if (line[i]) {
00111         line[i++] = '\0';
00112     }
00113
00114     command = find_command (word);
00115
00116     if (!command) {
00117         std::cerr << word << ": No such command for airinv." << std::endl;
00118         return -1;
00119     }
00120
00121     /* Get argument to command, if any. */
00122     while (whitespace (line[i])) {
00123         i++;
00124     }
00125
00126     word = line + i;
00127
00128     /* Call the function. */
00129     return (*(command->func)) (word);
00130 }
00131
00136 COMMAND* find_command (char* name) {
00137     register int i;
00138
00139     for (i = 0; commands[i].name; i++) {
00140         if (strcmp (name, commands[i].name) == 0) {
00141             return (&commands[i]);
00142         }
00143     }
00144
00145     return (COMMAND*) NULL;
00146 }
00147
00152 char* stripwhite (char* string) {
00153     register char *s, *t;
00154
00155     for (s = string; whitespace (*s); s++) {
00156     }
00157
00158     if (*s == 0) {
00159         return s;
00160     }
00161
00162     t = s + strlen (s) - 1;
00163     while (t > s && whitespace (*t)) {
00164         t--;
00165     }
00166     *++t = '\0';
00167
00168     return s;
00169 }

```



```

00170
00171 /* ***** */
00172 /* */
00173 /*          Interface to Readline Completion          */
00174 /* */
00175 /* ***** */
00176
00177 char* command_generator (char* text, int state);
00178 char** fileman_completion (char* text, int start, int end);
00179
00185 void initialize_readline() {
00186     /* Allow conditional parsing of the ~/.inputrc file. */
00187     rl_readline_name = "airinv";
00188
00189     /* Tell the completer that we want a crack first. */
00190     rl_attempted_completion_function = (rl_completion_func_t*) fileman_completion;
00191 }
00192
00200 char** fileman_completion (char* text, int start, int end) {
00201     char **matches;
00202
00203     matches = (char**) NULL;
00204
00210     if (start == 0) {
00211         matches = completion_matches (text, command_generator);
00212     }
00213
00214     return matches;
00215 }
00216
00222 char* command_generator (char* text, int state) {
00223     static int list_index, len;
00224     char* name;
00225
00231     if (!state) {
00232         list_index = 0;
00233         len = strlen (text);
00234     }
00235
00236     /* Return the next name which partially matches from the command list. */
00237     while (name = commands[list_index].name) {
00238         ++list_index;
00239
00240         if (strncmp (name, text, len) == 0) {
00241             return dupstr (name);
00242         }
00243     }
00244
00245     /* If no names matched, then return NULL. */
00246     return (char*) NULL;
00247 }
00248
00249 /* ***** */
00250 /* */
00251 /*          airinv Commands          */
00252 /* */
00253 /* ***** */
00254
00259 static char syscom[1024];
00260
00264 void com_list (char* arg) {
00265     if (!arg) {
00266         arg = "";
00267     }
00268
00269     std::ostringstream oStr;
00270     oStr << "ls -FClg " << arg;
00271     return system (oStr.c_str());
00272 }
00273
00274 int com_view (char* arg) {
00275     if (!valid_argument ("view", arg)) {
00276         return 1;
00277     }
00278
00279     std::ostringstream oStr;
00280     oStr << "more " << arg;
00281     return system (syscom);
00282 }
00283
00284 int com_rename (char* arg) {
00285     too_dangerous ("rename");
00286     return 1;
00287 }
00288
00289 int com_stat (char* arg) {
00290     struct stat finfo;

```

```

00291
00292     if (!valid_argument ("stat", arg)) {
00293         return 1;
00294     }
00295
00296     if (stat (arg, &finfo) == -1) {
00297         perror (arg);
00298         return 1;
00299     }
00300
00301     std::cout << "Statistics for '" << arg << "':" << std::endl;
00302
00303     const std::string lPluralEnd1 = (finfo.st_nlink == 1) ? "" : "s";
00304     const std::string lPluralEnd2 = (finfo.st_size == 1) ? "" : "s";
00305     std::cout << arg << " has "
00306               << finfo.st_nlink << " link" << lPluralEnd1 << ", and is "
00307               << finfo.st_size << " byte" << lPluralEnd2 << " in length."
00308               << std::endl;
00309     std::cout << " Inode Last Change at: " << ctime (&finfo.st_ctime) << std::endl;
00310     std::cout << " Last access at: " << ctime (&finfo.st_atime) << std::endl;
00311     std::cout << " Last modified at: " << ctime (&finfo.st_mtime) << std::endl;
00312     return 0;
00313 }
00314
00315 int com_delete (char* arg) {
00316     too_dangerous ("delete");
00317     return 1;
00318 }
00319
00324 int com_help (char* arg) {
00325     register int i;
00326     int printed = 0;
00327
00328     for (i = 0; commands[i].name; i++) {
00329         if (!*arg || (strcmp (arg, commands[i].name) == 0)) {
00330             printf ("%s\t\t%s.\n", commands[i].name, commands[i].doc);
00331             printed++;
00332         }
00333     }
00334
00335     if (!printed) {
00336         printf ("No commands match '%s'. Possibilities are:\n", arg);
00337
00338         for (i = 0; commands[i].name; i++) {
00339             /* Print in six columns. */
00340             if (printed == 6) {
00341                 printed = 0;
00342                 printf ("\n");
00343             }
00344
00345             printf ("%s\t", commands[i].name);
00346             printed++;
00347         }
00348
00349         if (printed)
00350             printf ("\n");
00351     }
00352     return 0;
00353 }
00354
00355 /* Change to the directory ARG. */
00356 int com_cd (char* arg) {
00357     if (chdir (arg) == -1) {
00358         perror (arg);
00359         return 1;
00360     }
00361
00362     com_pwd ("");
00363     return 0;
00364 }
00365
00366 /* Print out the current working directory. */
00367 int com_pwd (char* ignore) {
00368     char dir[1024], *s;
00369
00370     s = getwd (dir);
00371     if (s == 0) {
00372         printf ("Error getting pwd: %s\n", dir);
00373         return 1;
00374     }
00375
00376     printf ("Current directory is %s\n", dir);
00377     return 0;
00378 }
00379
00380 /* The user wishes to quit using this program. Just set DONE non-zero. */
00381 int com_quit (char* arg) {

```

```

00382     done = 1;
00383     return 0;
00384 }
00385
00386 /* Function which tells you that you can't do this. */
00387 void too_dangerous (char* caller) {
00388     fprintf (stderr,
00389             "%s: Too dangerous for me to distribute. Write it yourself.\n",
00390             caller);
00391 }
00392
00393 /* Return non-zero if ARG is a valid argument for CALLER, else print
00394  * an error message and return zero. */
00395 int valid_argument (char* caller, char* arg) {
00396     if (!arg || !*arg) {
00397         fprintf (stderr, "%s: Argument required.\n", caller);
00398         return 0;
00399     }
00400
00401     return 1;
00402 }
00403
00404 #endif // _AIRINV_READLINE_AUTOCOMP_HPP

```

### 33.637 stdair/ui/cmdline/SReadline.hpp File Reference

C++ wrapper around libreadline.

```

#include <stdio.h>
#include <readline/readline.h>
#include <readline/history.h>
#include <readline/keymaps.h>
#include <string>
#include <fstream>
#include <vector>
#include <stdexcept>
#include <map>
#include <boost/algorithm/string/trim.hpp>
#include <boost/tokenizer.hpp>
#include <boost/function.hpp>

```

#### Classes

- class [swift::SKeymap](#)  
*The readline keymap wrapper.*
- class [swift::SReadline](#)  
*The readline library wrapper.*

#### Namespaces

- [swift](#)  
*The wrapper namespace.*

#### 33.637.1 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](#).

## 33.638 SReadline.hpp

```

00001
00011 //
00012 // Date:      17 December 2005
00013 //           03 April   2006
00014 //           20 April   2006
00015 //           07 May     2006
00016 //
00017 // Copyright (c) Sergey Satskiy 2005 - 2006
00018 //           <sergesatsky@yahoo.com>
00019 //
00020 // Permission to copy, use, modify, sell and distribute this software
00021 // is granted provided this copyright notice appears in all copies.
00022 // This software is provided "as is" without express or implied
00023 // warranty, and with no claim as to its suitability for any purpose.
00024 //
00025
00026 #ifndef SREADLINE_H
00027 #define SREADLINE_H
00028
00029 #include <stdio.h>
00030
00031 #include <readline/readline.h>
00032 #include <readline/history.h>
00033 #include <readline/keymaps.h>
00034
00035 #include <string>
00036 #include <fstream>
00037 #include <vector>
00038 #include <stdexcept>
00039 #include <map>
00040
00041 #include <boost/algorithm/string/trim.hpp>
00042 #include <boost/tokenizer.hpp>
00043 #include <boost/function.hpp>
00044
00045 namespace {
00054     typedef std::vector<std::string> TokensStorage;
00055
00059     typedef std::vector<TokensStorage> CompletionsStorage;
00060
00064     typedef boost::function<int (int, int)> KeyCallback;
00065
00069     typedef std::map<int, KeyCallback> KeysBind;
00070
00074     const size_t DefaultHistoryLimit (64);
00075
00079     CompletionsStorage Completions;
00080
00084     TokensStorage Tokens;
00085
00089     std::map<Keymap, KeysBind> Keymaps;
00090
00094     bool KeymapWasSetup (false);
00095
00099     Keymap Earlykeymap (0);
00100
00101
00108     char* Generator (const char* text, int State);
00109
00110
00118     char** UserCompletion (const char* text, int start, int end);
00119
00120
00128     int KeyDispatcher (int Count, int Key);
00129
00130
00135     int StartupHook (void);
00136
00137
00145     template <typename Container>
00146     bool AreTokensEqual (const Container& Pattern, const Container& Input) {
00147         if (Input.size() > Pattern.size()) {
00148             return false;
00149         }
00150
00151         typename Container::const_iterator k (Pattern.begin());
00152         typename Container::const_iterator j (Input.begin());
00153         for ( ; j != Input.end(); ++k, ++j) {
00154             const std::string lPattern = *k;
00155             if (lPattern == "%file") {
00156                 continue;
00157             }
00158
00159             const std::string lInput = *j;

```

```

00160         if (lPattern != lInput) {
00161             return false;
00162         }
00163     }
00164     return true;
00165 }
00166
00167 // See description near the prototype
00168 template <typename ContainerType>
00169 void SplitTokens (const std::string& Source, ContainerType& Container) {
00170     typedef boost::tokenizer<boost::char_separator<char> > TokenizerType;
00171
00172     // Set of token separators
00173     boost::char_separator<char> Separators (" \\t\\n");
00174     // Tokens provider
00175     TokenizerType Tokenizer (Source, Separators);
00176
00177     Container.clear();
00178     for (TokenizerType::const_iterator k (Tokenizer.begin());
00179          k != Tokenizer.end(); ++k) {
00179         // Temporary storage for the token, in order to trim that latter
00181         std::string SingleToken (*k);
00182
00183         boost::algorithm::trim (SingleToken);
00184         Container.push_back (SingleToken);
00185     }
00186 }
00187
00188 // See description near the prototype
00189 char** UserCompletion (const char* text, int start, int end) {
00190     // No default completion at all
00191     rl_attempted_completion_over = 1;
00192
00193     if (Completions.empty() == true) {
00194         return NULL;
00195     }
00196
00197     // Memorise all the previous tokens
00198     std::string PreInput (rl_line_buffer, start);
00199     SplitTokens (PreInput, Tokens);
00200
00201     // Detect whether we should call the standard file name completer
00202     // or a custom one
00203     bool FoundPretender (false);
00204
00205     for (CompletionsStorage::const_iterator k (Completions.begin());
00206          k != Completions.end(); ++k) {
00207         const TokensStorage& lTokenStorage = *k;
00208         if (AreTokensEqual (lTokenStorage, Tokens) == false) {
00209             continue;
00210         }
00211
00212         if (lTokenStorage.size() > Tokens.size()) {
00213             FoundPretender = true;
00214             if (lTokenStorage [Tokens.size()] == "%file") {
00215                 // Standard file name completer - called for the "%file" keyword
00216                 return rl_completion_matches (text, rl_filename_completion_function);
00217             }
00218         }
00219     }
00220
00221     if (FoundPretender) {
00222         return rl_completion_matches (text, Generator);
00223     }
00224     return NULL;
00225 }
00226
00227 // See description near the prototype
00228 char* Generator (const char* text, int State) {
00229     static int Length;
00230     static CompletionsStorage::const_iterator Iterator;
00231
00232     if ( State == 0 ) {
00233         Iterator = Completions.begin();
00234         Length = strlen (text);
00235     }
00236
00237     for ( ; Iterator != Completions.end(); ++Iterator) {
00238         const TokensStorage& lCompletion = *Iterator;
00239         if (AreTokensEqual (lCompletion, Tokens) == false) {
00240             continue;
00241         }
00242
00243         if (lCompletion.size() > Tokens.size()) {
00244             if (lCompletion [Tokens.size()] == "%file") {
00245                 continue;
00246             }

```

```

00247
00248     const char* lCompletionCharStr (lCompletion [Tokens.size()].c_str());
00249     if (strcmp (text, lCompletionCharStr, Length) == 0) {
00250         // Readline will free the allocated memory
00251         const size_t lCompletionSize = strlen (lCompletionCharStr) + 1;
00252         char* NewString (static_cast<char*> (malloc (lCompletionSize)));
00253         strcpy (NewString, lCompletionCharStr);
00254
00255         ++Iterator;
00256
00257         return NewString;
00258     }
00259 }
00260 }
00261
00262 return NULL;
00263 }
00264
00265 // See the description near the prototype
00266 int KeyDispatcher (int Count, int Key) {
00267     std::map< Keymap, KeysBind >::iterator Set (Keymaps.find (rl_get_keymap()));
00268     if (Set == Keymaps.end()) {
00269         // Most probably it happens because the header was
00270         // included into many compilation units and the
00271         // keymap setting calls were made in different files.
00272         // This is the problem of "global" data.
00273         // The storage of all the registered keymaps is in anonymous
00274         // namespace.
00275         throw std::runtime_error ("Error selecting a keymap.");
00276     }
00277 }
00278
00279 (Set->second)[Key] (Count, Key);
00280 return 0;
00281 }
00282
00283 // See the description near the prototype
00284 int StartupHook (void) {
00285     if (KeymapWasSetup) {
00286         rl_set_keymap (Earlykeymap);
00287     }
00288     return 0;
00289 }
00290
00291 } // Anonymous namespace
00292
00293 namespace swift {
00294
00295     class SKeymap {
00296     private:
00297         // Readline keymap
00298         Keymap keymap;
00299
00300     public:
00301         explicit SKeymap (bool PrintableBound = false) : keymap (NULL) {
00302             if (PrintableBound == true) {
00303                 // Printable characters are bound
00304                 keymap = rl_make_keymap();
00305             } else {
00306                 // Empty keymap
00307                 keymap = rl_make_bare_keymap();
00308             }
00309
00310             if (keymap == NULL) {
00311                 throw std::runtime_error ("Cannot allocate keymap.");
00312             }
00313
00314             // Register a new keymap in the global list
00315             Keymaps [keymap] = KeysBind();
00316         }
00317
00318         explicit SKeymap (Keymap Pattern) : keymap (rl_copy_keymap (Pattern)) {
00319             if (keymap == NULL) {
00320                 throw std::runtime_error ("Cannot allocate keymap.");
00321             }
00322
00323             // Register a new keymap in the global list
00324             Keymaps [keymap] = KeysBind();
00325         }
00326
00327         ~SKeymap() {
00328             // Deregister the keymap
00329             Keymaps.erase (keymap);
00330             rl_discard_keymap (keymap);
00331         }
00332     };

```

```

00359
00366 void Bind (int Key, KeyCallback Callback) {
00367     Keymaps [keymap][Key] = Callback;
00368
00369     if (rl_bind_key_in_map (Key, KeyDispatcher, keymap) != 0) {
00370         // Remove from the map just bound key
00371         Keymaps [keymap].erase (Key);
00372         throw std::runtime_error ("Invalid key.");
00373     }
00374 }
00375
00381 void Unbind (int Key) {
00382     rl_unbind_key_in_map (Key, keymap);
00383     Keymaps [keymap].erase (Key);
00384 }
00385
00386 // void Bind (const std::string& Sequence, boost::function<int (int, int)>);
00387 // void Unbind (std::string& Sequence);
00388
00389 public:
00395 SKeymap (const SKeymap& rhs) {
00396     if (this == &rhs) {
00397         return;
00398     }
00399     keymap = rl_copy_keymap (rhs.keymap);
00400 }
00401
00407 SKeymap& operator= (const SKeymap& rhs) {
00408     if (this == &rhs) {
00409         return *this;
00410     }
00411     keymap = rl_copy_keymap (rhs.keymap);
00412     return *this;
00413 }
00414
00415 friend class SReadline;
00416 };
00417
00424 class SReadline {
00425 public:
00431 SReadline (const size_t Limit = DefaultHistoryLimit) :
00432     HistoryLimit (Limit), HistoryFileName (""),
00433     OriginalCompletion (rl_attempted_completion_function) {
00434     rl_startup_hook = StartupHook;
00435     rl_attempted_completion_function = UserCompletion;
00436     using_history();
00437 }
00438
00446 SReadline( const std::string & historyFileName,
00447     const size_t Limit = DefaultHistoryLimit ) :
00448     HistoryLimit( Limit ),
00449     HistoryFileName( historyFileName ),
00450     OriginalCompletion( rl_attempted_completion_function )
00451 {
00452     rl_startup_hook = StartupHook;
00453     rl_attempted_completion_function = UserCompletion;
00454     using_history();
00455     LoadHistory( HistoryFileName );
00456 }
00457
00462 ~SReadline() {
00463     rl_attempted_completion_function = OriginalCompletion;
00464     SaveHistory (HistoryFileName);
00465 }
00466
00473 std::string GetLine (const std::string& Prompt) {
00474     bool Unused;
00475     return GetLine (Prompt, Unused);
00476 }
00477
00486 template <typename Container>
00487 std::string GetLine (const std::string& Prompt, Container& ReadTokens) {
00488     bool Unused;
00489     return GetLine (Prompt, ReadTokens, Unused);
00490 }
00491
00501 template <typename Container>
00502 std::string GetLine (const std::string& Prompt, Container& ReadTokens,
00503     bool& BreakOut) {
00504     std::string Input (GetLine (Prompt, BreakOut));
00505     SplitTokens (Input, ReadTokens);
00506     return Input;
00507 }
00508
00509
00517 std::string GetLine (const std::string& Prompt, bool& BreakOut) {
00518     BreakOut = true;

```

```

00519
00520     char* ReadLine (readline (Prompt.c_str()));
00521     if (ReadLine == NULL) {
00522         return std::string();
00523     }
00524
00525     // It's OK
00526     BreakOut = false;
00527     std::string Input (ReadLine);
00528     free (ReadLine); ReadLine = NULL;
00529
00530     boost::algorithm::trim (Input);
00531     if (Input.empty() == false) {
00532         if (history_length == 0
00533             || Input != history_list()[ history_length - 1 ]->line) {
00534             add_history (Input.c_str());
00535
00536             if (history_length >= static_cast<int> (HistoryLimit)) {
00537                 stifle_history (HistoryLimit);
00538             }
00539         }
00540     }
00541
00542     return Input;
00543 }
00544
00545
00551     template <typename ContainerType>
00552     void GetHistory (ContainerType& Container) {
00553         for (int k (0); k < history_length; ++k ) {
00554             Container.push_back (history_list()[k]->line);
00555         }
00556     }
00557
00564     bool SaveHistory (std::ostream& OS) {
00565         if (!OS) {
00566             return false;
00567         }
00568
00569         for (int k (0); k < history_length; ++k) {
00570             OS << history_list()[ k ]->line << std::endl;
00571         }
00572         return true;
00573     }
00574
00581     bool SaveHistory (const std::string& FileName) {
00582         if (FileName.empty() == true) {
00583             return false;
00584         }
00585
00586         std::ofstream OS (FileName.c_str());
00587         return SaveHistory (OS);
00588     }
00589
00594     void ClearHistory() {
00595         clear_history();
00596     }
00597
00604     bool LoadHistory (std::istream& IS) {
00605         if (!IS) {
00606             return false;
00607         }
00608
00609         ClearHistory();
00610         std::string OneLine;
00611
00612         while (!getline (IS, OneLine).eof()) {
00613             boost::algorithm::trim( OneLine );
00614             if ((history_length == 0)
00615                 || OneLine != history_list()[history_length - 1]->line) {
00616                 add_history (OneLine.c_str());
00617             }
00618         }
00619         stifle_history (HistoryLimit);
00620         return true;
00621     }
00622
00629     bool LoadHistory (const std::string& FileName) {
00630         if (FileName.empty() == true) {
00631             return false;
00632         }
00633
00634         std::ifstream IS (FileName.c_str());
00635         return LoadHistory (IS);
00636     }
00637
00657     template <typename ContainerType>

```



```

00658     void RegisterCompletions (const ContainerType& Container) {
00659         Completions.clear();
00660         for (typename ContainerType::const_iterator k (Container.begin());
00661             k != Container.end(); ++k) {
00662             std::vector<std::string> OneLine;
00663             const std::string& kStr = static_cast<std::string> (*k);
00664             SplitTokens (kStr, OneLine);
00665             Completions.push_back (OneLine);
00666         }
00667     }
00668 }
00669
00675 void SetKeymap (SKeymap& NewKeymap) {
00676     rl_set_keymap (NewKeymap.keymap);
00677     KeymapWasSetup = true;
00678     Earlykeymap = NewKeymap.keymap;
00679 }
00680
00681 private:
00682 // ////////////////////////////////// Attributes //////////////////////////////////
00683 const size_t HistoryLimit;
00684
00692 const std::string HistoryFileName;
00693
00697 rl_completion_func_t* OriginalCompletion;
00698 };
00699
00700 }; // namespace swift
00701
00702 #endif
00703

```

### 33.639 test/stdair/MPBomRoot.cpp File Reference

### 33.640 MPBomRoot.cpp

```

00001
00005 // //////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////
00008 // STL
00009 #include <cassert>
00010 // StdAir Test
00011 #include <test/stdair/MPBomRoot.hpp>
00012
00013 namespace myprovider {
00014
00015 // //////////////////////////////////
00016 BomRoot::BomRoot (const Key_T& iKey) : stdair::BomRoot (iKey) {
00017 }
00018
00019 // //////////////////////////////////
00020 BomRoot::~BomRoot () {
00021 }
00022
00023 }

```

### 33.641 test/stdair/MPBomRoot.hpp File Reference

### 33.642 MPBomRoot.hpp

```

00001 #ifndef __MYPROVIDER_BOMROOT_HPP
00002 #define __MYPROVIDER_BOMROOT_HPP
00003
00008 // //////////////////////////////////
00009 // Import section
00010 // //////////////////////////////////
00011 // STL
00012 #include <string>
00013 // StdAir
00014 #include <stdair/bom/BomRoot.hpp>
00015
00016 namespace myprovider {
00017
00020 class BomRoot : public stdair::BomRoot {
00021 public:
00022 // ////////////////////////////////// Display support methods //////////////////////////////////
00024     std::string toString() const { return describeKey(); }

```

```

00025
00028     const std::string describeKey() const { return std::string (""); }
00029
00030 public:
00034     BomRoot (const Key_T&);
00036     ~BomRoot ();
00038     BomRoot ();
00039     BomRoot (const BomRoot&);
00040 };
00041
00042 }
00046 #endif // __MYPROVIDER_BOMROOT_HPP

```

### 33.643 test/stdair/MPInventory.cpp File Reference

#### 33.644 MPInventory.cpp

```

00001
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <cassert>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 // StdAir Test
00013 #include <test/stdair/MPInventory.hpp>
00014
00015 namespace myprovider {
00016
00017 // //////////////////////////////////////
00018 Inventory::Inventory (const Key_T& iKey) : stdair::Inventory (iKey) {
00019 }
00020
00021 // //////////////////////////////////////
00022 Inventory::~Inventory () {
00023 }
00024
00025 // //////////////////////////////////////
00026 std::string Inventory::toString() const {
00027     std::ostringstream ostr;
00028     ostr << _key.toString();
00029     return ostr.str();
00030 }
00031
00032 // //////////////////////////////////////
00033 const std::string Inventory::describeKey() const {
00034     return _key.toString();
00035 }
00036
00037 }

```

### 33.645 test/stdair/MPInventory.hpp File Reference

#### 33.646 MPInventory.hpp

```

00001 #ifndef __MYPROVIDER_INVENTORY_HPP
00002 #define __MYPROVIDER_INVENTORY_HPP
00003
00008 // //////////////////////////////////////
00009 // Import section
00010 // //////////////////////////////////////
00011 // STL
00012 #include <list>
00013 // StdAir
00014 #include <stdair/bom/Inventory.hpp>
00015
00016 namespace myprovider {
00017
00018 class Inventory : public stdair::Inventory {
00019 public:
00020     // ////////////////////////////////// Display support methods //////////////////////////////////
00022     std::string toString() const;
00023
00026     const std::string describeKey() const;
00027
00028 public:
00032     Inventory (const Key_T&);
00034     ~Inventory();

```

```

00036     Inventory ();
00037     Inventory (const Inventory&);
00038 };
00039
00040 // //////////// Type definitions ////////////
00042 typedef std::list<Inventory*> InventoryList_T;
00043
00044 }
00048 #endif // __MYPROVIDER_INVENTORY_HPP

```

### 33.647 test/stdair/StandardAirlineITTestSuite.cpp File Reference

### 33.648 StandardAirlineITTestSuite.cpp

```

00001
00005 // ////////////////////////////////////////////
00006 // Import section
00007 // ////////////////////////////////////////////
00008 // STL
00009 #include <sstream>
00010 #include <fstream>
00011 #include <string>
00012 // Boost MPL
00013 #include <boost/mpl/push_back.hpp>
00014 #include <boost/mpl/vector.hpp>
00015 #include <boost/mpl/at.hpp>
00016 #include <boost/mpl/assert.hpp>
00017 #include <boost/type_traits/is_same.hpp>
00018 // Boost Unit Test Framework (UTF)
00019 #define BOOST_TEST_DYN_LINK
00020 #define BOOST_TEST_MAIN
00021 #define BOOST_TEST_MODULE StdAirTest
00022 #if BOOST_VERSION >= 103900
00023 #include <boost/test/unit_test.hpp>
00024 #else // BOOST_VERSION >= 103900
00025 #include <boost/test/test_tools.hpp>
00026 #include <boost/test/results_reporter.hpp>
00027 #include <boost/test/unit_test_suite.hpp>
00028 #include <boost/test/output_test_stream.hpp>
00029 #include <boost/test/unit_test_log.hpp>
00030 #include <boost/test/framework.hpp>
00031 #include <boost/test/detail/unit_test_parameters.hpp>
00032 #endif // BOOST_VERSION >= 103900
00033 // Boost Serialisation
00034 #include <boost/archive/text_oarchive.hpp>
00035 #include <boost/archive/text_iarchive.hpp>
00036 // StdAir
00037 #include <stdair/stdair_inventory_types.hpp>
00038 #include <stdair/service/Logger.hpp>
00039 #include <stdair/STDAIR_Service.hpp>
00040 #include <stdair/basic/float_utils.hpp>
00041 #include <stdair/bom/BomDisplay.hpp>
00042 #include <stdair/bom/BomRoot.hpp>
00043 #include <stdair/bom/BomManager.hpp>
00044 #include <stdair/factory/FacBom.hpp>
00045 #include <stdair/factory/FacBomManager.hpp>
00046 // StdAir Test Suite
00047 #include <test/stdair/StdairTestLib.hpp>
00048 #include <test/stdair/MPInventory.hpp>
00049
00050 namespace boost_utf = boost::unit_test;
00051
00052 #if BOOST_VERSION >= 103900
00053
00054 // (Boost) Unit Test XML Report
00055 std::ofstream utfReportStream ("StandardAirlineITTestSuite_utfresults.xml");
00056
00060 struct UnitTestConfig {
00062     UnitTestConfig() {
00063         boost_utf::unit_test_log.set_stream (utfReportStream);
00064         boost_utf::unit_test_log.set_format (boost_utf::XML);
00065         boost_utf::unit_test_log.set_threshold_level (boost_utf::log_test_units);
00066         // boost_utf::unit_test_log.set_threshold_level (boost_utf::log_successful_tests);
00067     }
00068
00070     ~UnitTestConfig() {
00071     }
00072 };
00073
00074
00075 // ////////////////////////////////// Main: Unit Test Suite //////////////////////////////////
00076
00077 // Set the UTF configuration (re-direct the output to a specific file)

```

```

00078 BOOST_GLOBAL_FIXTURE (UnitTestFixture);
00079
00080 // Start the test suite
00081 BOOST_AUTO_TEST_SUITE (master_test_suite)
00082
00083
00087 BOOST_AUTO_TEST_CASE (float_comparison_test) {
00088     float a = 0.2f;
00089     a = 5*a;
00090     const float b = 1.0f;
00091
00092     // Test the Boost way
00093     BOOST_CHECK_MESSAGE (a == b, "The two floats (" << a << " and " << b
00094                         << ") should be equal, but are not");
00095     BOOST_CHECK_CLOSE (a, b, 0.0001);
00096
00097     // Test the Google way
00098     const FloatingPoint<float> lhs (a), rhs (b);
00099     BOOST_CHECK_MESSAGE (lhs.AlmostEquals (rhs),
00100                         "The two floats (" << a << " and " << b
00101                         << ") should be equal, but are not");
00102 }
00103
00108 BOOST_AUTO_TEST_CASE (mpl_structure_test) {
00109     const stdair::ClassCode_T lBookingClassCodeA ("A");
00110     const stdair_test::BookingClass lA (lBookingClassCodeA);
00111     const stdair_test::Cabin lCabin (lA);
00112
00113     BOOST_CHECK_EQUAL (lCabin.toString(), lBookingClassCodeA);
00114     BOOST_CHECK_MESSAGE (lCabin.toString() == lBookingClassCodeA,
00115                         "The cabin key, '" << lCabin.toString()
00116                         << "' is not equal to '" << lBookingClassCodeA << "'");
00117
00118     // MPL
00119     typedef boost::mpl::vector<stdair_test::BookingClass> MPL_BookingClass;
00120     typedef boost::mpl::push_back<MPL_BookingClass,
00121                                 stdair_test::Cabin>::type types;
00122
00123     if (boost::is_same<stdair_test::BookingClass,
00124                 stdair_test::Cabin::child>::value == false) {
00125         BOOST_ERROR ("The two types must be equal, but are not");
00126     }
00127
00128     if (boost::is_same<boost::mpl::at_c<types, 1>::type,
00129                 stdair_test::Cabin>::value == false) {
00130         BOOST_ERROR ("The type must be stdair_test::Cabin, but is not");
00131     }
00132 }
00133
00137 BOOST_AUTO_TEST_CASE (stdair_service_initialisation_test) {
00138     // Output log File
00139     const std::string lLogFilename ("StandardAirlineITTestSuite_init.log");
00140
00141     // Set the log parameters
00142     std::ofstream logOutputFile;
00143
00144     // Open and clean the log outputfile
00145     logOutputFile.open (lLogFilename.c_str());
00146     logOutputFile.clear();
00147
00148     // Initialise the stdair BOM
00149     const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG,
00150     logOutputFile);
00151     stdair::STDAIR_Service stdairService (lLogParams);
00152
00152     // Retrieve (a reference on) the top of the persistent BOM tree
00153     stdair::BomRoot& lPersistentBomRoot = stdairService.getPersistentBomRoot();
00154
00155     // Retrieve the BomRoot key, and compare it to the expected one
00156     const std::string& lBomRootKeyStr = lPersistentBomRoot.describeKey();
00157     const std::string lBomRootString (" -- ROOT -- ");
00158
00159     // DEBUG
00160     STDAIR_LOG_DEBUG ("The BOM root key is '" << lBomRootKeyStr
00161                     << "'. It should be equal to '" << lBomRootString << "'");
00162
00163     BOOST_CHECK_EQUAL (lBomRootKeyStr, lBomRootString);
00164     BOOST_CHECK_MESSAGE (lBomRootKeyStr == lBomRootString,
00165                         "The BOM root key, '" << lBomRootKeyStr
00166                         << "', should be equal to '" << lBomRootString
00167                         << "', but is not.");
00168
00169     // Build a sample BOM tree
00170     stdairService.buildSampleBom();
00171
00172     // DEBUG: Display the whole BOM tree
00173     const std::string& lCSVDump = stdairService.csvDisplay ();

```

```

00174     STDAIR_LOG_DEBUG (lCSVDump);
00175
00176     // Close the Log outputFile
00177     logOutputFile.close();
00178 }
00179
00183 BOOST_AUTO_TEST_CASE (bom_structure_instantiation_test) {
00184     // Step 0.0: initialisation
00185     // Create the root of a Bom tree (i.e., a BomRoot object)
00186     stdair::BomRoot& lBomRoot =
00187         stdair::FacBom<stdair::BomRoot>::instance().
00188         create();
00189
00189     // Step 0.1: Inventory level
00190     // Create an Inventory (BA)
00191     const stdair::AirlineCode_T lBAAirlineCode ("BA");
00192     const stdair::InventoryKey lBAKey (lBAAirlineCode);
00193     myprovider::Inventory& lBAInv =
00194         stdair::FacBom<myprovider::Inventory>::instance().
00195         create (lBAKey);
00196     stdair::FacBomManager::addToList (lBomRoot, lBAInv);
00197
00197     BOOST_CHECK_EQUAL (lBAInv.describeKey(), lBAAirlineCode);
00198     BOOST_CHECK_MESSAGE (lBAInv.describeKey() == lBAAirlineCode,
00199         "The inventory key, '" << lBAInv.describeKey()
00200         << "', should be equal to '" << lBAAirlineCode
00201         << "', but is not");
00202
00203     // Create an Inventory for AF
00204     const stdair::AirlineCode_T lFAAirlineCode ("AF");
00205     const stdair::InventoryKey lAFKey (lFAAirlineCode);
00206     myprovider::Inventory& lAFInv =
00207         stdair::FacBom<myprovider::Inventory>::instance().
00208         create (lAFKey);
00209     stdair::FacBomManager::addToList (lBomRoot, lAFInv);
00210
00210     BOOST_CHECK_EQUAL (lAFInv.describeKey(), lFAAirlineCode);
00211     BOOST_CHECK_MESSAGE (lAFInv.describeKey() == lFAAirlineCode,
00212         "The inventory key, '" << lAFInv.describeKey()
00213         << "', should be equal to '" << lFAAirlineCode
00214         << "', but is not");
00215
00216     // Browse the inventories
00217     const myprovider::InventoryList_T& lInventoryList =
00218         stdair::BomManager::getList<myprovider::Inventory> (lBomRoot);
00219     const std::string lInventoryKeyArray[2] = {lBAAirlineCode, lFAAirlineCode};
00220     short idx = 0;
00221     for (myprovider::InventoryList_T::const_iterator itInv =
00222         lInventoryList.begin(); itInv != lInventoryList.end();
00223         ++itInv, ++idx) {
00224         const myprovider::Inventory* lInv_ptr = *itInv;
00225         BOOST_REQUIRE (lInv_ptr != NULL);
00226
00227         BOOST_CHECK_EQUAL (lInventoryKeyArray[idx], lInv_ptr->describeKey());
00228         BOOST_CHECK_MESSAGE (lInventoryKeyArray[idx] == lInv_ptr->describeKey(),
00229             "They inventory key, '" << lInventoryKeyArray[idx]
00230             << "', does not match that of the Inventory object: '"
00231             << lInv_ptr->describeKey() << "'");
00232     }
00233 }
00234
00238 BOOST_AUTO_TEST_CASE (bom_structure_serialisation_test) {
00239
00240     // Backup (thanks to Boost.Serialisation) file
00241     const std::string lBackupFilename = "StandardAirlineITTestSuite_serial.txt";
00242
00243     // Output log File
00244     const std::string lLogFilename ("StandardAirlineITTestSuite_serial.log");
00245
00246     // Set the log parameters
00247     std::ofstream logOutputFile;
00248
00249     // Open and clean the log outputfile
00250     logOutputFile.open (lLogFilename.c_str());
00251     logOutputFile.clear();
00252
00253     // Initialise the stdair BOM
00254     const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG,
00255         logOutputFile);
00256     stdair::STDAIR_Service stdairService (lLogParams);
00257
00257     // Build a sample BOM tree
00258     stdairService.buildSampleBom();
00259
00259     // Retrieve (a reference on) the top of the persistent BOM tree
00260     stdair::BomRoot& lPersistentBomRoot = stdairService.getPersistentBomRoot();
00261
00262

```

```

00263 // DEBUG: Display the whole BOM tree
00264 const std::string& lCSVDump = stdairService.csvDisplay ();
00265 STDAIR_LOG_DEBUG (lCSVDump);
00266
00267 // Clone the persistent BOM
00268 stdairService.clonePersistentBom ();
00269
00270 // Retrieve the BomRoot key, and compare it to the expected one
00271 const std::string lBAInvKeyStr ("BA");
00272 stdair::Inventory* lBAInv_ptr =
00273     lPersistentBomRoot.getInventory (lBAInvKeyStr);
00274
00275 // DEBUG
00276 STDAIR_LOG_DEBUG ("There should be an Inventory object corresponding to the '"
00277     << lBAInvKeyStr << "' key.");
00278
00279 BOOST_REQUIRE_MESSAGE (lBAInv_ptr != NULL,
00280     "An Inventory object should exist with the key, '"
00281     << lBAInvKeyStr << "'.");
00282
00283 // create and open a character archive for output
00284 std::ofstream ofs (lBackupFilename.c_str());
00285
00286 // save data to archive
00287 {
00288     boost::archive::text_oarchive oa (ofs);
00289     // write class instance to archive
00290     oa << lPersistentBomRoot;
00291     // archive and stream closed when destructors are called
00292 }
00293
00294 // ... some time later restore the class instance to its original state
00295 stdair::BomRoot& lRestoredBomRoot =
00296     stdair::FacBom<stdair::BomRoot>::instance().
00297     create();
00298 {
00299     // create and open an archive for input
00300     std::ifstream ifs (lBackupFilename.c_str());
00301     boost::archive::text_iarchive ia(ifs);
00302     // read class state from archive
00303     ia >> lRestoredBomRoot;
00304     // archive and stream closed when destructors are called
00305 }
00306
00307 // DEBUG: Display the whole restored BOM tree
00308 const std::string& lRestoredCSVDump =
00309     stdairService.csvDisplay(lRestoredBomRoot);
00310 STDAIR_LOG_DEBUG (lRestoredCSVDump);
00311
00312 // Retrieve the BomRoot key, and compare it to the expected one
00313 const std::string& lBomRootKeyStr = lRestoredBomRoot.describeKey();
00314 const std::string lBomRootString (" -- ROOT -- ");
00315
00316 // DEBUG
00317 STDAIR_LOG_DEBUG ("The BOM root key is '" << lBomRootKeyStr
00318     << "'. It should be equal to '" << lBomRootString << "'");
00319
00320 BOOST_CHECK_EQUAL (lBomRootKeyStr, lBomRootString);
00321 BOOST_CHECK_MESSAGE (lBomRootKeyStr == lBomRootString,
00322     "The BOM root key, '" << lBomRootKeyStr
00323     << "', should be equal to '" << lBomRootString
00324     << "', but is not.");
00325
00326 // Retrieve the Inventory
00327 stdair::Inventory* lRestoredBAInv_ptr =
00328     lRestoredBomRoot.getInventory (lBAInvKeyStr);
00329
00330 // DEBUG
00331 STDAIR_LOG_DEBUG ("There should be an Inventory object corresponding to the '"
00332     << lBAInvKeyStr << "' key in the restored BOM root.");
00333
00334 BOOST_CHECK_MESSAGE (lRestoredBAInv_ptr != NULL,
00335     "An Inventory object should exist with the key, '"
00336     << lBAInvKeyStr << "' in the restored BOM root.");
00337
00338 // Close the Log output file
00339 logOutputFile.close();
00340 }
00341
00342 BOOST_AUTO_TEST_CASE (bom_structure_clone_test) {
00343     // Output log file
00344     const std::string lLogFilename ("StandardAirlineITTestSuite_clone.log");
00345
00346     // Set the log parameters
00347     std::ofstream logOutputFile;
00348 }

```

```

00352 // Open and clean the log outputfile
00353 logOutputFile.open (lLogFilename.c_str());
00354 logOutputFile.clear();
00355
00356 // Initialise the stdair BOM
00357 const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG,
logOutputFile);
00358 stdair::STDAIR_Service stdairService (lLogParams);
00359
00360 // Build a sample BOM tree
00361 stdairService.buildSampleBom();
00362
00363 // Retrieve (a constant reference on) the top of the persistent BOM tree
00364 const stdair::BomRoot& lPersistentBomRoot =
00365     stdairService.getPersistentBomRoot();
00366
00367 // DEBUG: Display the whole persistent BOM tree
00368 const std::string& lCSVDump = stdairService.csvDisplay ();
00369 STDAIR_LOG_DEBUG ("Display the persistent BOM tree.");
00370 STDAIR_LOG_DEBUG (lCSVDump);
00371
00372 // Clone the persistent BOM
00373 stdairService.clonePersistentBom ();
00374
00375 // Retrieve (a reference on) the top of the clone BOM tree
00376 stdair::BomRoot& lCloneBomRoot = stdairService.getBomRoot();
00377
00378 // DEBUG: Display the clone BOM tree after the clone process.
00379 const std::string& lAfterCloneCSVDump =
00380     stdairService.csvDisplay(lCloneBomRoot);
00381 STDAIR_LOG_DEBUG ("Display the clone BOM tree after the clone process.");
00382 STDAIR_LOG_DEBUG (lAfterCloneCSVDump);
00383
00384 // Retrieve the clone BomRoot key, and compare it to the persistent BomRoot
00385 // key.
00386 const std::string& lCloneBomRootKeyStr = lCloneBomRoot.describeKey();
00387 const std::string& lPersistentBomRootKeyStr =
00388     lPersistentBomRoot.describeKey();
00389
00390 // DEBUG
00391 STDAIR_LOG_DEBUG ("The clone BOM root key is '" << lCloneBomRootKeyStr
00392 << "'. It should be equal to '"
00393 << lPersistentBomRootKeyStr << "'");
00394
00395 BOOST_CHECK_EQUAL (lCloneBomRootKeyStr, lPersistentBomRootKeyStr);
00396 BOOST_CHECK_MESSAGE (lCloneBomRootKeyStr == lPersistentBomRootKeyStr,
00397     "The clone BOM root key, '" << lCloneBomRootKeyStr
00398 << "', should be equal to '" << lPersistentBomRootKeyStr
00399 << "', but is not.");
00400
00401 // Retrieve the BA inventory in the clone BOM root
00402 const std::string lBAInvKeyStr ("BA");
00403 stdair::Inventory* lCloneBAInv_ptr =
00404     lCloneBomRoot.getInventory (lBAInvKeyStr);
00405
00406 // DEBUG
00407 STDAIR_LOG_DEBUG ("There should be an Inventory object corresponding to the '"
00408 << lBAInvKeyStr << "' key in the clone BOM root.");
00409
00410 BOOST_CHECK_MESSAGE (lCloneBAInv_ptr != NULL,
00411     "An Inventory object should exist with the key, '"
00412 << lBAInvKeyStr << "' in the clone BOM root.");
00413
00414 // Close the Log outputFile
00415 logOutputFile.close();
00416 }
00417
00418 // End the test suite
00419 BOOST_AUTO_TEST_SUITE_END()
00420
00421 #else // BOOST_VERSION >= 103900
00422 boost_utf::test_suite* init_unit_test_suite (int, char* []) {
00423     boost_utf::test_suite* test = BOOST_TEST_SUITE ("Unit test example 1");
00424     return test;
00425 }
00426 #endif // BOOST_VERSION >= 103900
00427

```

### 33.649 test/stdair/StdairTestLib.hpp File Reference

```

#include <string>
#include <sstream>

```

## Classes

- struct [stdair\\_test::BookingClass](#)
- struct [stdair\\_test::Cabin](#)

## Namespaces

- [stdair\\_test](#)

## 33.650 StdairTestLib.hpp

```

00001 #ifndef __STDAIR_TST_STDAIR_TEST_LIB_HPP
00002 #define __STDAIR_TST_STDAIR_TEST_LIB_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 #include <string>
00008 #include <sstream>
00009
00013 namespace stdair_test {
00014
00016     struct BookingClass {
00017         std::string _classCode;
00019         BookingClass (const std::string& iClassCode)
00020             : _classCode (iClassCode) {
00021         }
00022
00024         std::string toString() const {
00025             std::ostringstream oStr;
00026             oStr << _classCode;
00027             return oStr.str();
00028         }
00029     };
00030
00032     struct Cabin {
00033         BookingClass _bookingClass;
00034         Cabin (const BookingClass& iBkgClass)
00035             : _bookingClass (iBkgClass) {
00036         }
00037
00039         std::string toString() const {
00040             std::ostringstream oStr;
00041             oStr << _bookingClass._classCode;
00042             return oStr.str();
00043         }
00044
00046         typedef BookingClass child;
00047     };
00048 }
00049
00050 #endif // __STDAIR_TST_STDAIR_TEST_LIB_HPP

```



