AirInv

1.00.0

Generated by Doxygen 1.8.1.1

Tue Feb 12 2013 13:03:00

Contents

1 AirInv Documentation

1.1 Getting Started

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

1.2 AirInv at SourceForge

- Project page

- Download AirInv
- Open a ticket for a bug or feature
- Mailing lists
- Forums
  - Discuss about Development issues
  - Ask for Help
  - Discuss AirInv

1.3 AirInv Development

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

1.4 External Libraries

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

1.5 Support AirInv

1.6 About AirInv

AirInv is a C++ library of airline inventory management classes and functions, mainly targeting simulation purposes. N

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

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

AirInv 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 AirInv 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 AirInv.

2 People

2.1 Project Admins

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

2.2 Developers

- Anh Quan Nguyen quannaus@users.sourceforge.net (N)
- Denis Arnaud denis_arnaud@users.sourceforge.net (N)
- Son Nguyen Kim snguyenkim@users.sourceforge.net (N)
- Nicolas Bondoux nbondoux@users.sourceforge.net (N)

2.3 Retired Developers

- Patrick Grandjean pgrandjean@users.sourceforge.net (N)
- Ngoc-Thach Hoang hoangngocthach@users.sourceforge.net (N)

2.4 Contributors

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

2.5 Distribution Maintainers

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

Note

(N) - Amadeus employees.

3 Coding Rules

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

3.1 Default Naming Rules for Variables

Variables names follow Java naming conventions. Examples:

- `lNumberOfPassengers`
- `lSeatAvailability`

3.2 Default Naming Rules for Functions

Function names follow Java naming conventions. Example:

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

3.3 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`

3.4 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`

3.5 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.

4 Copyright and License

4.1 GNU LESSER GENERAL PUBLIC LICENSE

4.1.1 Version 2.1, February 1999

4.2 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.

4.3 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.
   ```

1. 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.

1. 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.

1. 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.

1. 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.

1. 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.

4.3.1 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.

1. 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.

4.3.2 END OF TERMS AND CONDITIONS

4.4 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

5 Documentation Rules

5.1 General Rules

All classes in AirInv 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 AirInv 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
};
```

5.2 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.
 *
 * ----------------------------------------------------------------------
 *
 * AirInv - C++ Airline Inventory Management Library
 *
 * Copyright (C) 2009-2010  (\see authors file for a list of contributors)
 *
 * \see copyright file for license information
 *
```

```
 * ------------------------------------------------------------------------
 */
```

5.3 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);
```

6 Main features

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

6.1 Network generation

- Network/graph generation

6.2 Inventory generation

- Inventory generation

6.3 Finding travel solutions

- Matching of travel solutions with user requests

6.4 Distributed inventories

- Inventory independent partitions
- MPI-based distribution

6.5 Other features

- CSV input file parsing
- Memory handling

7 Make a Difference

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

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

- Start using AirSched
- Tell your friends about AirSched and help them to get started using it
- If you find a bug, report it to us. Without your help we can never hope to produce a bug free code.
- Help us to improve the documentation by providing information about documentation bugs
- Answer support requests in the AirSched discussion forums on SourceForge. If you know the answer to a question, help others to overcome their AirSched problems.
- Help us to improve our algorithms. If you know of a better way (e.g. that is faster or requires less memory) to implement some of our algorithms, then let us know.
- Help us to port AirSched to new platforms. If you manage to compile AirSched on a new platform, then tell us how you did it.
- Send us your code. If you have a good AirSched compatible code, which you can release under the LGP-Lv2.1, and you think it should be included in AirSched, then send it to us.
- Become an AirSched developer. Send us an e-mail and tell what you can do for AirSched.

8 Make a new release

8.1 Introduction

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

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

8.2 Initialisation

Clone locally the full `Git project`:

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

8.3 Branch creation

Create the branch, on your local clone, corresponding to the new release (say, 0.5.0):

```
cd ~/dev/sim/airinvgit
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 airinv.spec
```

8.4 Commit and publish the release branch

Commit the new release:

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

8.5 Update the change-log in the trunk as well

Update the change-log in the ChangeLog and RPM specification files:

```
cd ~/dev/sim/airinvgit
git checkout trunk
vi ChangeLog
vi airinv.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
```

8.6 Create distribution packages

Create the distribution packages using the following command:

```
cd ~/dev/sim/airinvgit
git checkout 0.5.0
rm -rf build && mkdir -p build
cd build
cmake -DCMAKE_INSTALL_PREFIX=/home/user/dev/deliveries/airinv-0.5.0 \
 -DWITH_STDAIR_PREFIX=/home/user/dev/deliveries/stdair-stable \
 -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, `airinv-0.5.0.tar.gz` and `airinv-0.5.0.tar.bz2`.

8.7 Generation the RPM packages

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

```
cd ~/dev/sim/airinvgit
git checkout 0.5.0
rm -rf build && mkdir -p build
cd build
cmake -DCMAKE_INSTALL_PREFIX=/home/user/dev/deliveries/airinv-0.5.0 \
 -DWITH_STDAIR_PREFIX=/home/user/dev/deliveries/stdair-stable \
 -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 airinv.spec ~/dev/packages/SPECS \
  && cp airinv-0.5.0.tar.bz2 ~/dev/packages/SOURCES
cd ~/dev/packages/SPECS
rpmbuild -ba airinv.spec
rpmlint -i ../SPECS/airinv.spec ../SRPMS/airinv-0.5.0-1.fc15.src.rpm \
  ../RPMS/noarch/airinv-* ../RPMS/i686/airinv-*
```

8.8 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 AirInv's Git repository.

8.9 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, airinv-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.

8.10 Upload the files to SourceForge

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

8.11 Upload the documentation to SourceForge

In order to update the Web site files, either:

- synchronise them with rsync and SSH:

  ```
  cd ~/dev/sim/airinvgit
  git checkout 0.5.0
  rsync -aiv doc/html/ doc/latex/refman.pdf joe,airinv@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.

8.12 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.

8.13 Send an email on the announcement mailing-list

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

9 Installation

9.1 Table of Contents

9.2 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 airinv-devel airinv-doc
```

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

9.3 Airinv Requirements

Airinv should compile without errors or warnings on most GNU/Linux systems, on UNIX systems like Solaris SunOS, and on POSIX based environments for Microsoft Windows like Cygwin or MinGW with MSYS. It can be also built on Microsoft Windows NT/2000/XP/Vista/7 using Microsoft's Visual C++ .NET, but our support for this compiler is limited. For GNU/Linux, SunOS, Cygwin and MinGW we assume that you have at least the following GNU software installed on your computer:

- GNU Autotools:

  - autoconf,
  - automake,
  - libtool,
  - make, version 3.72.1 or later (check version with 'make –version')

- GCC - GNU C++ Compiler (g++), version 4.3.x or later (check version with 'gcc –version')
- Boost - C++ STL extensions, version 1.35 or later (check version with 'grep "define BOOST_LIB_VERSION" /usr/include/boost/version.hpp')
- MySQL - Database client libraries, version 5.0 or later (check version with 'mysql –version')
- SOCI - C++ database client library wrapper, version 3.0.0 or later (check version with 'soci-config –version')

Optionally, you might need a few additional programs: Doxygen, LaTeX, Dvips and Ghostscript, to generate the HTML and PDF documentation.

We strongly recommend that you use recent stable releases of the GCC, if possible. We do not actively work on supporting older versions of the GCC, and they may therefore (without prior notice) become unsupported in future releases of Airinv.

9.4 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 a file '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 <tt>'CMakeLists.txt'</tt> is used to create the \c '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.

9.5 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.

9.6 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.

9.7 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=P-
REFIX', 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=P-
REFIX' 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.

9.8 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=P-
REFIX' 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’.

9.9 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
```

9.10 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’.

9.11 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.

9.12 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
```

9.13 'cmake' Invocation

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

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

See also

Installation Names for more details, including other options available for fine-tuning the installation locations.

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

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

The 'cmake' script produces an ouput like this:

```
export LIBSUFFIX_4_CMAKE="-DLIB_SUFFIX=64"
export INSTALL_BASEDIR=/home/user/dev/deliveries
cmake -DCMAKE_INSTALL_PREFIX=${INSTALL_BASEDIR}/airinv-0.5.0 \
 -DWITH_STDAIR_PREFIX=${INSTALL_BASEDIR}/stdair-stable \
 -DWITH_AIRRAC_PREFIX=${INSTALL_BASEDIR}/airrac-stable \
 -DWITH_RMOL_PREFIX=${INSTALL_BASEDIR}/rmol-stable \
 -DCMAKE_BUILD_TYPE:STRING=Debug -DINSTALL_DOC:BOOL=ON ${LIBSUFFIX_4_CMAKE} ..
-- 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: 0ee8dcc3e3dd1d1d442c4054fbfa4cacc1182e6a trunk
-- Requires Boost-1.41
-- Boost version: 1.46.0
-- Found the following Boost libraries:
--   regex
--   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 Readline without specifying any version
-- Found Readline: /usr/include
-- Found Readline version: 6.2
-- Requires MySQL without specifying any version
-- Using mysql-config: /usr/bin/mysql_config
-- Found MySQL: /usr/lib64/mysql/libmysqlclient.so
-- Found MySQL version: 5.5.14
-- Requires SOCI-3.0
-- Using soci-config: /usr/bin/soci-config
-- SOCI headers are buried
-- Found SOCI: /usr/lib64/libsoci_core.so (Required is at least version "3.0")
-- Found SOCIMySQL: /usr/lib64/libsoci_mysql.so (Required is at least version "3.0")
-- Found SOCI with MySQL back-end support version: 3.0.0
-- Requires StdAir-0.37
-- Found StdAir version: 0.38.0
-- Requires Doxygen without specifying any version
-- Found Doxygen: /usr/bin/doxygen
-- Found DoxygenWrapper: /usr/bin/doxygen
-- Found Doxygen version: 1.7.4
-- Had to set the linker language for 'airraclib' to CXX
-- Had to set the linker language for 'rmollib' to CXX
-- Had to set the linker language for 'airinvlib' to CXX
-- Test 'InventoryTestSuite' to be built with 'InventoryTestSuite.cpp'
--
-- ===========================================================
-- -----------------------------------
-- ---     Project Information    ---
```

```
-- ---------------------------------
-- PROJECT_NAME ................... : airinv
-- PACKAGE_PRETTY_NAME ............ : AirInv
-- PACKAGE ........................ : airinv
-- PACKAGE_NAME ................... : AIRINV
-- PACKAGE_BRIEF .................. : C++ Simulated Airline Inventory Management System library
-- PACKAGE_VERSION ................ : 0.5.0
-- GENERIC_LIB_VERSION ............ : 0.5.0
-- GENERIC_LIB_SOVERSION .......... : 0.5
--
-- ---------------------------------
-- ---     Build Configuration    ---
-- ---------------------------------
-- Modules to build ............... : airrac;rmol;airinv
-- Libraries to build/install ..... : airraclib;rmollib;airinvlib
-- Binaries to build/install ...... : airrac;rmol;airinv_parseInventory;airinv
-- Modules to test ................ : airinv
-- Binaries to test ............... : InventoryTestSuitetst
--
-- * Module ....................... : airrac
--   + Layers to build ............ : .;basic;bom;factory;command;service
--   + Dependencies on other layers :
--   + Libraries to build/install . : airraclib
--   + Executables to build/install : airrac
--   + Tests to perform ........... :
-- * Module ....................... : rmol
--   + Layers to build ............ : .;basic;bom;factory;command;service
--   + Dependencies on other layers : airraclib
--   + Libraries to build/install . : rmollib
--   + Executables to build/install : rmol
--   + Tests to perform ........... :
-- * Module ....................... : airinv
--   + Layers to build ............ : .;basic;bom;factory;command;service
--   + Dependencies on other layers : airraclib;rmollib
--   + Libraries to build/install . : airinvlib
--   + Executables to build/install : airinv_parseInventory;airinv
--   + Tests to perform ........... : InventoryTestSuitetst
--
-- BUILD_SHARED_LIBS .............. : ON
-- CMAKE_BUILD_TYPE ............... : Debug
--  * CMAKE_C_FLAGS ............... :
--  * CMAKE_CXX_FLAGS ............. : -Wall -Werror
--  * BUILD_FLAGS ................. :
--  * COMPILE_FLAGS ............... :
-- CMAKE_MODULE_PATH .............. : /home/dan/dev/sim/airinv/airinvgithub/config/
-- CMAKE_INSTALL_PREFIX ........... : /home/dan/dev/deliveries/airinv-0.5.0
--
-- * 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/dan/dev/deliveries/airinv-0.5.0/lib64
-- INSTALL_BIN_DIR ................ : /home/dan/dev/deliveries/airinv-0.5.0/bin
-- INSTALL_INCLUDE_DIR ............ : /home/dan/dev/deliveries/airinv-0.5.0/include
-- INSTALL_DATA_DIR ............... : /home/dan/dev/deliveries/airinv-0.5.0/share
-- INSTALL_SAMPLE_DIR ............. : /home/dan/dev/deliveries/airinv-0.5.0/share/airinv/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 .......... : 0.5.0
-- CPACK_PACKAGE_DESCRIPTION_FILE . : /home/dan/dev/sim/airinv/airinvgithub/README
-- CPACK_RESOURCE_FILE_LICENSE .... : /home/dan/dev/sim/airinv/airinvgithub/COPYING
-- CPACK_GENERATOR ................ : TBZ2
-- CPACK_DEBIAN_PACKAGE_DEPENDS ... :
```

```
-- CPACK_SOURCE_GENERATOR ......... : TBZ2;TGZ
-- CPACK_SOURCE_PACKAGE_FILE_NAME . : airinv-0.5.0
--
-- ---------------------------------
-- ---     External libraries    ---
-- ---------------------------------
--
-- * Boost:
--   - Boost_VERSION ............... : 104600
--   - Boost_LIB_VERSION ........... : 1_46
--   - Boost_HUMAN_VERSION ......... : 1.46.0
--   - Boost_INCLUDE_DIRS .......... : /usr/include
--   - Boost required components .. : regex;program_options;date_time;iostreams;serialization;filesystem;unit_
--   - Boost required libraries ... : optimized;/usr/lib64/libboost_regex-mt.so;debug;/usr/lib64/libboost_rege
--
-- * Readline:
--   - READLINE_VERSION ............ : 6.2
--   - READLINE_INCLUDE_DIR ........ : /usr/include
--   - READLINE_LIBRARY ............ : /usr/lib64/libreadline.so
--
-- * MySQL:
--   - MYSQL_VERSION ............... : 5.5.14
--   - MYSQL_INCLUDE_DIR ........... : /usr/include/mysql
--   - MYSQL_LIBRARIES ............. : /usr/lib64/mysql/libmysqlclient.so
--
-- * SOCI:
--   - SOCI_VERSION ................ : 3.0.0
--   - SOCI_INCLUDE_DIR ............ : /usr/include/soci
--   - SOCIMYSQL_INCLUDE_DIR ....... : /usr/include/soci
--   - SOCI_LIBRARIES .............. : /usr/lib64/libsoci_core.so
--   - SOCIMYSQL_LIBRARIES ......... : /usr/lib64/libsoci_mysql.so
--
-- * StdAir:
--   - STDAIR_VERSION .............. : 0.38.0
--   - STDAIR_BINARY_DIRS .......... : /home/dan/dev/deliveries/stdair-0.38.0/bin
--   - STDAIR_EXECUTABLES .......... : stdair
--   - STDAIR_LIBRARY_DIRS ......... : /home/dan/dev/deliveries/stdair-0.38.0/lib64
--   - STDAIR_LIBRARIES ............ : stdairlib;stdairuicllib
--   - STDAIR_INCLUDE_DIRS ......... : /home/dan/dev/deliveries/stdair-0.38.0/include
--   - STDAIR_SAMPLE_DIR ........... : /home/dan/dev/deliveries/stdair-0.38.0/share/stdair/samples
--
-- Change a value with: cmake -D<Variable>=<Value>
-- ================================================================
--
-- Configuring done
-- Generating done
-- Build files have been written to: /home/dan/dev/sim/airinv/airinvgithub/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_airinv
[  0%] Built target hdr_cfg_airrac
[ 13%] Built target airraclib
[ 13%] Built target hdr_cfg_rmol
[ 38%] Built target rmollib
[ 98%] Built target airinvlib
[100%] Built target InventoryTestSuitetst
Scanning dependencies of target check_airinvtst
Test project /home/dan/dev/sim/airinv/airinvgithub/build/test/airinv
    Start 1: InventoryTestSuitetst
1/1 Test #1: InventoryTestSuitetst .............   Passed    0.08 sec

100% tests passed, 0 tests failed out of 1

Total Test time (real) =   0.35 sec
[100%] Built target check_airinvtst
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/airinvgit
rm -rf build && mkdir build
cd build
```

to remove everything.

10 Linking with Airinv

10.1 Table of Contents

10.2 Introdution

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

10.3 Dependencies

The Airinv library depends on several other C++ components.

10.3.1 StdAir

Among them, as for now, only StdAir has been packaged. The support for StdAir is taken in charge by a dedicated M4 macro file (namely, 'stdair.m4'), from the configuration script (generated thanks to 'configure.ac').

Figure 1: Airinv Dependencies

10.4 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 Airinv based program 'my_prog.cpp', you should use the following command:

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

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

10.5 Using the airinv-config script

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

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

```
airinv-config --help
```

If the 'airinv-config' command is not found by your shell, you should add its location '$prefix/bin' to the PATH environment variable, e.g.:

```
export PATH=/usr/local/bin:$PATH
```

10.6 M4 macro for the GNU Autotools

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

- 'Airinv_VERSION' (e.g., defined to 0.23.0)
- 'Airinv_CFLAGS' (e.g., defined to '-I${prefix}/include')
- 'Airinv_LIBS' (e.g., defined to '-L${prefix}/lib -lairinv')

10.7 Using Airinv with dynamic linking

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

11 Test Rules

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

11.1 The Test File

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

The test file should print relevant data to a standard output that can be used to verify the functionality. All relevant parameter settings should be tested.

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

11.2 The Reference File

Consider a test file named 'module_test.cpp'. A reference file named 'module_test.ref' should accompany the test file. The reference file contains a reference printout of the standard output generated when running the test program. The reference file should be maintained using version control and updated according to the test file.

11.3 Testing IT++ Library

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

```
% make check
```

after successful compilation of the IT++ library.

12 Users Guide

12.1 Table of Contents

12.2 Introduction

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

12.3 Get Started

12.3.1 Get the AirInv library

Clone locally the full `Git project`:

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

12.3.2 Build the AirInv project

Link with StdAir, create the distribution package (say, 0.5.0) and compile using the following commands:

```
cd ~/dev/sim/airinvgit
rm -rf build && mkdir -p build
cd build
cmake -DCMAKE_INSTALL_PREFIX=~/dev/deliveries/airinv-0.5.0 \
 -DWITH_STDAIR_PREFIX=~/dev/deliveries/stdair-stable \
 -DCMAKE_BUILD_TYPE:STRING=Debug -DINSTALL_DOC:BOOL=ON ..
make
```

12.3.3 Build and Run the Tests

After building the AirInv project, the following commands run the tests:

```
cd ~/dev/sim/airinvgit
cd build
make check
```

As a result, you should obtain a similar report:

```
[  0%] Built target hdr_cfg_airinv
[ 96%] Built target airinvlib
[100%] Built target AirlineScheduleTestSuitetst
Scanning dependencies of target check_airinvtst
Test project /home/dan/dev/sim/airinv/airinvgithub/build/test/airinv
    Start 1: AirlineScheduleTestSuitetst
1/1 Test #1: AirlineScheduleTestSuitetst ......   Passed    0.15 sec

100% tests passed, 0 tests failed out of 1

Total Test time (real) =   0.40 sec
[100%] Built target check_airinvtst
Scanning dependencies of target check
[100%] Built target check
```

12.3.4 Install the AirInv Project (Binaries, Documentation)

After the step Build the AirInv project, to install the library and its header files, type:

```
cd ~/dev/sim/airinvgit
cd build
make install
```

You can check that the executables and other required files have been copied into the given final directory:

```
cd ~dev/deliveries/airinv-0.5.0
```

To generate the AirInv project documentation, the commands are:

```
cd ~/dev/sim/airinvgit
cd build
make doc
```

The AirInv project documentation is available in the following formats: HTML, LaTeX. Those documents are available in a subdirectory:

```
cd ~/dev/sim/airinvgit
cd build
cd doc
```

12.4 Input file of AirInv Project

The schedule input file structure should look like the following sample:

Each line, beyond the header, represents a schedule entry, i.e., the specification of a given flight-period (see `AIRINV::FlightPeriodStruct`). The fields are as follows:

- Flights section

  - AirlineCode (e.g., BA)
  - FlightNumber (e.g., 9)
  - Start of the flight departure period (e.g., 2007-04-20)
  - End of the flight departure period (e.g., 2007-06-30)
  - Day-Of-the-Week for the flight departure period (DOW) (e.g., 0000011)
  - Leg section
  - Segment section
- Leg section
  - BoardPoint (e.g., LHR)
  - OffPoint (e.g., BKK)
  - BoardTime (e.g., 22:00)
  - ArrivalTime (e.g., 15:15)
  - ArrivalDateOffSet (e.g., +1)
  - ElapsedTime (e.g., 11:15)
  - Leg-cabin section
- Leg-cabin section
  - Cabin code (e.g., F, J, W or Y)
  - Capacity (e.g., respectively 5, 12, 20 or 300)
- Segment section
  - Specificity flag:
    * 0 means that all the segments behave the same way, i.e., have got the same dressing (distribution and order of the booking classes per cabin)
    * 1 means that each segment bahave differently. The full specification of each of those segments must therefore be given.
  - Segment-cabin section
  - Fare family section
- Segment-cabin section
  - Cabin code (e.g., F, J, W or Y)
  - List of (one-letter-code) booking classes for the cabin (e.g, respectively FA, JCDI, WT or YBHKMLSQ)
- Fare family section
  - Fare family code (e.g., 1)
  - List of (one-letter-code) booking classes for the fare family (e.g, respectively FA, JCDI, WT or YBHKMLSQ)

Some fare input examples (including the example above named schedule03.csv) are given in the StdAirproject.

12.5 The schedule BOM Tree

The schedule-related Business Object Model (BOM) tree is a structure allowing to store all the AIRINV::FlightPeriodStruct objects of the simulation. That is why parsing an input file, containing the specification for all the flight-periods, is more convenient (

See also

the previous section Input file of AirInv Project).

As it may be time consuming, and it for sure requires some know-how, to first build such a schedule input file, a small sample BOM tree is provided by default when needed.

12.5.1 Build of the schedule BOM tree

First, a BOM root object (i.e., a root for all the classes in the project) is instantiated by the `stdair::STDAIR_ServiceContext` context object, when the `stdair::STDAIR_Service` is itself instantiated (during the instantiation of the `AIRINV::AIRINV_Service` object).

The corresponding `type` (class) `stdair::BomRoot` is defined in the StdAir library.

Then, the BOM root can be either constructed thanks to the `AIRINV::AIRINV_Service::buildSampleBom()` method:

```
void buildSampleBom();
```

or can be constructed using the schedule input file described above thanks to the `AIRINV::AIRINV_Service::parseAndLoad (const stdair::Filename_T&)` method:

```
void parseAndLoad (const AIRINV::InventoryFilePath
  &);
```

12.5.2 Display of the schedule BOM tree

Note

That feature (of BOM tree display) has not been implemented yet. Do not hesitate to `open a ticket` if you would like to have it implemented more quickly.

The schedule BOM tree can be displayed as done in the `batches::airinv.cpp` program:

When the default BOM tree is used (`-b/-builtin` option of the main program `airinv.cpp`), the schedule BOM tree display (for now, corresponding to `schedule01.csv` parsed by `AIRINV::parseInventory`) should look like:

```
================================================================
BomRoot:  -- ROOT --
================================================================
++++++++++++++++++++++++++++++++++++++++++++++++
Inventory: SQ
++++++++++++++++++++++++++++++++++++++++++++++++
****************************************
FlightDate: SQ11, 2010-Jan-15
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Jan-15, SIN-BKK, 2010-Jan-15, 08:20:00, 2010-Jan-15, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-15, SIN-BKK 2010-Jan-15, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 2, 298
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-15, SIN-BKK 2010-Jan-15, Y, 1, 0, 0, 0, 2, 298, 0,
```

```
SQ11 2010-Jan-15, SIN-BKK 2010-Jan-15, Y, 2, 0, 0, 0, 2, 298, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-15, SIN-BKK 2010-Jan-15, Y, 1, Y, 300 (0), 0, 0, 0, 2, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Jan-15, SIN-BKK 2010-Jan-15, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Jan-16
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Jan-16, SIN-BKK, 2010-Jan-16, 08:20:00, 2010-Jan-16, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-16, SIN-BKK 2010-Jan-16, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 1.83244e-319, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-16, SIN-BKK 2010-Jan-16, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-16, SIN-BKK 2010-Jan-16, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-16, SIN-BKK 2010-Jan-16, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Jan-16, SIN-BKK 2010-Jan-16, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Jan-17
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Jan-17, SIN-BKK, 2010-Jan-17, 08:20:00, 2010-Jan-17, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-17, SIN-BKK 2010-Jan-17, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 1.58896e-319, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-17, SIN-BKK 2010-Jan-17, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-17, SIN-BKK 2010-Jan-17, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
```

```
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-17, SIN-BKK 2010-Jan-17, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Jan-17, SIN-BKK 2010-Jan-17, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Jan-18
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Jan-18, SIN-BKK, 2010-Jan-18, 08:20:00, 2010-Jan-18, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-18, SIN-BKK 2010-Jan-18, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-18, SIN-BKK 2010-Jan-18, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-18, SIN-BKK 2010-Jan-18, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-18, SIN-BKK 2010-Jan-18, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Jan-18, SIN-BKK 2010-Jan-18, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Jan-19
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Jan-19, SIN-BKK, 2010-Jan-19, 08:20:00, 2010-Jan-19, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-19, SIN-BKK 2010-Jan-19, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-19, SIN-BKK 2010-Jan-19, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-19, SIN-BKK 2010-Jan-19, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-19, SIN-BKK 2010-Jan-19, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Jan-19, SIN-BKK 2010-Jan-19, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
```

```
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Jan-20
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Jan-20, SIN-BKK, 2010-Jan-20, 08:20:00, 2010-Jan-20, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-20, SIN-BKK 2010-Jan-20, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-20, SIN-BKK 2010-Jan-20, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-20, SIN-BKK 2010-Jan-20, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-20, SIN-BKK 2010-Jan-20, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Jan-20, SIN-BKK 2010-Jan-20, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Jan-21
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Jan-21, SIN-BKK, 2010-Jan-21, 08:20:00, 2010-Jan-21, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-21, SIN-BKK 2010-Jan-21, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-21, SIN-BKK 2010-Jan-21, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-21, SIN-BKK 2010-Jan-21, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-21, SIN-BKK 2010-Jan-21, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Jan-21, SIN-BKK 2010-Jan-21, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Jan-22
****************************************
```

```
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Jan-22, SIN-BKK, 2010-Jan-22, 08:20:00, 2010-Jan-22, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-22, SIN-BKK 2010-Jan-22, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-22, SIN-BKK 2010-Jan-22, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-22, SIN-BKK 2010-Jan-22, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-22, SIN-BKK 2010-Jan-22, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Jan-22, SIN-BKK 2010-Jan-22, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Jan-23
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Jan-23, SIN-BKK, 2010-Jan-23, 08:20:00, 2010-Jan-23, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-23, SIN-BKK 2010-Jan-23, Y, 300, 300, 0, 0, 0, 0, 0, 0, 6.64029e-
      319, 0, 300, 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-23, SIN-BKK 2010-Jan-23, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-23, SIN-BKK 2010-Jan-23, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-23, SIN-BKK 2010-Jan-23, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Jan-23, SIN-BKK 2010-Jan-23, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Jan-24
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
```

```
SQ11 2010-Jan-24, SIN-BKK, 2010-Jan-24, 08:20:00, 2010-Jan-24, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-24, SIN-BKK 2010-Jan-24, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-24, SIN-BKK 2010-Jan-24, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-24, SIN-BKK 2010-Jan-24, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-24, SIN-BKK 2010-Jan-24, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Jan-24, SIN-BKK 2010-Jan-24, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Jan-25
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Jan-25, SIN-BKK, 2010-Jan-25, 08:20:00, 2010-Jan-25, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-25, SIN-BKK 2010-Jan-25, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-25, SIN-BKK 2010-Jan-25, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-25, SIN-BKK 2010-Jan-25, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-25, SIN-BKK 2010-Jan-25, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Jan-25, SIN-BKK 2010-Jan-25, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Jan-26
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Jan-26, SIN-BKK, 2010-Jan-26, 08:20:00, 2010-Jan-26, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
```

```
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-26, SIN-BKK 2010-Jan-26, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-26, SIN-BKK 2010-Jan-26, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-26, SIN-BKK 2010-Jan-26, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-26, SIN-BKK 2010-Jan-26, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Jan-26, SIN-BKK 2010-Jan-26, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Jan-27
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Jan-27, SIN-BKK, 2010-Jan-27, 08:20:00, 2010-Jan-27, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-27, SIN-BKK 2010-Jan-27, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-27, SIN-BKK 2010-Jan-27, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-27, SIN-BKK 2010-Jan-27, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-27, SIN-BKK 2010-Jan-27, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Jan-27, SIN-BKK 2010-Jan-27, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Jan-28
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Jan-28, SIN-BKK, 2010-Jan-28, 08:20:00, 2010-Jan-28, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-28, SIN-BKK 2010-Jan-28, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
```

```
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-28, SIN-BKK 2010-Jan-28, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-28, SIN-BKK 2010-Jan-28, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-28, SIN-BKK 2010-Jan-28, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Jan-28, SIN-BKK 2010-Jan-28, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Jan-29
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Jan-29, SIN-BKK, 2010-Jan-29, 08:20:00, 2010-Jan-29, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-29, SIN-BKK 2010-Jan-29, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-29, SIN-BKK 2010-Jan-29, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Jan-29, SIN-BKK 2010-Jan-29, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-29, SIN-BKK 2010-Jan-29, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Jan-29, SIN-BKK 2010-Jan-29, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Jan-30
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Jan-30, SIN-BKK, 2010-Jan-30, 08:20:00, 2010-Jan-30, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-30, SIN-BKK 2010-Jan-30, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
```

```
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-30, SIN-BKK 2010-Jan-30, Y, 1, 0, 0, 0, 300, 0,
SQ11 2010-Jan-30, SIN-BKK 2010-Jan-30, Y, 2, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-30, SIN-BKK 2010-Jan-30, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Jan-30, SIN-BKK 2010-Jan-30, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Jan-31
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Jan-31, SIN-BKK, 2010-Jan-31, 08:20:00, 2010-Jan-31, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Jan-31, SIN-BKK 2010-Jan-31, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Jan-31, SIN-BKK 2010-Jan-31, Y, 1, 0, 0, 0, 300, 0,
SQ11 2010-Jan-31, SIN-BKK 2010-Jan-31, Y, 2, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Jan-31, SIN-BKK 2010-Jan-31, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Jan-31, SIN-BKK 2010-Jan-31, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Feb-01
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-01, SIN-BKK, 2010-Feb-01, 08:20:00, 2010-Feb-01, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-01, SIN-BKK 2010-Feb-01, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
```

```
SQ11 2010-Feb-01, SIN-BKK 2010-Feb-01, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-01, SIN-BKK 2010-Feb-01, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-01, SIN-BKK 2010-Feb-01, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-01, SIN-BKK 2010-Feb-01, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Feb-02
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-02, SIN-BKK, 2010-Feb-02, 08:20:00, 2010-Feb-02, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-02, SIN-BKK 2010-Feb-02, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-02, SIN-BKK 2010-Feb-02, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-02, SIN-BKK 2010-Feb-02, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-02, SIN-BKK 2010-Feb-02, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-02, SIN-BKK 2010-Feb-02, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Feb-03
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-03, SIN-BKK, 2010-Feb-03, 08:20:00, 2010-Feb-03, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-03, SIN-BKK 2010-Feb-03, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-03, SIN-BKK 2010-Feb-03, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-03, SIN-BKK 2010-Feb-03, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
```

```
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-03, SIN-BKK 2010-Feb-03, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-03, SIN-BKK 2010-Feb-03, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Feb-04
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-04, SIN-BKK, 2010-Feb-04, 08:20:00, 2010-Feb-04, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-04, SIN-BKK 2010-Feb-04, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-04, SIN-BKK 2010-Feb-04, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-04, SIN-BKK 2010-Feb-04, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-04, SIN-BKK 2010-Feb-04, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-04, SIN-BKK 2010-Feb-04, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Feb-05
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-05, SIN-BKK, 2010-Feb-05, 08:20:00, 2010-Feb-05, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-05, SIN-BKK 2010-Feb-05, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-05, SIN-BKK 2010-Feb-05, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-05, SIN-BKK 2010-Feb-05, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-05, SIN-BKK 2010-Feb-05, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
```

```
SQ11 2010-Feb-05, SIN-BKK 2010-Feb-05, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Feb-06
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-06, SIN-BKK, 2010-Feb-06, 08:20:00, 2010-Feb-06, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-06, SIN-BKK 2010-Feb-06, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-06, SIN-BKK 2010-Feb-06, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-06, SIN-BKK 2010-Feb-06, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-06, SIN-BKK 2010-Feb-06, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-06, SIN-BKK 2010-Feb-06, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Feb-07
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-07, SIN-BKK, 2010-Feb-07, 08:20:00, 2010-Feb-07, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-07, SIN-BKK 2010-Feb-07, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-07, SIN-BKK 2010-Feb-07, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-07, SIN-BKK 2010-Feb-07, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-07, SIN-BKK 2010-Feb-07, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-07, SIN-BKK 2010-Feb-07, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Feb-08
```

```
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-08, SIN-BKK, 2010-Feb-08, 08:20:00, 2010-Feb-08, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-08, SIN-BKK 2010-Feb-08, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-08, SIN-BKK 2010-Feb-08, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-08, SIN-BKK 2010-Feb-08, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-08, SIN-BKK 2010-Feb-08, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-08, SIN-BKK 2010-Feb-08, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Feb-09
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-09, SIN-BKK, 2010-Feb-09, 08:20:00, 2010-Feb-09, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-09, SIN-BKK 2010-Feb-09, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-09, SIN-BKK 2010-Feb-09, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-09, SIN-BKK 2010-Feb-09, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-09, SIN-BKK 2010-Feb-09, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-09, SIN-BKK 2010-Feb-09, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Feb-10
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
```

```
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-10, SIN-BKK, 2010-Feb-10, 08:20:00, 2010-Feb-10, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-10, SIN-BKK 2010-Feb-10, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-10, SIN-BKK 2010-Feb-10, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-10, SIN-BKK 2010-Feb-10, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-10, SIN-BKK 2010-Feb-10, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-10, SIN-BKK 2010-Feb-10, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Feb-11
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-11, SIN-BKK, 2010-Feb-11, 08:20:00, 2010-Feb-11, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-11, SIN-BKK 2010-Feb-11, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-11, SIN-BKK 2010-Feb-11, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-11, SIN-BKK 2010-Feb-11, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-11, SIN-BKK 2010-Feb-11, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-11, SIN-BKK 2010-Feb-11, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Feb-12
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-12, SIN-BKK, 2010-Feb-12, 08:20:00, 2010-Feb-12, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
```

```
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-12, SIN-BKK 2010-Feb-12, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-12, SIN-BKK 2010-Feb-12, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-12, SIN-BKK 2010-Feb-12, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-12, SIN-BKK 2010-Feb-12, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-12, SIN-BKK 2010-Feb-12, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Feb-13
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-13, SIN-BKK, 2010-Feb-13, 08:20:00, 2010-Feb-13, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-13, SIN-BKK 2010-Feb-13, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-13, SIN-BKK 2010-Feb-13, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-13, SIN-BKK 2010-Feb-13, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-13, SIN-BKK 2010-Feb-13, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-13, SIN-BKK 2010-Feb-13, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Feb-14
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-14, SIN-BKK, 2010-Feb-14, 08:20:00, 2010-Feb-14, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-14, SIN-BKK 2010-Feb-14, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
```

```
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-14, SIN-BKK 2010-Feb-14, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-14, SIN-BKK 2010-Feb-14, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-14, SIN-BKK 2010-Feb-14, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-14, SIN-BKK 2010-Feb-14, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Feb-15
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-15, SIN-BKK, 2010-Feb-15, 08:20:00, 2010-Feb-15, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-15, SIN-BKK 2010-Feb-15, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-15, SIN-BKK 2010-Feb-15, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-15, SIN-BKK 2010-Feb-15, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-15, SIN-BKK 2010-Feb-15, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-15, SIN-BKK 2010-Feb-15, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Feb-16
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-16, SIN-BKK, 2010-Feb-16, 08:20:00, 2010-Feb-16, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-16, SIN-BKK 2010-Feb-16, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
```

```
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-16, SIN-BKK 2010-Feb-16, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-16, SIN-BKK 2010-Feb-16, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-16, SIN-BKK 2010-Feb-16, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-16, SIN-BKK 2010-Feb-16, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Feb-17
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-17, SIN-BKK, 2010-Feb-17, 08:20:00, 2010-Feb-17, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-17, SIN-BKK 2010-Feb-17, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-17, SIN-BKK 2010-Feb-17, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-17, SIN-BKK 2010-Feb-17, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-17, SIN-BKK 2010-Feb-17, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-17, SIN-BKK 2010-Feb-17, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Feb-18
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-18, SIN-BKK, 2010-Feb-18, 08:20:00, 2010-Feb-18, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-18, SIN-BKK 2010-Feb-18, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
```

```
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-18, SIN-BKK 2010-Feb-18, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-18, SIN-BKK 2010-Feb-18, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-18, SIN-BKK 2010-Feb-18, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-18, SIN-BKK 2010-Feb-18, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Feb-19
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-19, SIN-BKK, 2010-Feb-19, 08:20:00, 2010-Feb-19, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-19, SIN-BKK 2010-Feb-19, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-19, SIN-BKK 2010-Feb-19, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-19, SIN-BKK 2010-Feb-19, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-19, SIN-BKK 2010-Feb-19, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-19, SIN-BKK 2010-Feb-19, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Feb-20
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-20, SIN-BKK, 2010-Feb-20, 08:20:00, 2010-Feb-20, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-20, SIN-BKK 2010-Feb-20, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-20, SIN-BKK 2010-Feb-20, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-20, SIN-BKK 2010-Feb-20, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
```

```
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-20, SIN-BKK 2010-Feb-20, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-20, SIN-BKK 2010-Feb-20, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Feb-21
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-21, SIN-BKK, 2010-Feb-21, 08:20:00, 2010-Feb-21, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-21, SIN-BKK 2010-Feb-21, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-21, SIN-BKK 2010-Feb-21, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-21, SIN-BKK 2010-Feb-21, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-21, SIN-BKK 2010-Feb-21, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-21, SIN-BKK 2010-Feb-21, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Feb-22
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-22, SIN-BKK, 2010-Feb-22, 08:20:00, 2010-Feb-22, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-22, SIN-BKK 2010-Feb-22, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-22, SIN-BKK 2010-Feb-22, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-22, SIN-BKK 2010-Feb-22, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-22, SIN-BKK 2010-Feb-22, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
```

```
      0, 0, 0, 0, 0,
SQ11 2010-Feb-22, SIN-BKK 2010-Feb-22, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Feb-23
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-23, SIN-BKK, 2010-Feb-23, 08:20:00, 2010-Feb-23, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-23, SIN-BKK 2010-Feb-23, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-23, SIN-BKK 2010-Feb-23, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-23, SIN-BKK 2010-Feb-23, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-23, SIN-BKK 2010-Feb-23, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-23, SIN-BKK 2010-Feb-23, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Feb-24
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-24, SIN-BKK, 2010-Feb-24, 08:20:00, 2010-Feb-24, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-24, SIN-BKK 2010-Feb-24, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-24, SIN-BKK 2010-Feb-24, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-24, SIN-BKK 2010-Feb-24, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-24, SIN-BKK 2010-Feb-24, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-24, SIN-BKK 2010-Feb-24, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
```

```
FlightDate: SQ11, 2010-Feb-25
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-25, SIN-BKK, 2010-Feb-25, 08:20:00, 2010-Feb-25, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-25, SIN-BKK 2010-Feb-25, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-25, SIN-BKK 2010-Feb-25, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-25, SIN-BKK 2010-Feb-25, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-25, SIN-BKK 2010-Feb-25, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-25, SIN-BKK 2010-Feb-25, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Feb-26
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-26, SIN-BKK, 2010-Feb-26, 08:20:00, 2010-Feb-26, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-26, SIN-BKK 2010-Feb-26, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-26, SIN-BKK 2010-Feb-26, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-26, SIN-BKK 2010-Feb-26, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-26, SIN-BKK 2010-Feb-26, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-26, SIN-BKK 2010-Feb-26, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Feb-27
****************************************
****************************************
Leg-Dates:
----------
```

```
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-27, SIN-BKK, 2010-Feb-27, 08:20:00, 2010-Feb-27, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-27, SIN-BKK 2010-Feb-27, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-27, SIN-BKK 2010-Feb-27, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-27, SIN-BKK 2010-Feb-27, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-27, SIN-BKK 2010-Feb-27, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-27, SIN-BKK 2010-Feb-27, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ11, 2010-Feb-28
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ11 2010-Feb-28, SIN-BKK, 2010-Feb-28, 08:20:00, 2010-Feb-28, 11:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ11 2010-Feb-28, SIN-BKK 2010-Feb-28, Y, 300, 300, 0, 0, 0, 0, 0, 0, 0, 0, 300
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ11 2010-Feb-28, SIN-BKK 2010-Feb-28, Y, 1, 0, 0, 0, 0, 300, 0,
SQ11 2010-Feb-28, SIN-BKK 2010-Feb-28, Y, 2, 0, 0, 0, 0, 300, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ11 2010-Feb-28, SIN-BKK 2010-Feb-28, Y, 1, Y, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ11 2010-Feb-28, SIN-BKK 2010-Feb-28, Y, 2, M, 300 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Jan-15
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Jan-15, SIN-HND, 2010-Jan-15, 09:20:00, 2010-Jan-15, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
```

```
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-15, SIN-HND 2010-Jan-15, Y, 200, 200, 2.082e+121, 5.53287e-48, 5.
      20268e-90, 0, 1.31346e-47, 1.05119e-153, 2.78986e+179, 0, 200, 9, 3.66962e-62, 1
      .0854e-71, 6.74783e-67, 6.9835e-77, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-15, SIN-HND 2010-Jan-15, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-15, SIN-HND 2010-Jan-15, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-15, SIN-HND 2010-Jan-15, Y, 1, Y13856, 200 (0), 0, 0, 0, 0, 0 (0)
      , 0, 0, 0, 0, 0, 0,
SQ12 2010-Jan-15, SIN-HND 2010-Jan-15, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Jan-16
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Jan-16, SIN-HND, 2010-Jan-16, 09:20:00, 2010-Jan-16, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-16, SIN-HND 2010-Jan-16, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 2.63638e-319, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-16, SIN-HND 2010-Jan-16, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-16, SIN-HND 2010-Jan-16, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-16, SIN-HND 2010-Jan-16, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Jan-16, SIN-HND 2010-Jan-16, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Jan-17
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Jan-17, SIN-HND, 2010-Jan-17, 09:20:00, 2010-Jan-17, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
```

```
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-17, SIN-HND 2010-Jan-17, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 2.39291e-319, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-17, SIN-HND 2010-Jan-17, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-17, SIN-HND 2010-Jan-17, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-17, SIN-HND 2010-Jan-17, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Jan-17, SIN-HND 2010-Jan-17, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Jan-18
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Jan-18, SIN-HND, 2010-Jan-18, 09:20:00, 2010-Jan-18, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-18, SIN-HND 2010-Jan-18, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 2.14469e-319, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-18, SIN-HND 2010-Jan-18, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-18, SIN-HND 2010-Jan-18, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-18, SIN-HND 2010-Jan-18, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Jan-18, SIN-HND 2010-Jan-18, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Jan-19
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Jan-19, SIN-HND, 2010-Jan-19, 09:20:00, 2010-Jan-19, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-19, SIN-HND 2010-Jan-19, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
```

```
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-19, SIN-HND 2010-Jan-19, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-19, SIN-HND 2010-Jan-19, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-19, SIN-HND 2010-Jan-19, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Jan-19, SIN-HND 2010-Jan-19, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Jan-20
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Jan-20, SIN-HND, 2010-Jan-20, 09:20:00, 2010-Jan-20, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-20, SIN-HND 2010-Jan-20, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-20, SIN-HND 2010-Jan-20, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-20, SIN-HND 2010-Jan-20, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-20, SIN-HND 2010-Jan-20, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Jan-20, SIN-HND 2010-Jan-20, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Jan-21
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Jan-21, SIN-HND, 2010-Jan-21, 09:20:00, 2010-Jan-21, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-21, SIN-HND 2010-Jan-21, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
```

```
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-21, SIN-HND 2010-Jan-21, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-21, SIN-HND 2010-Jan-21, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-21, SIN-HND 2010-Jan-21, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Jan-21, SIN-HND 2010-Jan-21, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Jan-22
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Jan-22, SIN-HND, 2010-Jan-22, 09:20:00, 2010-Jan-22, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-22, SIN-HND 2010-Jan-22, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-22, SIN-HND 2010-Jan-22, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-22, SIN-HND 2010-Jan-22, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-22, SIN-HND 2010-Jan-22, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Jan-22, SIN-HND 2010-Jan-22, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Jan-23
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Jan-23, SIN-HND, 2010-Jan-23, 09:20:00, 2010-Jan-23, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-23, SIN-HND 2010-Jan-23, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-23, SIN-HND 2010-Jan-23, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-23, SIN-HND 2010-Jan-23, Y, 2, 0, 0, 0, 0, 200, 0,
```

```
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-23, SIN-HND 2010-Jan-23, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Jan-23, SIN-HND 2010-Jan-23, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Jan-24
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Jan-24, SIN-HND, 2010-Jan-24, 09:20:00, 2010-Jan-24, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-24, SIN-HND 2010-Jan-24, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-24, SIN-HND 2010-Jan-24, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-24, SIN-HND 2010-Jan-24, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-24, SIN-HND 2010-Jan-24, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Jan-24, SIN-HND 2010-Jan-24, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Jan-25
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Jan-25, SIN-HND, 2010-Jan-25, 09:20:00, 2010-Jan-25, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-25, SIN-HND 2010-Jan-25, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-25, SIN-HND 2010-Jan-25, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-25, SIN-HND 2010-Jan-25, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
```

```
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-25, SIN-HND 2010-Jan-25, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Jan-25, SIN-HND 2010-Jan-25, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Jan-26
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Jan-26, SIN-HND, 2010-Jan-26, 09:20:00, 2010-Jan-26, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-26, SIN-HND 2010-Jan-26, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-26, SIN-HND 2010-Jan-26, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-26, SIN-HND 2010-Jan-26, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-26, SIN-HND 2010-Jan-26, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Jan-26, SIN-HND 2010-Jan-26, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Jan-27
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Jan-27, SIN-HND, 2010-Jan-27, 09:20:00, 2010-Jan-27, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-27, SIN-HND 2010-Jan-27, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-27, SIN-HND 2010-Jan-27, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-27, SIN-HND 2010-Jan-27, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-27, SIN-HND 2010-Jan-27, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Jan-27, SIN-HND 2010-Jan-27, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
```

```
****************************************
****************************************
FlightDate: SQ12, 2010-Jan-28
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Jan-28, SIN-HND, 2010-Jan-28, 09:20:00, 2010-Jan-28, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-28, SIN-HND 2010-Jan-28, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-28, SIN-HND 2010-Jan-28, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-28, SIN-HND 2010-Jan-28, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-28, SIN-HND 2010-Jan-28, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Jan-28, SIN-HND 2010-Jan-28, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Jan-29
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Jan-29, SIN-HND, 2010-Jan-29, 09:20:00, 2010-Jan-29, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-29, SIN-HND 2010-Jan-29, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-29, SIN-HND 2010-Jan-29, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-29, SIN-HND 2010-Jan-29, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-29, SIN-HND 2010-Jan-29, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Jan-29, SIN-HND 2010-Jan-29, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Jan-30
****************************************
****************************************
```

```
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Jan-30, SIN-HND, 2010-Jan-30, 09:20:00, 2010-Jan-30, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-30, SIN-HND 2010-Jan-30, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-30, SIN-HND 2010-Jan-30, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-30, SIN-HND 2010-Jan-30, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-30, SIN-HND 2010-Jan-30, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Jan-30, SIN-HND 2010-Jan-30, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Jan-31
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Jan-31, SIN-HND, 2010-Jan-31, 09:20:00, 2010-Jan-31, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Jan-31, SIN-HND 2010-Jan-31, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Jan-31, SIN-HND 2010-Jan-31, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Jan-31, SIN-HND 2010-Jan-31, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Jan-31, SIN-HND 2010-Jan-31, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Jan-31, SIN-HND 2010-Jan-31, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Feb-01
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-01, SIN-HND, 2010-Feb-01, 09:20:00, 2010-Feb-01, 12:00:00, 07:40:
```

```
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-01, SIN-HND 2010-Feb-01, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-01, SIN-HND 2010-Feb-01, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-01, SIN-HND 2010-Feb-01, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-01, SIN-HND 2010-Feb-01, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-01, SIN-HND 2010-Feb-01, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Feb-02
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-02, SIN-HND, 2010-Feb-02, 09:20:00, 2010-Feb-02, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-02, SIN-HND 2010-Feb-02, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-02, SIN-HND 2010-Feb-02, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-02, SIN-HND 2010-Feb-02, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-02, SIN-HND 2010-Feb-02, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-02, SIN-HND 2010-Feb-02, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Feb-03
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-03, SIN-HND, 2010-Feb-03, 09:20:00, 2010-Feb-03, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
```

```
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-03, SIN-HND 2010-Feb-03, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-03, SIN-HND 2010-Feb-03, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-03, SIN-HND 2010-Feb-03, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-03, SIN-HND 2010-Feb-03, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-03, SIN-HND 2010-Feb-03, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Feb-04
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-04, SIN-HND, 2010-Feb-04, 09:20:00, 2010-Feb-04, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-04, SIN-HND 2010-Feb-04, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-04, SIN-HND 2010-Feb-04, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-04, SIN-HND 2010-Feb-04, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-04, SIN-HND 2010-Feb-04, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-04, SIN-HND 2010-Feb-04, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Feb-05
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-05, SIN-HND, 2010-Feb-05, 09:20:00, 2010-Feb-05, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-05, SIN-HND 2010-Feb-05, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
****************************************
```

```
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-05, SIN-HND 2010-Feb-05, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-05, SIN-HND 2010-Feb-05, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-05, SIN-HND 2010-Feb-05, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-05, SIN-HND 2010-Feb-05, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Feb-06
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-06, SIN-HND, 2010-Feb-06, 09:20:00, 2010-Feb-06, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-06, SIN-HND 2010-Feb-06, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-06, SIN-HND 2010-Feb-06, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-06, SIN-HND 2010-Feb-06, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-06, SIN-HND 2010-Feb-06, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-06, SIN-HND 2010-Feb-06, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Feb-07
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-07, SIN-HND, 2010-Feb-07, 09:20:00, 2010-Feb-07, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-07, SIN-HND 2010-Feb-07, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
```

```
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-07, SIN-HND 2010-Feb-07, Y, 1, 0, 0, 0, 200, 0,
SQ12 2010-Feb-07, SIN-HND 2010-Feb-07, Y, 2, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-07, SIN-HND 2010-Feb-07, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-07, SIN-HND 2010-Feb-07, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Feb-08
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-08, SIN-HND, 2010-Feb-08, 09:20:00, 2010-Feb-08, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-08, SIN-HND 2010-Feb-08, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-08, SIN-HND 2010-Feb-08, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-08, SIN-HND 2010-Feb-08, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-08, SIN-HND 2010-Feb-08, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-08, SIN-HND 2010-Feb-08, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Feb-09
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-09, SIN-HND, 2010-Feb-09, 09:20:00, 2010-Feb-09, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-09, SIN-HND 2010-Feb-09, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-09, SIN-HND 2010-Feb-09, Y, 1, 0, 0, 0, 0, 200, 0,
```

```
SQ12 2010-Feb-09, SIN-HND 2010-Feb-09, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-09, SIN-HND 2010-Feb-09, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-09, SIN-HND 2010-Feb-09, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Feb-10
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-10, SIN-HND, 2010-Feb-10, 09:20:00, 2010-Feb-10, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-10, SIN-HND 2010-Feb-10, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-10, SIN-HND 2010-Feb-10, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-10, SIN-HND 2010-Feb-10, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-10, SIN-HND 2010-Feb-10, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-10, SIN-HND 2010-Feb-10, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Feb-11
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-11, SIN-HND, 2010-Feb-11, 09:20:00, 2010-Feb-11, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-11, SIN-HND 2010-Feb-11, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-11, SIN-HND 2010-Feb-11, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-11, SIN-HND 2010-Feb-11, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
```

```
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-11, SIN-HND 2010-Feb-11, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-11, SIN-HND 2010-Feb-11, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Feb-12
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-12, SIN-HND, 2010-Feb-12, 09:20:00, 2010-Feb-12, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-12, SIN-HND 2010-Feb-12, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-12, SIN-HND 2010-Feb-12, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-12, SIN-HND 2010-Feb-12, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-12, SIN-HND 2010-Feb-12, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-12, SIN-HND 2010-Feb-12, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Feb-13
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-13, SIN-HND, 2010-Feb-13, 09:20:00, 2010-Feb-13, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-13, SIN-HND 2010-Feb-13, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-13, SIN-HND 2010-Feb-13, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-13, SIN-HND 2010-Feb-13, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-13, SIN-HND 2010-Feb-13, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-13, SIN-HND 2010-Feb-13, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
```

```
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Feb-14
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-14, SIN-HND, 2010-Feb-14, 09:20:00, 2010-Feb-14, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-14, SIN-HND 2010-Feb-14, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-14, SIN-HND 2010-Feb-14, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-14, SIN-HND 2010-Feb-14, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-14, SIN-HND 2010-Feb-14, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-14, SIN-HND 2010-Feb-14, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Feb-15
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-15, SIN-HND, 2010-Feb-15, 09:20:00, 2010-Feb-15, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-15, SIN-HND 2010-Feb-15, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-15, SIN-HND 2010-Feb-15, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-15, SIN-HND 2010-Feb-15, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-15, SIN-HND 2010-Feb-15, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-15, SIN-HND 2010-Feb-15, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Feb-16
****************************************
```

```
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-16, SIN-HND, 2010-Feb-16, 09:20:00, 2010-Feb-16, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-16, SIN-HND 2010-Feb-16, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-16, SIN-HND 2010-Feb-16, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-16, SIN-HND 2010-Feb-16, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-16, SIN-HND 2010-Feb-16, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-16, SIN-HND 2010-Feb-16, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Feb-17
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-17, SIN-HND, 2010-Feb-17, 09:20:00, 2010-Feb-17, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-17, SIN-HND 2010-Feb-17, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-17, SIN-HND 2010-Feb-17, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-17, SIN-HND 2010-Feb-17, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-17, SIN-HND 2010-Feb-17, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-17, SIN-HND 2010-Feb-17, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Feb-18
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
```

```
SQ12 2010-Feb-18, SIN-HND, 2010-Feb-18, 09:20:00, 2010-Feb-18, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-18, SIN-HND 2010-Feb-18, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-18, SIN-HND 2010-Feb-18, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-18, SIN-HND 2010-Feb-18, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-18, SIN-HND 2010-Feb-18, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-18, SIN-HND 2010-Feb-18, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Feb-19
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-19, SIN-HND, 2010-Feb-19, 09:20:00, 2010-Feb-19, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-19, SIN-HND 2010-Feb-19, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-19, SIN-HND 2010-Feb-19, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-19, SIN-HND 2010-Feb-19, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-19, SIN-HND 2010-Feb-19, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-19, SIN-HND 2010-Feb-19, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Feb-20
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-20, SIN-HND, 2010-Feb-20, 09:20:00, 2010-Feb-20, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
```

```
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-20, SIN-HND 2010-Feb-20, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-20, SIN-HND 2010-Feb-20, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-20, SIN-HND 2010-Feb-20, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-20, SIN-HND 2010-Feb-20, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-20, SIN-HND 2010-Feb-20, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Feb-21
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-21, SIN-HND, 2010-Feb-21, 09:20:00, 2010-Feb-21, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-21, SIN-HND 2010-Feb-21, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-21, SIN-HND 2010-Feb-21, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-21, SIN-HND 2010-Feb-21, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-21, SIN-HND 2010-Feb-21, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-21, SIN-HND 2010-Feb-21, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Feb-22
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-22, SIN-HND, 2010-Feb-22, 09:20:00, 2010-Feb-22, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-22, SIN-HND 2010-Feb-22, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
```

```
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-22, SIN-HND 2010-Feb-22, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-22, SIN-HND 2010-Feb-22, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-22, SIN-HND 2010-Feb-22, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-22, SIN-HND 2010-Feb-22, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Feb-23
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-23, SIN-HND, 2010-Feb-23, 09:20:00, 2010-Feb-23, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-23, SIN-HND 2010-Feb-23, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-23, SIN-HND 2010-Feb-23, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-23, SIN-HND 2010-Feb-23, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-23, SIN-HND 2010-Feb-23, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-23, SIN-HND 2010-Feb-23, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Feb-24
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-24, SIN-HND, 2010-Feb-24, 09:20:00, 2010-Feb-24, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-24, SIN-HND 2010-Feb-24, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
```

```
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-24, SIN-HND 2010-Feb-24, Y, 1, 0, 0, 0, 200, 0,
SQ12 2010-Feb-24, SIN-HND 2010-Feb-24, Y, 2, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-24, SIN-HND 2010-Feb-24, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-24, SIN-HND 2010-Feb-24, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Feb-25
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-25, SIN-HND, 2010-Feb-25, 09:20:00, 2010-Feb-25, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-25, SIN-HND 2010-Feb-25, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-25, SIN-HND 2010-Feb-25, Y, 1, 0, 0, 0, 200, 0,
SQ12 2010-Feb-25, SIN-HND 2010-Feb-25, Y, 2, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-25, SIN-HND 2010-Feb-25, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-25, SIN-HND 2010-Feb-25, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Feb-26
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-26, SIN-HND, 2010-Feb-26, 09:20:00, 2010-Feb-26, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-26, SIN-HND 2010-Feb-26, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
```

```
SQ12 2010-Feb-26, SIN-HND 2010-Feb-26, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-26, SIN-HND 2010-Feb-26, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-26, SIN-HND 2010-Feb-26, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-26, SIN-HND 2010-Feb-26, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Feb-27
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-27, SIN-HND, 2010-Feb-27, 09:20:00, 2010-Feb-27, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-27, SIN-HND 2010-Feb-27, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-27, SIN-HND 2010-Feb-27, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-27, SIN-HND 2010-Feb-27, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-27, SIN-HND 2010-Feb-27, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-27, SIN-HND 2010-Feb-27, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
****************************************
FlightDate: SQ12, 2010-Feb-28
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset,
      Elapsed, Distance, Capacity,
SQ12 2010-Feb-28, SIN-HND, 2010-Feb-28, 09:20:00, 2010-Feb-28, 12:00:00, 07:40:
      00, 0, -05:00:00, 6300, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group,
      CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
SQ12 2010-Feb-28, SIN-HND 2010-Feb-28, Y, 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 200
      , 9, 0, 0, 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
SQ12 2010-Feb-28, SIN-HND 2010-Feb-28, Y, 1, 0, 0, 0, 0, 200, 0,
SQ12 2010-Feb-28, SIN-HND 2010-Feb-28, Y, 2, 0, 0, 0, 0, 200, 0,
****************************************
****************************************
Subclasses:
```

```
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs,
      GrpBks (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
SQ12 2010-Feb-28, SIN-HND 2010-Feb-28, Y, 1, Y, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
SQ12 2010-Feb-28, SIN-HND 2010-Feb-28, Y, 2, M, 200 (0), 0, 0, 0, 0, 0 (0), 0,
      0, 0, 0, 0, 0,
****************************************
```

12.6 Exploring the Predefined BOM Tree

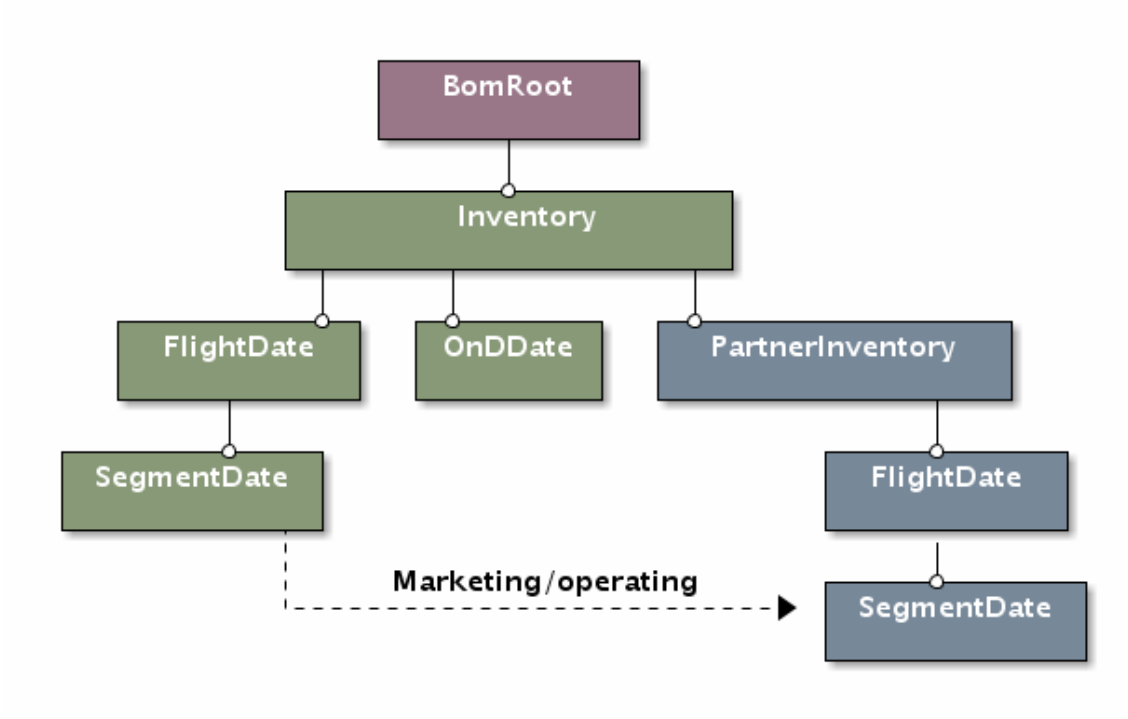


Figure 2: AirInv BOM tree

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

12.6.1 Airline Network BOM Tree

- `AIRINV::ReachableUniverse`
- `AIRINV::OriginDestinationSet`
- `AIRINV::SegmentPathPeriod`

12.6.2 Airline Schedule BOM Tree

- `stdair::Inventory`
- `stdair::FlightPeriod`
- `stdair::SegmentPeriod`
- `stdair::OnDPeriod`

12.7 Extending the BOM Tree

12.8 The travel solution calculation procedure

The project AirInv aims at calculating a list of `travel solutions` for every incoming `booking request`.

13 Supported Systems

13.1 Table of Contents

13.2 Introdution

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

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

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

14 AirInv Supported Systems (Previous Releases)

14.1 AirInv 3.9.1

14.2 AirInv 3.9.0

14.3 AirInv 3.8.1

15 Tutorials

15.1 Table of Contents

15.2 Preparing the AirSched Project for Development

The source code for these examples can be found in the `batches` and `test/airsched` directories. They are compiled along with the rest of the `AirSched` project. See the Users Guide for more details on how to build the `AirSched` project.

15.3 Your first networkBuilde

15.3.1 Summary of the different steps

All the steps below can be found in the same order in the batch `AirSched.cpp` program.

First, we instanciate the AIRSCHED_Service object:

Then, we construct a default sample list of travel solutions and a default booking request (as mentionned in ug_procedure_bookingrequest and ug_procedure_travelsolution parts):

For basic use, the default BOM tree can be built using:

The main step is the network building (see The travel solution calculation procedure):

15.3.2 Result of the Batch Program

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

What is interesting is to compare the travel solution list (here reduced to a single travel solution) displayed before:

and after the network building:

Between the two groups of dashes, we can see that a network option structure has been added by the network builder: the price is 450 EUR for the Y class, the ticket is refundable but there are exchange fees and the customer must stay over on saturday night.

Let's return to our default BOM tree display: the only network rule stored was a match for the travel solution into consideration (same origin airport, same destination airport, flight date included in the network rule date range, same airline "BA", ...).

By looking at the network rule trip type "RT", we can guess we face a round trip network: that means the price given in the default bom tree construction in `stdair::CmdBomManager.hpp` has been divided by 2 because we are considering either an inbound trip or an outbound one.

15.4 Network building with an input file

15.4.1 How to build a network input file?

The objective here is to build a network input file to network builde the default travel solution list built using:

This travel solution list, reduced to a singleton, can be displayed as done before:

We deduce:

- we need a network rule whose origin-destination couple is "LHR, SYD".
- the date range must include the date "2011-06-10".
- the time range must include the time "21:45".
- the airline operating is "BA", so it must be the airline pricing.

We can deduce a part of our network rule file :

We have no information about stay duration and advance purchase (such information are contained into the booking request): so let us put "0" to embrace all the requests possible.

No information for the point-of-sale and the channel too: let us consider all the channels ("IN", "DN", "IF" and DF") and all the points of sale (the origin "LHR", the destination "SYD" and the rest-of-the-world "ROW") existing. To access this information, we could look into the default booking request.

The input file is now:

Let us say we have just the Economy cabin "Y" and Bristish Airways prices ticket for class "Y".

No information about the trip type, so we duplicate all the network rules for both type: one-way "OW" and round-trip "RT" (to access this information, we could look to the default booking request).

The network options are all set to a default value "T" (meaning true) and the network values are chosen to be all distinct.

We obtain:

15.4.2 Building the BOM tree with an input file

The steps are the same as before Summary of the different steps except the bom tree must be built using the network input file :

15.4.3 Result of the Batch Program

When the `AirSched.cpp` program is run with the `-f` option linking with the file built just above:

```
~/AirSched -f ~/<YourFileName>.csv
```

the last lines of the log output should look like:

```
[D]~/AirSchedgit/AirSched/batches/AirSched.cpp:223: Travel solutions:
    [0] [0] BA, 9, 2011-06-10, LHR, SYD, 21:45 --- Y, 145, 1 1 1 ---
```

We have just one network option added to the travel solution. We can deduce from the price value 145 that the network builder used the network rule number 15 to price the travel solution. We have an inbound or outbound trip of a round trip: the total price 290 has been divided by 2.

16 Command-Line Test to Demonstrate How To Test the AirInv Project

```
 */
// //////////////////////////////////////////////////////////////////////
// Import section
// //////////////////////////////////////////////////////////////////////
// STL
#include <sstream>
#include <fstream>
#include <string>
// Boost Unit Test Framework (UTF)
#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MAIN
#define BOOST_TEST_MODULE InventoryTestSuite
#include <boost/test/unit_test.hpp>
// StdAir
#include <stdair/basic/BasLogParams.hpp>
#include <stdair/basic/BasDBParams.hpp>
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/bom/TravelSolutionStruct.hpp>
#include <stdair/bom/BookingRequestStruct.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/stdair_exceptions.hpp>
// Airinv
#include <airinv/AIRINV_Types.hpp>
#include <airinv/AIRINV_Master_Service.hpp>
#include <airinv/config/airinv-paths.hpp>

namespace boost_utf = boost::unit_test;
```

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

// ////////////////////////////////////////////////////////////////////
bool testInventoryHelper (const unsigned short iTestFlag,
                          const stdair::Filename_T& iInventoryInputFilename,
                          const stdair::Filename_T& iScheduleInputFilename,
                          const stdair::Filename_T& iODInputFilename,
                          const stdair::Filename_T& iFRAT5InputFilename,
                          const stdair::Filename_T& iFFDisutilityInputFilename,
                          const stdair::Filename_T& iYieldInputFilename,
                          const bool isBuiltin,
                          const bool isForSchedule) {

  // Output log File
  std::ostringstream oStr;
  oStr << "InventoryTestSuite_" << iTestFlag << ".log";
  const stdair::Filename_T lLogFilename (oStr.str());

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

  // Initialise the AirInv service object
  stdair::BasLogParams lLogParams (stdair::LOG::DEBUG,
                                   logOutputFile);

  // Initialise the inventory service
  AIRINV::AIRINV_Master_Service airinvService (
      lLogParams);

  // Parameters for the sale
  std::string lSegmentDateKey;
  stdair::ClassCode_T lClassCode;
  const stdair::PartySize_T lPartySize (2);

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

    // Build the default sample BOM tree (filled with inventories) for AirInv
   airinvService.buildSampleBom();

   // Define a specific segment-date key for the sample BOM tree
   lSegmentDateKey = "BA,9,2011-06-10,LHR,SYD";
   lClassCode = "Q";

  } else {

    if (isForSchedule == true) {
      // Build the BOM tree from parsing a schedule file (and O&D list)
      stdair::ScheduleFilePath lScheduleFilePath (iScheduleInputFilename);
      stdair::ODFilePath lODFilePath (iODInputFilename);
      stdair::FRAT5FilePath lFRAT5FilePath (iFRAT5InputFilename);
      stdair::FFDisutilityFilePath lFFDisutilityFilePath (
      iFFDisutilityInputFilename);
      AIRRAC::YieldFilePath lYieldFilePath (iYieldInputFilename);
      airinvService.parseAndLoad (lScheduleFilePath, lODFilePath,
                                  lFRAT5FilePath, lFFDisutilityFilePath,
                                  lYieldFilePath);

      // Define a specific segment-date key for the schedule-based inventory
      lSegmentDateKey = "SQ,11,2010-01-15,SIN,BKK";
      lClassCode = "Y";

    } else {

      // Build the BOM tree from parsing an inventory dump file
      AIRINV::InventoryFilePath lInventoryFilePath (
      iInventoryInputFilename);
      airinvService.parseAndLoad (lInventoryFilePath);

      // Define a specific segment-date key for the inventory parsed file
```

```
      //const std::string lSegmentDateKey ("SV, 5, 2010-03-11, KBP, JFK,
       08:00:00");
      lSegmentDateKey =  "SV, 5, 2010-03-11, KBP, JFK, 08:00:00";
      lClassCode = "J";
    }

  }

  // Make a booking
  const bool hasSaleBeenSuccessful =
    airinvService.sell (lSegmentDateKey, lClassCode, lPartySize);

  // DEBUG: Display the list of travel solutions
  const std::string& lCSVDump = airinvService.csvDisplay();
  STDAIR_LOG_DEBUG (lCSVDump);

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

  if (hasSaleBeenSuccessful == false) {
    STDAIR_LOG_DEBUG ("No sale can be made for '" << lSegmentDateKey
                      << "'");
  }

  return hasSaleBeenSuccessful;

}

// ///////////////// 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 (airinv_simple_inventory_sell) {

  // Input file name
  const stdair::Filename_T lInventoryInputFilename (STDAIR_SAMPLE_DIR
                                                    "/invdump01.csv");

  // State whether the BOM tree should be built-in or parsed from an input file
  const bool isBuiltin = false;
  // State whether the BOM tree should be built from a schedule file (instead
       of from an inventory dump)
  const bool isForSchedule = false;

  // Try sell a default segment.
  bool hasTestBeenSuccessful = false;
  BOOST_CHECK_NO_THROW (hasTestBeenSuccessful =
                        testInventoryHelper (0, lInventoryInputFilename,
                                             " ", " ", " ", " ", " ", isBuiltin
      , isForSchedule));
  BOOST_CHECK_EQUAL (hasTestBeenSuccessful, true);

}

BOOST_AUTO_TEST_CASE (airinv_simple_inventory_sell_built_in) {

  // State whether the BOM tree should be built-in or parsed from an input file
  const bool isBuiltin = true;
  // State whether the BOM tree should be built from a schedule file (instead
       of from an inventory dump)
  const bool isForSchedule = false;

  // Try sell a default segment.
  bool hasTestBeenSuccessful = false;
  BOOST_CHECK_NO_THROW (hasTestBeenSuccessful =
                        testInventoryHelper (1, " ", " ", " ", " ", " ", " ",
                                             isBuiltin, isForSchedule));
  BOOST_CHECK_EQUAL (hasTestBeenSuccessful, true);

}

BOOST_AUTO_TEST_CASE (airinv_simple_inventory_sell_schedule) {

  // Input file names
  const stdair::Filename_T lScheduleInputFilename (STDAIR_SAMPLE_DIR
                                                   "/schedule01.csv");
  const stdair::Filename_T lODInputFilename (STDAIR_SAMPLE_DIR
                                             "/ond01.csv");
  const stdair::Filename_T lFRAT5InputFilename (STDAIR_SAMPLE_DIR
                                                "/frat5.csv");
  const stdair::Filename_T lFFDisutilityInputFilename (STDAIR_SAMPLE_DIR
                                                       "/ffDisutility.csv");
```

```
  const stdair::Filename_T lYieldInputFilename (STDAIR_SAMPLE_DIR
                                                "/yieldstore01.csv");

  // State whether the BOM tree should be built-in or parsed from an input file
  const bool isBuiltin = false;
  // State whether the BOM tree should be built from a schedule file (instead
       of from an inventory dump)
  const bool isForSchedule = true;

  // Try sell a default segment.
  bool hasTestBeenSuccessful = false;
  BOOST_CHECK_NO_THROW (hasTestBeenSuccessful =
                        testInventoryHelper (2, " ",
                                             lScheduleInputFilename,
                                             lODInputFilename,
                                             lFRAT5InputFilename,
                                             lFFDisutilityInputFilename,
                                             lYieldInputFilename,
                                             isBuiltin, isForSchedule));
  BOOST_CHECK_EQUAL (hasTestBeenSuccessful, true);

}

BOOST_AUTO_TEST_CASE (airinv_error_inventory_input_file) {

  // Inventory input file name
  const stdair::Filename_T lMissingInventoryFilename (STDAIR_SAMPLE_DIR
                                                      "/missingFile.csv");

  // State whether the BOM tree should be built-in or parsed from an input file
  const bool isBuiltin = false;
  // State whether the BOM tree should be built from a schedule file (instead
       of from an inventory dump)
  const bool isForSchedule = false;

  // Try sell a default segment.
  BOOST_CHECK_THROW (testInventoryHelper (3, lMissingInventoryFilename,
                                          " ", " ", " ", " ", " ", isBuiltin,
      isForSchedule),
                     AIRINV::InventoryInputFileNotFoundException
      );

}

BOOST_AUTO_TEST_CASE (airinv_error_schedule_input_file) {

  // Schedule input file name
  const stdair::Filename_T lMissingScheduleFilename (STDAIR_SAMPLE_DIR
                                                     "/missingFile.csv");
  const stdair::Filename_T lFRAT5InputFilename (STDAIR_SAMPLE_DIR
                                                "/frat5.csv");
  const stdair::Filename_T lFFDisutilityInputFilename (STDAIR_SAMPLE_DIR
                                                       "/ffDisutility.csv");

  // State whether the BOM tree should be built-in or parsed from an input file
  const bool isBuiltin = false;
  // State whether the BOM tree should be built from a schedule file (instead
       of from an inventory dump)
  const bool isForSchedule = true;

  // Try sell a default segment.
  BOOST_CHECK_THROW (testInventoryHelper (4, " ", lMissingScheduleFilename,
                                          " ", lFRAT5InputFilename,
                                          lFFDisutilityInputFilename, " ",
                                          isBuiltin, isForSchedule),
                     AIRINV::ScheduleInputFileNotFoundException
      );

}

BOOST_AUTO_TEST_CASE (airinv_error_yield_input_file) {

  // Input file names
  const stdair::Filename_T lScheduleInputFilename (STDAIR_SAMPLE_DIR
                                                   "/schedule01.csv");
  const stdair::Filename_T lODInputFilename (STDAIR_SAMPLE_DIR
                                             "/ond01.csv");
  const stdair::Filename_T lFRAT5InputFilename (STDAIR_SAMPLE_DIR
                                                "/frat5.csv");
  const stdair::Filename_T lFFDisutilityInputFilename (STDAIR_SAMPLE_DIR
                                                       "/ffDisutility.csv");
  const stdair::Filename_T lYieldInputFilename (STDAIR_SAMPLE_DIR
                                                "/missingFile.csv");

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

```
      of from an inventory dump)
  const bool isForSchedule = true;

  // Try sell a default segment.
  BOOST_CHECK_THROW (testInventoryHelper (5, " ",
                                          lScheduleInputFilename,
                                          lODInputFilename,
                                          lFRAT5InputFilename,
                                          lFFDisutilityInputFilename,
                                          lYieldInputFilename,
                                          isBuiltin, isForSchedule),
                     AIRRAC::YieldInputFileNotFoundException);

}

BOOST_AUTO_TEST_CASE (airinv_error_flight_date_duplication) {

  // Input file names
  const stdair::Filename_T lScheduleInputFilename (STDAIR_SAMPLE_DIR
                                                   "/scheduleError01.csv");
  const stdair::Filename_T lODInputFilename (STDAIR_SAMPLE_DIR
                                             "/ond01.csv");
  const stdair::Filename_T lFRAT5InputFilename (STDAIR_SAMPLE_DIR
                                                "/frat5.csv");
  const stdair::Filename_T lFFDisutilityInputFilename (STDAIR_SAMPLE_DIR
                                                       "/ffDisutility.csv");
  const stdair::Filename_T lYieldInputFilename (STDAIR_SAMPLE_DIR
                                                "/missingFile.csv");

  // State whether the BOM tree should be built-in or parsed from an input file
  const bool isBuiltin = false;
  // State whether the BOM tree should be built from a schedule file (instead
      of from an inventory dump)
  const bool isForSchedule = true;

  // Try sell a default segment.
  BOOST_CHECK_THROW (testInventoryHelper (6, " ",
                                          lScheduleInputFilename,
                                          lODInputFilename,
                                          lFRAT5InputFilename,
                                          lFFDisutilityInputFilename,
                                          lYieldInputFilename,
                                          isBuiltin, isForSchedule),
                     AIRINV::FlightDateDuplicationException
      );

}

BOOST_AUTO_TEST_CASE (airinv_error_schedule_parsing_failed) {

  // Input file names
  const stdair::Filename_T lScheduleInputFilename (STDAIR_SAMPLE_DIR
                                                   "/scheduleError02.csv");
  const stdair::Filename_T lODInputFilename (STDAIR_SAMPLE_DIR
                                             "/ond01.csv");
  const stdair::Filename_T lFRAT5InputFilename (STDAIR_SAMPLE_DIR
                                                "/frat5.csv");
  const stdair::Filename_T lFFDisutilityInputFilename (STDAIR_SAMPLE_DIR
                                                       "/ffDisutility.csv");
  const stdair::Filename_T lYieldInputFilename (STDAIR_SAMPLE_DIR
                                                "/yieldstore01.csv");

  // State whether the BOM tree should be built-in or parsed from an input file
  const bool isBuiltin = false;
  // State whether the BOM tree should be built from a schedule file (instead
      of from an inventory dump)
  const bool isForSchedule = true;

  // Try sell a default segment.
  BOOST_CHECK_THROW (testInventoryHelper (7, " ",
                                          lScheduleInputFilename,
                                          lODInputFilename,
                                          lFRAT5InputFilename,
                                          lFFDisutilityInputFilename,
                                          lYieldInputFilename,
                                          isBuiltin, isForSchedule),
                     AIRINV::ScheduleFileParsingFailedException
      );

}

// End the test suite
BOOST_AUTO_TEST_SUITE_END()

/*!
```

17 Namespace Index

17.1 Namespace List

Here is a list of all namespaces with brief descriptions:

18 Class Index

18.1 Class Hierarchy

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

19 Class Index

19.1 Class List

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

20 File Index

20.1 File List

Here is a list of all files with brief descriptions:

21 Namespace Documentation

21.1 AIRINV Namespace Reference

Namespaces

- namespace FFDisutilityParserHelper
- namespace FRAT5ParserHelper
- namespace InventoryParserHelper
- namespace ScheduleParserHelper
- namespace DCPParserHelper

Classes

- class AIRINV_Master_Service

  Interface for the AIRINV Services.
- class AIRINV_Service

  Interface for the AIRINV Services.
- class InventoryFileParsingFailedException
- class ScheduleFileParsingFailedException
- class MissingPartnerFlightDateWithinScheduleFile
- class FRAT5FileParsingFailedException
- class FFDisutilityFileParsingFailedException
- class SegmentDateNotFoundException
- class InventoryInputFileNotFoundException
- class ScheduleInputFileNotFoundException
- class FRAT5InputFileNotFoundException
- class FFDisutilityInputFileNotFoundException
- class FlightDateDuplicationException
- class BookingException
- class InventoryNotFoundException
- class FlightDateNotFoundException
- class InventoryFilePath
- struct DefaultMap
- struct FlightTypeCode

- struct FlightVisibilityCode
- class BomAbstract
- class BomRootHelper
- class BookingClassHelper
- struct BookingClassStruct
- struct BucketStruct

  Utility Structure for the parsing of Bucket structures.
- struct DCPEventStruct
- struct FareFamilyStruct

  Utility Structure for the parsing of fare family details.
- struct FFDisutilityStruct
- class FlightDateHelper
- struct FlightDateStruct
- struct FlightPeriodStruct
- struct FRAT5Struct
- class InventoryHelper
- class LegCabinHelper
- struct LegCabinStruct
- struct LegStruct
- class SegmentCabinHelper

  Class representing the actual business functions for an airline segment-cabin.
- struct SegmentCabinStruct

  Utility Structure for the parsing of SegmentCabin details.
- class SegmentDateHelper
- class SegmentSnapshotTableHelper
- struct SegmentStruct
- class FFDisutilityParser

  Class wrapping the parser entry point.
- class FFDisutilityFileParser
- class FRAT5Parser

  Class wrapping the parser entry point.
- class FRAT5FileParser
- class InventoryBuilder

  Class handling the generation / instantiation of the Inventory BOM.
- class InventoryGenerator

  Class handling the generation / instantiation of the Inventory BOM.
- class InventoryManager
- class InventoryParser

  Class wrapping the parser entry point.
- class InventoryFileParser
- class ScheduleParser

  Class wrapping the parser entry point.
- class FlightPeriodFileParser
- class DCPEventGenerator
- class DCPParser
- class DCPRuleFileParser
- class FacAirinvMasterServiceContext

  Factory for Bucket.
- class FacAirinvServiceContext
- class FacBomAbstract
- class FacServiceAbstract
- class FacSupervisor
- struct FlightRequestStatus

- class AirInvServer
- class Connection
- struct header
- struct Reply
- struct Request
- class RequestHandler

  The common handler for all incoming requests.
- class RequestParser

  Parser for incoming requests.
- class AIRINV_Master_ServiceContext
- class AIRINV_ServiceContext

  Class holding the context of the AirInv services.
- class ServiceAbstract

Typedefs

- typedef boost::shared_ptr < AIRINV_Service > AIRINV_ServicePtr_T
- typedef boost::shared_ptr < AIRINV_Master_Service > AIRINV_Master_ServicePtr_T
- typedef std::map< const stdair::AirlineCode_T, AIRINV_ServicePtr_T > AIRINV_ServicePtr_Map_T
- typedef std::map< const stdair::DTD_T, double > FRAT5Curve_T
- typedef char char_t
- typedef boost::spirit::classic::file_iterator < char_t > iterator_t
- typedef boost::spirit::classic::scanner < iterator_t > scanner_t
- typedef boost::spirit::classic::rule < scanner_t > rule_t
- typedef boost::spirit::classic::int_parser < unsigned int, 10, 1, 1 > int1_p_t
- typedef boost::spirit::classic::uint_parser < unsigned int, 10, 2, 2 > uint2_p_t
- typedef boost::spirit::classic::uint_parser < unsigned int, 10, 1, 2 > uint1_2_p_t
- typedef boost::spirit::classic::uint_parser < unsigned int, 10, 1, 3 > uint1_3_p_t
- typedef boost::spirit::classic::uint_parser < unsigned int, 10, 4, 4 > uint4_p_t
- typedef boost::spirit::classic::uint_parser < unsigned int, 10, 1, 4 > uint1_4_p_t
- typedef boost::spirit::classic::chset < char_t > chset_t

- typedef boost::spirit::classic::impl::loop_traits < chset_t, unsigned int, unsigned int >::type repeat_p_t
- typedef boost::spirit::classic::bounded < uint2_p_t, unsigned int > bounded2_p_t
- typedef boost::spirit::classic::bounded < uint1_2_p_t, unsigned int > bounded1_2_p_t
- typedef boost::spirit::classic::bounded < uint1_3_p_t, unsigned int > bounded1_3_p_t
- typedef boost::spirit::classic::bounded < uint4_p_t, unsigned int > bounded4_p_t
- typedef boost::spirit::classic::bounded < uint1_4_p_t, unsigned int > bounded1_4_p_t
- typedef std::set < stdair::AirportCode_T > AirportList_T
- typedef std::vector < stdair::AirportCode_T > AirportOrderedList_T
- typedef std::vector < BookingClassStruct > BookingClassStructList_T
- typedef std::vector< BucketStruct > BucketStructList_T
- typedef std::vector < FareFamilyStruct > FareFamilyStructList_T
- typedef std::vector < LegCabinStruct > LegCabinStructList_T
- typedef std::vector< LegStruct > LegStructList_T
- typedef std::vector < SegmentCabinStruct > SegmentCabinStructList_T
- typedef std::vector < SegmentStruct > SegmentStructList_T
- typedef std::map< const stdair::Date_T, stdair::SegmentCabin * > DepartureDateSegmentCabinMap_T
- typedef std::map< const std::string, DepartureDateSegmentCabinMap_T > SimilarSegmentCabinSetMap_T
- typedef boost::shared_ptr < boost::thread > ThreadShrPtr_T
- typedef std::vector < ThreadShrPtr_T > ThreadShrPtrList_T
- typedef boost::shared_ptr < Connection > ConnectionShrPtr_T

Variables

- const std::string DEFAULT_AIRLINE_CODE = "BA"
- const FRAT5Curve_T DEFAULT_PICKUP_FRAT5_CURVE

21.1.1 Typedef Documentation

21.1.1.1 typedef boost::shared_ptr<AIRINV_Service> AIRINV::AIRINV_ServicePtr_T

(Smart) Pointer on the AirInv (slave) service handler.

Definition at line 210 of file AIRINV_Types.hpp.

21.1.1.2 typedef boost::shared_ptr<AIRINV_Master_Service> AIRINV::AIRINV_Master_ServicePtr_T

(Smart) Pointer on the AirInv master service handler.

Definition at line 215 of file AIRINV_Types.hpp.

21.1.1.3 typedef std::map<const stdair::AirlineCode_T, AIRINV_ServicePtr_T> AIRINV::AIRINV_ServicePtr_Map_T

Type defining a map of airline codes and the corresponding airline inventories.

Definition at line 222 of file AIRINV_Types.hpp.

21.1.1.4 typedef std::map<const stdair::DTD_T, double> AIRINV::FRAT5Curve_T

Define the FRAT5 curve.

Definition at line 227 of file AIRINV_Types.hpp.

21.1.1.5 typedef char AIRINV::char_t

Definition at line 31 of file BasParserTypes.hpp.

21.1.1.6 typedef boost::spirit::classic::file_iterator<char_t> AIRINV::iterator_t

Definition at line 35 of file BasParserTypes.hpp.

21.1.1.7 typedef boost::spirit::classic::scanner<iterator_t> AIRINV::scanner_t

Definition at line 36 of file BasParserTypes.hpp.

21.1.1.8 typedef boost::spirit::classic::rule<scanner_t> AIRINV::rule_t

Definition at line 37 of file BasParserTypes.hpp.

21.1.1.9 typedef boost::spirit::classic::int_parser<unsigned int, 10, 1, 1> AIRINV::int1_p_t

1-digit-integer parser

Definition at line 45 of file BasParserTypes.hpp.

21.1.1.10 typedef boost::spirit::classic::uint_parser<unsigned int, 10, 2, 2> AIRINV::uint2_p_t

2-digit-integer parser

Definition at line 48 of file BasParserTypes.hpp.

21.1.1.11 typedef boost::spirit::classic::uint_parser<unsigned int, 10, 1, 2> AIRINV::uint1_2_p_t

Up-to-2-digit-integer parser

Definition at line 51 of file BasParserTypes.hpp.

21.1.1.12 typedef boost::spirit::classic::uint_parser<unsigned int, 10, 1, 3> AIRINV::uint1_3_p_t

Up-to-3-digit-integer parser

Definition at line 54 of file BasParserTypes.hpp.

21.1.1.13 typedef boost::spirit::classic::uint_parser<unsigned int, 10, 4, 4> AIRINV::uint4_p_t

4-digit-integer parser

Definition at line 57 of file BasParserTypes.hpp.

21.1.1.14 typedef boost::spirit::classic::uint_parser<unsigned int, 10, 1, 4> AIRINV::uint1_4_p_t

Up-to-4-digit-integer parser

Definition at line 60 of file BasParserTypes.hpp.

21.1.1.15 typedef boost::spirit::classic::chset<char_t> AIRINV::chset_t

character set

Definition at line 63 of file BasParserTypes.hpp.

21.1.1.16 typedef boost::spirit::classic::impl::loop_traits<chset_t, unsigned int, unsigned int>::type AIRINV::repeat_p_t

(Repeating) sequence of a given number of characters: repeat_p(min, max)

Definition at line 69 of file BasParserTypes.hpp.

21.1.1.17 typedef boost::spirit::classic::bounded<uint2_p_t, unsigned int> AIRINV::bounded2_p_t

Bounded-number-of-integers parser

Definition at line 72 of file BasParserTypes.hpp.

21.1.1.18 typedef boost::spirit::classic::bounded<uint1_2_p_t, unsigned int> AIRINV::bounded1_2_p_t

Definition at line 73 of file BasParserTypes.hpp.

21.1.1.19 typedef boost::spirit::classic::bounded<uint1_3_p_t, unsigned int> AIRINV::bounded1_3_p_t

Definition at line 74 of file BasParserTypes.hpp.

21.1.1.20 typedef boost::spirit::classic::bounded<uint4_p_t, unsigned int> AIRINV::bounded4_p_t

Definition at line 75 of file BasParserTypes.hpp.

21.1.1.21 typedef boost::spirit::classic::bounded<uint1_4_p_t, unsigned int> AIRINV::bounded1_4_p_t

Definition at line 76 of file BasParserTypes.hpp.

21.1.1.22 typedef std::set<stdair::AirportCode_T> AIRINV::AirportList_T

Define lists of Airport Codes.

Definition at line 16 of file AirportList.hpp.

21.1.1.23 typedef std::vector<stdair::AirportCode_T> AIRINV::AirportOrderedList_T

Definition at line 17 of file AirportList.hpp.

21.1.1.24 typedef std::vector<BookingClassStruct> AIRINV::BookingClassStructList_T

List of BookingClass structures.

Definition at line 60 of file BookingClassStruct.hpp.

21.1.1.25 typedef std::vector<BucketStruct> AIRINV::BucketStructList_T

List of Bucket structures.

Definition at line 44 of file BucketStruct.hpp.

21.1.1.26 typedef std::vector<FareFamilyStruct> AIRINV::FareFamilyStructList_T

List of FareFamily-Detail structures.

Definition at line 59 of file FareFamilyStruct.hpp.

21.1.1.27 typedef std::vector<LegCabinStruct> AIRINV::LegCabinStructList_T

List of LegCabin-Detail strucutres.

Definition at line 52 of file LegCabinStruct.hpp.

21.1.1.28 typedef std::vector<LegStruct> AIRINV::LegStructList_T

List of Leg structures.

Definition at line 57 of file LegStruct.hpp.

21.1.1.29 typedef std::vector<SegmentCabinStruct> AIRINV::SegmentCabinStructList_T

List of SegmentCabin-Detail strucutres.

Definition at line 48 of file SegmentCabinStruct.hpp.

21.1.1.30 typedef std::vector<SegmentStruct> AIRINV::SegmentStructList_T

List of Segment strucutres.

Definition at line 43 of file SegmentStruct.hpp.

21.1.1.31 typedef std::map<const stdair::Date_T, stdair::SegmentCabin*> AIRINV::DepartureDateSegmentCabinMap_T

Definition at line 31 of file InventoryManager.hpp.

21.1.1.32 typedef std::map<const std::string, DepartureDateSegmentCabinMap_T> AIRINV::SimilarSegmentCabinSetMap_T

Definition at line 33 of file InventoryManager.hpp.

21.1.1.33 typedef boost::shared_ptr<boost::thread> AIRINV::ThreadShrPtr_T

Definition at line 15 of file AirInvServer_ASIO.cpp.

21.1.1.34 typedef std::vector<ThreadShrPtr_T> AIRINV::ThreadShrPtrList_T

Definition at line 16 of file AirInvServer_ASIO.cpp.

21.1.1.35 typedef boost::shared_ptr<Connection> AIRINV::ConnectionShrPtr_T

Shared pointer on a Connection object.

Definition at line 71 of file Connection.hpp.

21.1.2 Variable Documentation

21.1.2.1 const std::string AIRINV::DEFAULT_AIRLINE_CODE = "BA"

Default airline name for the AIRINV_Service.

Definition at line 11 of file BasConst.cpp.

21.1.2.2 const FRAT5Curve_T AIRINV::DEFAULT_PICKUP_FRAT5_CURVE

Initial value:

```
DefaultMap::createPickupFRAT5Curve()
```

Default pick-up FRAT5 curve for Q-equivalent booking conversion.

Definition at line 14 of file BasConst.cpp.

21.2 AIRINV::DCPParserHelper Namespace Reference

Classes

- struct ParserSemanticAction
- struct storeDCPId
- struct storeOrigin
- struct storeDestination
- struct storeDateRangeStart
- struct storeDateRangeEnd
- struct storeStartRangeTime
- struct storeEndRangeTime
- struct storePOS
- struct storeCabinCode
- struct storeChannel
- struct storeAdvancePurchase
- struct storeSaturdayStay
- struct storeChangeFees
- struct storeNonRefundable
- struct storeMinimumStay
- struct storeDCP
- struct storeAirlineCode
- struct storeClass
- struct doEndDCP
- struct DCPRuleParser

Variables

- stdair::int1_p_t int1_p
- stdair::uint2_p_t uint2_p
- stdair::uint4_p_t uint4_p
- stdair::uint1_4_p_t uint1_4_p
- stdair::hour_p_t hour_p
- stdair::minute_p_t minute_p
- stdair::second_p_t second_p
- stdair::year_p_t year_p
- stdair::month_p_t month_p
- stdair::day_p_t day_p

21.2.1 Variable Documentation

21.2.1.1 stdair::int1_p_t AIRINV::DCPParserHelper::int1_p

Namespaces. 1-digit-integer parser

Definition at line 427 of file DCPParserHelper.cpp.

21.2.1.2 stdair::uint2_p_t AIRINV::DCPParserHelper::uint2_p

2-digit-integer parser

Definition at line 430 of file DCPParserHelper.cpp.

21.2.1.3 stdair::uint4_p_t AIRINV::DCPParserHelper::uint4_p

4-digit-integer parser

Definition at line 433 of file DCPParserHelper.cpp.

21.2.1.4 stdair::uint1_4_p_t AIRINV::DCPParserHelper::uint1_4_p

Up-to-4-digit-integer parser

Definition at line 436 of file DCPParserHelper.cpp.

Referenced by AIRINV::DCPParserHelper::DCPRuleParser::DCPRuleParser().

21.2.1.5 stdair::hour_p_t AIRINV::DCPParserHelper::hour_p

Time element parsers.

Definition at line 439 of file DCPParserHelper.cpp.

Referenced by AIRINV::DCPParserHelper::DCPRuleParser::DCPRuleParser().

21.2.1.6 stdair::minute_p_t AIRINV::DCPParserHelper::minute_p

Definition at line 440 of file DCPParserHelper.cpp.

Referenced by AIRINV::DCPParserHelper::DCPRuleParser::DCPRuleParser().

21.2.1.7 stdair::second_p_t AIRINV::DCPParserHelper::second_p

Definition at line 441 of file DCPParserHelper.cpp.

Referenced by AIRINV::DCPParserHelper::DCPRuleParser::DCPRuleParser().

21.2.1.8 stdair::year_p_t AIRINV::DCPParserHelper::year_p

Date element parsers.

Definition at line 444 of file DCPParserHelper.cpp.

Referenced by AIRINV::DCPParserHelper::DCPRuleParser::DCPRuleParser().

21.2.1.9 stdair::month_p_t AIRINV::DCPParserHelper::month_p

Definition at line 445 of file DCPParserHelper.cpp.

Referenced by AIRINV::DCPParserHelper::DCPRuleParser::DCPRuleParser().

21.2.1.10 stdair::day_p_t AIRINV::DCPParserHelper::day_p

Definition at line 446 of file DCPParserHelper.cpp.

Referenced by AIRINV::DCPParserHelper::DCPRuleParser::DCPRuleParser().

21.3 AIRINV::FFDisutilityParserHelper Namespace Reference

Classes

- struct ParserSemanticAction
- struct storeCurveKey

- struct storeDTD
- struct storeFFDisutilityValue
- struct doEndCurve
- struct FFDisutilityParser

Functions

- repeat_p_t key_p (chset_t("0-9A-Z").derived(), 1, 10)

21.3.1 Function Documentation

21.3.1.1 repeat_p_t AIRINV::FFDisutilityParserHelper::key_p (chset_t("0-9A-Z").derived() , 1 , 10)

Key Parser: repeat_p(1,10)[chset_p("0-9A-Z")]

Referenced by AIRINV::FFDisutilityParserHelper::FFDisutilityParser::definition< ScannerT >::definition().

21.4 AIRINV::FRAT5ParserHelper Namespace Reference

Classes

- struct ParserSemanticAction
- struct storeCurveKey
- struct storeDTD
- struct storeFRAT5Value
- struct doEndCurve
- struct FRAT5Parser

Functions

- repeat_p_t key_p (chset_t("0-9A-Z").derived(), 1, 10)

21.4.1 Function Documentation

21.4.1.1 repeat_p_t AIRINV::FRAT5ParserHelper::key_p (chset_t("0-9A-Z").derived() , 1 , 10)

Key Parser: repeat_p(1,10)[chset_p("0-9A-Z")]

Referenced by AIRINV::FRAT5ParserHelper::FRAT5Parser::definition< ScannerT >::definition().

21.5 AIRINV::InventoryParserHelper Namespace Reference

Classes

- struct ParserSemanticAction
- struct storeSnapshotDate
- struct storeAirlineCode
- struct storeFlightNumber
- struct storeFlightDate
- struct storeFlightTypeCode
- struct storeFlightVisibilityCode
- struct storeLegBoardingPoint
- struct storeLegOffPoint
- struct storeOperatingAirlineCode

- struct storeOperatingFlightNumber
- struct storeBoardingDate
- struct storeBoardingTime
- struct storeOffDate
- struct storeOffTime
- struct storeLegCabinCode
- struct storeSaleableCapacity
- struct storeAU
- struct storeUPR
- struct storeBookingCounter
- struct storeNAV
- struct storeGAV
- struct storeACP
- struct storeETB
- struct storeYieldUpperRange
- struct storeBucketAvaibality
- struct storeSeatIndex
- struct storeSegmentBoardingPoint
- struct storeSegmentOffPoint
- struct storeSegmentCabinCode
- struct storeSegmentCabinBookingCounter
- struct storeClassCode
- struct storeSubclassCode
- struct storeParentClassCode
- struct storeParentSubclassCode
- struct storeCumulatedProtection
- struct storeProtection
- struct storeNego
- struct storeNoShow
- struct storeOverbooking
- struct storeNbOfBkgs
- struct storeNbOfGroupBkgs
- struct storeNbOfPendingGroupBkgs
- struct storeNbOfStaffBkgs
- struct storeNbOfWLBkgs
- struct storeClassETB
- struct storeClassAvailability
- struct storeSegmentAvailability
- struct storeRevenueAvailability
- struct storeFamilyCode
- struct storeFClasses
- struct doEndFlightDate
- struct InventoryParser

Functions

- repeat_p_t airline_code_p (chset_t("0-9A-Z").derived(), 2, 3)
- bounded1_4_p_t flight_number_p (uint1_4_p.derived(), 0u, 9999u)
- bounded2_p_t year_p (uint2_p.derived(), 0u, 99u)
- bounded2_p_t month_p (uint2_p.derived(), 1u, 12u)
- bounded2_p_t day_p (uint2_p.derived(), 1u, 31u)
- repeat_p_t dow_p (chset_t("0-1").derived().derived(), 7, 7)
- repeat_p_t airport_p (chset_t("0-9A-Z").derived(), 3, 3)
- bounded1_2_p_t hours_p (uint1_2_p.derived(), 0u, 24u)

- bounded2_p_t minutes_p (uint2_p.derived(), 0u, 59u)
- bounded2_p_t seconds_p (uint2_p.derived(), 0u, 59u)
- chset_t cabin_code_p ("A-Z")
- chset_t class_code_p ("A-Z")
- chset_t passenger_type_p ("A-Z")
- repeat_p_t class_code_list_p (chset_t("A-Z").derived(), 1, 26)
- bounded1_3_p_t stay_duration_p (uint1_3_p.derived(), 0u, 999u)

Variables

- int1_p_t int1_p
- uint2_p_t uint2_p
- uint1_2_p_t uint1_2_p
- uint1_3_p_t uint1_3_p
- uint4_p_t uint4_p
- uint1_4_p_t uint1_4_p
- int1_p_t family_code_p

21.5.1 Function Documentation

21.5.1.1 repeat_p_t AIRINV::InventoryParserHelper::airline_code_p (chset_t("0-9A-Z").derived() , 2 , 3)

Airline Code Parser: repeat_p(2,3)[chset_p("0-9A-Z")]

Referenced by AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::definition().

21.5.1.2 bounded1_4_p_t AIRINV::InventoryParserHelper::flight_number_p (uint1_4_p. *derived()*, 0u , 9999u)

Flight Number Parser: limit_d(0u, 9999u)[uint1_4_p]

Referenced by AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::definition().

21.5.1.3 bounded2_p_t AIRINV::InventoryParserHelper::year_p (uint2_p. *derived()*, 0u , 99u)

Year Parser: limit_d(00u, 99u)[uint4_p]

Referenced by AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::definition().

21.5.1.4 bounded2_p_t AIRINV::InventoryParserHelper::month_p (uint2_p. *derived()*, 1u , 12u)

Month Parser: limit_d(1u, 12u)[uint2_p]

Referenced by AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::definition().

21.5.1.5 bounded2_p_t AIRINV::InventoryParserHelper::day_p (uint2_p. *derived()*, 1u , 31u)

Day Parser: limit_d(1u, 31u)[uint2_p]

Referenced by AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::definition().

21.5.1.6 repeat_p_t AIRINV::InventoryParserHelper::dow_p (chset_t("0-1").derived().derived() , 7 , 7)

DOW (Day-Of-the-Week) Parser: repeat_p(7)[chset_p("0-1")]

21.5.1.7 repeat_p_t AIRINV::InventoryParserHelper::airport_p (chset_t("0-9A-Z").derived() , 3 , 3)

Airport Parser: repeat_p(3)[chset_p("0-9A-Z")]

Referenced by AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::definition().

21.5.1.8 bounded1_2_p_t AIRINV::InventoryParserHelper::hours_p (uint1_2_p. *derived()*, 0u , 24u)

Hour Parser: limit_d(0u, 24u)[uint2_p]

Referenced by AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::definition().

21.5.1.9 bounded2_p_t AIRINV::InventoryParserHelper::minutes_p (uint2_p. *derived()*, 0u , 59u)

Minute Parser: limit_d(0u, 59u)[uint2_p]

Referenced by AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::definition().

21.5.1.10 bounded2_p_t AIRINV::InventoryParserHelper::seconds_p (uint2_p. *derived()*, 0u , 59u)

Second Parser: limit_d(0u, 59u)[uint2_p]

Referenced by AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::definition().

21.5.1.11 chset_t AIRINV::InventoryParserHelper::cabin_code_p ("A-Z")

Cabin code parser: chset_p("A-Z")

Referenced by AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::definition().

21.5.1.12 chset_t AIRINV::InventoryParserHelper::class_code_p ("A-Z")

Booking class code parser: chset_p("A-Z")

Referenced by AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::definition().

21.5.1.13 chset_t AIRINV::InventoryParserHelper::passenger_type_p ("A-Z")

Passenger type parser: chset_p("A-Z")

21.5.1.14 repeat_p_t AIRINV::InventoryParserHelper::class_code_list_p (chset_t("A-Z").derived() , 1 , 26)

Class Code List Parser: repeat_p(1,26)[chset_p("A-Z")]

Referenced by AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::definition().

21.5.1.15 bounded1_3_p_t AIRINV::InventoryParserHelper::stay_duration_p (uint1_3_p. *derived()*, 0u , 999u)

Stay duration Parser: limit_d(0u, 999u)[uint3_p]

21.5.2 Variable Documentation

21.5.2.1 int1_p_t AIRINV::InventoryParserHelper::int1_p

1-digit-integer parser

Definition at line 823 of file InventoryParserHelper.cpp.

21.5.2.2 uint2_p_t AIRINV::InventoryParserHelper::uint2_p

2-digit-integer parser

Definition at line 826 of file InventoryParserHelper.cpp.

21.5.2.3 uint1_2_p_t AIRINV::InventoryParserHelper::uint1_2_p

Up-to-2-digit-integer parser

Definition at line 829 of file InventoryParserHelper.cpp.

Referenced by AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::definition().

21.5.2.4 uint1_3_p_t AIRINV::InventoryParserHelper::uint1_3_p

Up-to-3-digit-integer parser

Definition at line 832 of file InventoryParserHelper.cpp.

Referenced by AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::definition().

21.5.2.5 uint4_p_t AIRINV::InventoryParserHelper::uint4_p

4-digit-integer parser

Definition at line 835 of file InventoryParserHelper.cpp.

21.5.2.6 uint1_4_p_t AIRINV::InventoryParserHelper::uint1_4_p

Up-to-4-digit-integer parser

Definition at line 838 of file InventoryParserHelper.cpp.

21.5.2.7 int1_p_t AIRINV::InventoryParserHelper::family_code_p

Family code parser

Definition at line 880 of file InventoryParserHelper.cpp.

Referenced by AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::definition().

21.6 AIRINV::ScheduleParserHelper Namespace Reference

Classes

- struct ParserSemanticAction
- struct storeAirlineCode
- struct storeFlightNumber
- struct storeDateRangeStart
- struct storeDateRangeEnd
- struct storeDow
- struct storeLegBoardingPoint
- struct storeLegOffPoint
- struct storeOperatingAirlineCode
- struct storeOperatingFlightNumber
- struct storeBoardingTime
- struct storeOffTime
- struct storeElapsedTime
- struct storeLegCabinCode
- struct storeCapacity
- struct storeSegmentSpecificity
- struct storeSegmentBoardingPoint
- struct storeSegmentOffPoint
- struct storeSegmentCabinCode
- struct storeClasses
- struct storeFamilyCode
- struct storeFRAT5CurveKey
- struct storeFFDisutilityCurveKey
- struct storeFClasses
- struct doEndFlight
- struct FlightPeriodParser

Functions

- repeat_p_t airline_code_p (chset_t("0-9A-Z").derived(), 2, 3)
- bounded1_4_p_t flight_number_p (uint1_4_p.derived(), 0u, 9999u)
- bounded4_p_t year_p (uint4_p.derived(), 2000u, 2099u)
- bounded2_p_t month_p (uint2_p.derived(), 1u, 12u)
- bounded2_p_t day_p (uint2_p.derived(), 1u, 31u)
- repeat_p_t dow_p (chset_t("0-1").derived().derived(), 7, 7)
- repeat_p_t airport_p (chset_t("0-9A-Z").derived(), 3, 3)
- bounded2_p_t hours_p (uint2_p.derived(), 0u, 23u)
- bounded2_p_t minutes_p (uint2_p.derived(), 0u, 59u)
- bounded2_p_t seconds_p (uint2_p.derived(), 0u, 59u)
- chset_t cabin_code_p ("A-Z")
- repeat_p_t key_p (chset_t("0-9A-Z").derived(), 1, 10)
- repeat_p_t class_code_list_p (chset_t("A-Z").derived(), 1, 26)

Variables

- int1_p_t int1_p
- uint2_p_t uint2_p
- uint4_p_t uint4_p
- uint1_4_p_t uint1_4_p
- int1_p_t family_code_p

21.6.1 Function Documentation

21.6.1.1 repeat_p_t AIRINV::ScheduleParserHelper::airline_code_p (chset_t("0-9A-Z").derived() , 2 , 3)

Airline Code Parser: repeat_p(2,3)[chset_p("0-9A-Z")]

Referenced by AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition().

21.6.1.2 bounded1_4_p_t AIRINV::ScheduleParserHelper::flight_number_p (uint1_4_p. *derived()*, 0u , 9999u)

Flight Number Parser: limit_d(0u, 9999u)[uint1_4_p]

Referenced by AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition().

21.6.1.3 bounded4_p_t AIRINV::ScheduleParserHelper::year_p (uint4_p. *derived()*, 2000u , 2099u)

Year Parser: limit_d(2000u, 2099u)[uint4_p]

Referenced by AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition().

21.6.1.4 bounded2_p_t AIRINV::ScheduleParserHelper::month_p (uint2_p. *derived()*, 1u , 12u)

Month Parser: limit_d(1u, 12u)[uint2_p]

Referenced by AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition().

21.6.1.5 bounded2_p_t AIRINV::ScheduleParserHelper::day_p (uint2_p. *derived()*, 1u , 31u)

Day Parser: limit_d(1u, 31u)[uint2_p]

Referenced by AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition().

21.6.1.6 repeat_p_t AIRINV::ScheduleParserHelper::dow_p (chset_t("0-1").derived().derived() , 7 , 7)

DOW (Day-Of-the-Week) Parser: repeat_p(7)[chset_p("0-1")]

Referenced by AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition().

21.6.1.7 repeat_p_t AIRINV::ScheduleParserHelper::airport_p (chset_t(”0-9A-Z”).derived() , 3 , 3)

Airport Parser: repeat_p(3)[chset_p("0-9A-Z")]

Referenced by AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition().

21.6.1.8 bounded2_p_t AIRINV::ScheduleParserHelper::hours_p (uint2_p. *derived(),* 0u , 23u)

Hour Parser: limit_d(0u, 23u)[uint2_p]

Referenced by AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition().

21.6.1.9 bounded2_p_t AIRINV::ScheduleParserHelper::minutes_p (uint2_p. *derived(),* 0u , 59u)

Minute Parser: limit_d(0u, 59u)[uint2_p]

Referenced by AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition().

21.6.1.10 bounded2_p_t AIRINV::ScheduleParserHelper::seconds_p (uint2_p. *derived(),* 0u , 59u)

Second Parser: limit_d(0u, 59u)[uint2_p]

Referenced by AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition().

21.6.1.11 chset_t AIRINV::ScheduleParserHelper::cabin_code_p (”A-Z”)

Cabin Code Parser: chset_p("A-Z")

Referenced by AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition().

21.6.1.12 repeat_p_t AIRINV::ScheduleParserHelper::key_p (chset_t(”0-9A-Z”).derived() , 1 , 10)

Key Parser: repeat_p(1,10)[chset_p("0-9A-Z")]

Referenced by AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition().

21.6.1.13 repeat_p_t AIRINV::ScheduleParserHelper::class_code_list_p (chset_t(”A-Z”).derived() , 1 , 26)

Class Code List Parser: repeat_p(1,26)[chset_p("A-Z")]

Referenced by AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition().

21.6.2 Variable Documentation

21.6.2.1 int1_p_t AIRINV::ScheduleParserHelper::int1_p

1-digit-integer parser

Definition at line 469 of file ScheduleParserHelper.cpp.

Referenced by AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition().

21.6.2.2 uint2_p_t AIRINV::ScheduleParserHelper::uint2_p

2-digit-integer parser

Definition at line 472 of file ScheduleParserHelper.cpp.

21.6.2.3 uint4_p_t AIRINV::ScheduleParserHelper::uint4_p

4-digit-integer parser

Definition at line 475 of file ScheduleParserHelper.cpp.

21.6.2.4 uint1_4_p_t AIRINV::ScheduleParserHelper::uint1_4_p

Up-to-4-digit-integer parser

Definition at line 478 of file ScheduleParserHelper.cpp.

21.6.2.5 int1_p_t AIRINV::ScheduleParserHelper::family_code_p

Family code parser

Definition at line 514 of file ScheduleParserHelper.cpp.

Referenced by AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition().

21.7 stdair Namespace Reference

Forward declarations.

Classes

- struct BomPropertyTree

21.7.1 Detailed Description

Forward declarations.

22 Class Documentation

22.1 AIRINV::AIRINV_Master_Service Class Reference

Interface for the AIRINV Services.

`#include <airinv/AIRINV_Master_Service.hpp>`

Public Member Functions

- AIRINV_Master_Service (const stdair::BasLogParams &, const stdair::BasDBParams &)
- AIRINV_Master_Service (const stdair::BasLogParams &)
- AIRINV_Master_Service (stdair::STDAIR_ServicePtr_T)
- AIRINV_Master_Service (stdair::STDAIR_ServicePtr_T, SEVMGR::SEVMGR_ServicePtr_T)
- void parseAndLoad (const InventoryFilePath &)
- void parseAndLoad (const stdair::ScheduleFilePath &, const stdair::ODFilePath &, const stdair::FRAT5FilePath &, const stdair::FFDisutilityFilePath &, const AIRRAC::YieldFilePath &)
- ~AIRINV_Master_Service ()
- void initSnapshotAndRMEvents (const stdair::Date_T &, const stdair::Date_T &)
- void buildSampleBom ()
- void clonePersistentBom ()
- void buildComplementaryLinks (stdair::BomRoot &)
- void calculateAvailability (stdair::TravelSolutionStruct &)
- bool sell (const std::string &iSegmentDateKey, const stdair::ClassCode_T &, const stdair::PartySize_T &)
- bool sell (const stdair::BookingClassID_T &, const stdair::PartySize_T &)
- bool cancel (const std::string &iSegmentDateKey, const stdair::ClassCode_T &, const stdair::PartySize_T &)
- bool cancel (const stdair::BookingClassID_T &, const stdair::PartySize_T &)
- void takeSnapshots (const stdair::SnapshotStruct &)
- void optimise (const stdair::RMEventStruct &)

- std::string jsonHandler (const stdair::JSONString &) const
- std::string jsonExportFlightDateList (const stdair::AirlineCode_T &iAirlineCode="all", const stdair::FlightNumber_T &iFlightNumber=0) const
- std::string jsonExportFlightDateObjects (const stdair::AirlineCode_T &, const stdair::FlightNumber_T &, const stdair::Date_T &iDepartureDate) const
- std::string list (const stdair::AirlineCode_T &iAirlineCode="all", const stdair::FlightNumber_T &iFlightNumber=0) const
- bool check (const stdair::AirlineCode_T &, const stdair::FlightNumber_T &, const stdair::Date_T &iDepartureDate) const
- std::string csvDisplay () const
- std::string csvDisplay (const stdair::AirlineCode_T &, const stdair::FlightNumber_T &, const stdair::Date_T &iDepartureDate) const

22.1.1 Detailed Description

Interface for the AIRINV Services.

Definition at line 47 of file AIRINV_Master_Service.hpp.

22.1.2 Constructor & Destructor Documentation

22.1.2.1 AIRINV::AIRINV_Master_Service::AIRINV_Master_Service (const stdair::BasLogParams & *iLogParams,* const stdair::BasDBParams & *iDBParams*)

Constructor.

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

A reference on an output stream is given, so that log outputs can be directed onto that stream.

Moreover, database connection parameters are given, so that a session can be created on the corresponding database.

Parameters

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

Definition at line 46 of file AIRINV_Master_Service.cpp.

22.1.2.2 AIRINV::AIRINV_Master_Service::AIRINV_Master_Service (const stdair::BasLogParams & *iLogParams*)

Constructor.

The initSlaveAirinvService() 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.

Parameters

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

Definition at line 68 of file AIRINV_Master_Service.cpp.

22.1.2.3 AIRINV::AIRINV_Master_Service::AIRINV_Master_Service (stdair::STDAIR_ServicePtr_T *ioSTDAIR_Service_ptr*)

Constructor.

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

Moreover, as no reference on any output stream is given, it is assumed that the StdAir log service has already been initialised with the proper log output stream by some other methods in the calling chain (for instance, when the

AIRINV_Master_Service is itself being initialised by another library service such as SIMCRS_Service).

Parameters

stdair::STDAIR_ServicePtr_T	Reference on the STDAIR service.

Definition at line 89 of file AIRINV_Master_Service.cpp.

22.1.2.4 AIRINV::AIRINV_Master_Service::AIRINV_Master_Service (stdair::STDAIR_ServicePtr_T *ioSTDAIR_Service_ptr,* SEVMGR::SEVMGR_ServicePtr_T *ioSEVMGR_Service_ptr*)

Constructor.

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

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

Parameters

stdair::STDAIR_ServicePtr_T	Reference on the STDAIR service.
stdair::SEVMGR_ServicePtr_T	Reference on the SEVMGR service.

Definition at line 106 of file AIRINV_Master_Service.cpp.

22.1.2.5 AIRINV::AIRINV_Master_Service::~AIRINV_Master_Service ( )

Destructor.

Definition at line 127 of file AIRINV_Master_Service.cpp.

22.1.3 Member Function Documentation

22.1.3.1 void AIRINV::AIRINV_Master_Service::parseAndLoad (const InventoryFilePath & *ilInventoryInputFilename*)

Parse the inventory dump and load it into memory.

The CSV file, describing the airline inventory for the simulator, is parsed and instantiated in memory accordingly.

Parameters

const	InventoryFilePath& Filename of the input inventory file.

Definition at line 255 of file AIRINV_Master_Service.cpp.

References clonePersistentBom(), and AIRINV::AIRINV_Service::parseAndLoad().

22.1.3.2 void AIRINV::AIRINV_Master_Service::parseAndLoad (const stdair::ScheduleFilePath & *iScheduleInputFilename,* const stdair::ODFilePath & *iODInputFilename,* const stdair::FRAT5FilePath & *iFRAT5InputFilename,* const stdair::FFDisutilityFilePath & *iFFDisutilityInputFilename,* const AIRRAC::YieldFilePath & *iYieldFilename*)

Parse the schedule and O&D input files, and load them into memory.

The CSV files, describing the airline schedule and the O&Ds for the simulator, are parsed and instantiated in memory accordingly.

Parameters

const	stdair::ScheduleFilePath& Filename of the input schedule file.
const	stdair::ODFilePath& Filename of the input O&D file.
const	stdair::FRAT5FilePath& Filename of the input FRAT5 file.
const	stdair::FFDisutilityFilePath& Filename of the input FF disutility file.
const	AIRRAC::YieldFilePath& Filename of the input yield file.

Definition at line 353 of file AIRINV_Master_Service.cpp.

References buildComplementaryLinks(), clonePersistentBom(), and AIRINV::AIRINV_Service::parseAndLoad().

22.1.3.3 void AIRINV::AIRINV_Master_Service::initSnapshotAndRMEvents (const stdair::Date_T & *iStartDate*, const stdair::Date_T & *iEndDate*)

Initialise the snapshot and RM events for the inventories.

Parameters

const	stdiar::Date_T& Parameters for the start date.
const	stdiar::Date_T& Parameters for the end date.

Definition at line 647 of file AIRINV_Master_Service.cpp.

References AIRINV::AIRINV_Service::initRMEvents().

22.1.3.4 void AIRINV::AIRINV_Master_Service::buildSampleBom ( )

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

The BOM tree is based on two actual inventories (one for BA, another for AF). Each inventory contains one flight. One of those flights has two legs (and therefore three segments).

Definition at line 291 of file AIRINV_Master_Service.cpp.

References buildComplementaryLinks(), AIRINV::AIRINV_Service::buildSampleBom(), and clonePersistentBom().

22.1.3.5 void AIRINV::AIRINV_Master_Service::clonePersistentBom ( )

Clone the persistent BOM object.

Definition at line 415 of file AIRINV_Master_Service.cpp.

References buildComplementaryLinks(), and AIRINV::AIRINV_Service::clonePersistentBom().

Referenced by buildSampleBom(), and parseAndLoad().

22.1.3.6 void AIRINV::AIRINV_Master_Service::buildComplementaryLinks (stdair::BomRoot & *ioBomRoot*)

Build all the complementary links in the given bom root object.

Note

Do nothing for now.

Definition at line 466 of file AIRINV_Master_Service.cpp.

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

22.1.3.7 void AIRINV::AIRINV_Master_Service::calculateAvailability (stdair::TravelSolutionStruct & *ioTravelSolution*)

Compute the availability for the given travel solution.

Definition at line 684 of file AIRINV_Master_Service.cpp.

References AIRINV::AIRINV_Service::calculateAvailability().

22.1.3.8 bool AIRINV::AIRINV_Master_Service::sell (const std::string & *iSegmentDateKey,* const stdair::ClassCode_T & *iClassCode,* const stdair::PartySize_T & *iPartySize*)

Register a booking.

Parameters

const	std::string& Key for the segment on which the sale is made.
const	stdair::ClassCode_T& Class code where the sale is made.
const	stdair::PartySize_T& Party size.

Returns

bool Whether or not the sale was successfull

Definition at line 714 of file AIRINV_Master_Service.cpp.

References AIRINV::AIRINV_Service::sell().

22.1.3.9 bool AIRINV::AIRINV_Master_Service::sell (const stdair::BookingClassID_T & *iClassID,* const stdair::PartySize_T & *iPartySize*)

Register a booking.

Parameters

const	stdair::BookingClassID_T&
const	stdair::PartySize_T& Party size.

Returns

bool Whether or not the sale was successfull

Definition at line 756 of file AIRINV_Master_Service.cpp.

References AIRINV::AIRINV_Service::sell().

22.1.3.10 bool AIRINV::AIRINV_Master_Service::cancel (const std::string & *iSegmentDateKey,* const stdair::ClassCode_T & *iClassCode,* const stdair::PartySize_T & *iPartySize*)

Register a cancellation.

Parameters

const	std::string& Key for the segment on which the cancellation is made.
const	stdair::ClassCode_T& Class code where the sale is made.
const	stdair::PartySize_T& Party size.

Returns

bool Whether or not the cancellation was successfull

Definition at line 797 of file AIRINV_Master_Service.cpp.

References AIRINV::AIRINV_Service::cancel().

22.1.3.11 bool AIRINV::AIRINV_Master_Service::cancel (const stdair::BookingClassID_T & *iClassID,* const stdair::PartySize_T & *iPartySize*)

Register a cancellation.

Parameters

stdair::Booking-ClassID_T&	
const	stdair::PartySize_T& Party size.

Returns

bool Whether or not the cancellation was successfull

Definition at line 839 of file AIRINV_Master_Service.cpp.

References AIRINV::AIRINV_Service::cancel().

22.1.3.12 void AIRINV::AIRINV_Master_Service::takeSnapshots (const stdair::SnapshotStruct & *iSnapshot*)

Take inventory snapshots.

Definition at line 881 of file AIRINV_Master_Service.cpp.

References AIRINV::AIRINV_Service::takeSnapshots().

22.1.3.13 void AIRINV::AIRINV_Master_Service::optimise (const stdair::RMEventStruct & *iRMEvent*)

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

Definition at line 907 of file AIRINV_Master_Service.cpp.

References AIRINV::AIRINV_Service::optimise().

22.1.3.14 std::string AIRINV::AIRINV_Master_Service::jsonHandler (const stdair::JSONString & *IJSONString*) const

Dispatch the JSon command string to the right JSon Service, according to the JSon command type.

Parameters

const	stdair::JSONString& Input string which contained the JSon command string.

Returns

std::string Output string in which the asking objects are logged/dumped in a JSon format.

Definition at line 472 of file AIRINV_Master_Service.cpp.

References AIRINV::AIRINV_Service::jsonHandler().

22.1.3.15 std::string AIRINV::AIRINV_Master_Service::jsonExportFlightDateList (const stdair::AirlineCode_T & *iAirlineCode* = `"all"`, const stdair::FlightNumber_T & *iFlightNumber* = `0`) const

Recursively dump, in the returned string and in JSON format, the flight-date list corresponding to the parameters given as input.

Parameters

const	AirlineCode& Airline for which the flight-dates should be displayed. If set to "all" (default), all the inventories will be displayed.
const	FlightNumber_T& Flight number for which all the departure dates should be displayed. If set to 0 (the default), all the flight numbers will be displayed.

Definition at line 496 of file AIRINV_Master_Service.cpp.

References AIRINV::AIRINV_Service::jsonExportFlightDateList().

22.1.3.16 std::string AIRINV::AIRINV_Master_Service::jsonExportFlightDateObjects (const stdair::AirlineCode_T & *iAirlineCode,* const stdair::FlightNumber_T & *iFlightNumber,* const stdair::Date_T & *iDepartureDate*) const

Recursively dump, in the returned string and in JSON format, the flight-date corresponding to the parameters given as input.

Parameters

const	stdair::AirlineCode_T& Airline code of the flight to dump.
const	stdair::FlightNumber_T& Flight number of the flight to dump.
const	stdair::Date_T& Departure date of the flight to dump.

Returns

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

Definition at line 521 of file AIRINV_Master_Service.cpp.

References AIRINV::AIRINV_Service::jsonExportFlightDateObjects().

22.1.3.17 std::string AIRINV::AIRINV_Master_Service::list (const stdair::AirlineCode_T & *iAirlineCode* = `"all"`, const stdair::FlightNumber_T & *iFlightNumber* = `0`) const

Display the list of flight-dates (contained within the BOM tree) corresponding to the parameters given as input.

Parameters

const	AirlineCode& Airline for which the flight-dates should be displayed. If set to "all" (the default), all the inventories will be displayed.
const	FlightNumber_T& Flight number for which all the departure dates should be displayed. If set to 0 (the default), all the flight numbers will be displayed.

Returns

std::string Output string in which the BOM tree is logged/dumped.

Definition at line 548 of file AIRINV_Master_Service.cpp.

References AIRINV::AIRINV_Service::list().

22.1.3.18 bool AIRINV::AIRINV_Master_Service::check (const stdair::AirlineCode_T & *iAirlineCode,* const stdair::FlightNumber_T & *iFlightNumber,* const stdair::Date_T & *iDepartureDate*) const

Check whether the given flight-date is a valid one.

Parameters

const	stdair::AirlineCode_T& Airline code of the flight to check.
const	stdair::FlightNumber_T& Flight number of the flight to check.
const	stdair::Date_T& Departure date of the flight to check.

Returns

bool Whether or not the given flight date is valid.

Definition at line 573 of file AIRINV_Master_Service.cpp.

References AIRINV::AIRINV_Service::check().

22.1.3.19 std::string AIRINV::AIRINV_Master_Service::csvDisplay ( ) const

Recursively display (dump in the returned string) the objects of the BOM tree.

Returns

std::string Output string in which the BOM tree is logged/dumped.

Definition at line 598 of file AIRINV_Master_Service.cpp.

References AIRINV::AIRINV_Service::csvDisplay().

22.1.3.20 std::string AIRINV::AIRINV_Master_Service::csvDisplay (const stdair::AirlineCode_T & *iAirlineCode,* const stdair::FlightNumber_T & *iFlightNumber,* const stdair::Date_T & *iDepartureDate*) const

Recursively display (dump in the returned string) the flight-date corresponding to the parameters given as input.

Parameters

const	stdair::AirlineCode_T& Airline code of the flight to display.
const	stdair::FlightNumber_T& Flight number of the flight to display.
const	stdair::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 621 of file AIRINV_Master_Service.cpp.

References AIRINV::AIRINV_Service::csvDisplay().

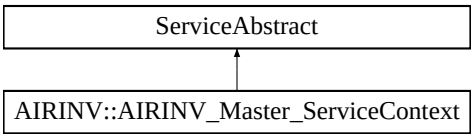
The documentation for this class was generated from the following files:

- airinv/AIRINV_Master_Service.hpp
- airinv/service/AIRINV_Master_Service.cpp

22.2 AIRINV::AIRINV_Master_ServiceContext Class Reference

`#include <airinv/service/AIRINV_Master_ServiceContext.hpp>`

Inheritance diagram for AIRINV::AIRINV_Master_ServiceContext:



Friends

- class AIRINV_Master_Service
- class FacAirinvMasterServiceContext

22.2.1 Detailed Description

Class holding the context of the Airinv services.

Definition at line 28 of file AIRINV_Master_ServiceContext.hpp.

22.2.2 Friends And Related Function Documentation

22.2.2.1 friend class AIRINV_Master_Service [friend]

The AIRINV_Master_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 34 of file AIRINV_Master_ServiceContext.hpp.

22.2.2.2 friend class FacAirinvMasterServiceContext [friend]

Definition at line 35 of file AIRINV_Master_ServiceContext.hpp.

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

- airinv/service/AIRINV_Master_ServiceContext.hpp
- airinv/service/AIRINV_Master_ServiceContext.cpp

22.3 AIRINV::AIRINV_Service Class Reference

Interface for the AIRINV Services.

`#include <airinv/AIRINV_Service.hpp>`

Public Member Functions

- AIRINV_Service (const stdair::BasLogParams &, const stdair::BasDBParams &)
- AIRINV_Service (const stdair::BasLogParams &)
- AIRINV_Service (stdair::STDAIR_ServicePtr_T)
- AIRINV_Service (stdair::STDAIR_ServicePtr_T, SEVMGR::SEVMGR_ServicePtr_T)
- void parseAndLoad (const AIRINV::InventoryFilePath &)
- void parseAndLoad (const stdair::ScheduleFilePath &, const stdair::ODFilePath &, const stdair::FRAT5FilePath &, const stdair::FFDisutilityFilePath &, const AIRRAC::YieldFilePath &)
- ~AIRINV_Service ()
- void buildSampleBom ()
- void clonePersistentBom ()
- void buildComplementaryLinks (stdair::BomRoot &)
- stdair::RMEventList_T initRMEvents (const stdair::Date_T &iStartDate, const stdair::Date_T &iEndDate)
- void calculateAvailability (stdair::TravelSolutionStruct &)
- bool sell (const std::string &iSegmentDateKey, const stdair::ClassCode_T &, const stdair::PartySize_T &)
- bool sell (const stdair::BookingClassID_T &, const stdair::PartySize_T &)
- bool cancel (const std::string &iSegmentDateKey, const stdair::ClassCode_T &, const stdair::PartySize_T &)
- bool cancel (const stdair::BookingClassID_T &, const stdair::PartySize_T &)
- void takeSnapshots (const stdair::AirlineCode_T &, const stdair::DateTime_T &)
- void optimise (const stdair::AirlineCode_T &, const stdair::KeyDescription_T &, const stdair::DateTime_T &)
- std::string jsonHandler (const stdair::JSONString &) const
- std::string jsonExportFlightDateList (const stdair::AirlineCode_T &iAirlineCode="all", const stdair::FlightNumber_T &iFlightNumber=0) const
- std::string jsonExportFlightDateObjects (const stdair::AirlineCode_T &, const stdair::FlightNumber_T &, const stdair::Date_T &iDepartureDate) const

- std::string list (const stdair::AirlineCode_T &iAirlineCode="all", const stdair::FlightNumber_T &iFlightNumber=0) const
- bool check (const stdair::AirlineCode_T &, const stdair::FlightNumber_T &, const stdair::Date_T &iDepartureDate) const
- std::string csvDisplay () const
- std::string csvDisplay (const stdair::AirlineCode_T &, const stdair::FlightNumber_T &, const stdair::Date_T &iDepartureDate) const

22.3.1 Detailed Description

Interface for the AIRINV Services.

Definition at line 40 of file AIRINV_Service.hpp.

22.3.2 Constructor & Destructor Documentation

22.3.2.1 AIRINV::AIRINV_Service::AIRINV_Service (const stdair::BasLogParams & *iLogParams,* const stdair::BasDBParams & *iDBParams*)

Constructor.

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

A reference on an output stream is given, so that log outputs can be directed onto that stream.

Moreover, database connection parameters are given, so that a session can be created on the corresponding database.

Parameters

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

Definition at line 89 of file AIRINV_Service.cpp.

22.3.2.2 AIRINV::AIRINV_Service::AIRINV_Service (const stdair::BasLogParams & *iLogParams*)

Constructor.

The initAirinvService() 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.

Parameters

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

Definition at line 60 of file AIRINV_Service.cpp.

22.3.2.3 AIRINV::AIRINV_Service::AIRINV_Service (stdair::STDAIR_ServicePtr_T *ioSTDAIR_Service_ptr*)

Constructor.

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

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

Parameters

stdair::STDAIR_ServicePtr_T	Reference on the STDAIR service.

Definition at line 120 of file AIRINV_Service.cpp.

22.3.2.4 AIRINV::AIRINV_Service::AIRINV_Service (stdair::STDAIR_ServicePtr_T *ioSTDAIR_Service_ptr,* SEVMGR::SEVMGR_ServicePtr_T *ioSEVMGR_Service_ptr*)

Constructor.

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

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

Parameters

stdair::STDAIR_ServicePtr_T	Reference on the STDAIR service.
stdair::SEVMGR_ServicePtr_T	Reference on the SEVMGR service.

Definition at line 148 of file AIRINV_Service.cpp.

22.3.2.5 AIRINV::AIRINV_Service::~AIRINV_Service ( )

Destructor.

Definition at line 177 of file AIRINV_Service.cpp.

22.3.3 Member Function Documentation

22.3.3.1 void AIRINV::AIRINV_Service::parseAndLoad (const **AIRINV::InventoryFilePath** & *ilnventoryInputFilename*)

Parse the inventory dump and load it into memory.

The CSV file, describing the airline inventory for the simulator, is parsed and instantiated in memory accordingly.

Parameters

const	AIRINV::InventoryFilePath& Filename of the input inventory file.

Definition at line 346 of file AIRINV_Service.cpp.

References buildComplementaryLinks(), AIRINV::InventoryParser::buildInventory(), and clonePersistentBom().

Referenced by AIRINV::AIRINV_Master_Service::parseAndLoad().

22.3.3.2 void AIRINV::AIRINV_Service::parseAndLoad (const stdair::ScheduleFilePath & *iScheduleInputFilename,* const stdair::ODFilePath & *iODInputFilename,* const stdair::FRAT5FilePath & *iFRAT5InputFilename,* const stdair::FFDisutilityFilePath & *iFFDisutilityInputFilename,* const AIRRAC::YieldFilePath & *iYieldFilename*)

Parse the schedule and O&D input files, and load them into memory.

The CSV files, describing the airline schedule and the O&Ds for the simulator, are parsed and instantiated in memory accordingly.

Parameters

const	stdair::ScheduleFilePath& Filename of the input schedule file.
const	stdair::ODFilePath& Filename of the input O&D file.
const	stdair::FRAT5FilePath& Filename of the input FRAT5 file.
const	stdair::FFDisutilityFilePath& Filename of the input FF disutility file.
const	AIRRAC::YieldFilePath& Filename of the input yield file.

Definition at line 404 of file AIRINV_Service.cpp.

References buildComplementaryLinks(), clonePersistentBom(), AIRINV::ScheduleParser::generateInventories(), AIRINV::FRAT5Parser::parse(), and AIRINV::FFDisutilityParser::parse().

22.3.3.3 void AIRINV::AIRINV_Service::buildSampleBom ( )

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

The BOM tree is based on two actual inventories (one for BA, another for AF). Each inventory contains one flight. One of those flights has two legs (and therefore three segments).

Definition at line 487 of file AIRINV_Service.cpp.

References buildComplementaryLinks(), and clonePersistentBom().

Referenced by AIRINV::AIRINV_Master_Service::buildSampleBom().

22.3.3.4 void AIRINV::AIRINV_Service::clonePersistentBom ( )

Clone the persistent BOM object.

Definition at line 570 of file AIRINV_Service.cpp.

References buildComplementaryLinks().

Referenced by buildSampleBom(), AIRINV::AIRINV_Master_Service::clonePersistentBom(), and parseAndLoad().

22.3.3.5 void AIRINV::AIRINV_Service::buildComplementaryLinks (stdair::BomRoot & *ioBomRoot*)

Build all the complementary links in the given bom root object. For instance, build the links between leg and segment date (as well as leg and segment cabin).

Definition at line 608 of file AIRINV_Service.cpp.

References AIRINV::InventoryManager::buildSimilarSegmentCabinSets(), AIRINV::InventoryManager::createDirectAccesses(), AIRINV::InventoryManager::initialiseListsOfUsablePolicies(), AIRINV::InventoryManager::initialiseYieldBasedNestingStructures(), and AIRINV::InventoryManager::setDefaultBidPriceVector().

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

22.3.3.6 stdair::RMEventList_T AIRINV::AIRINV_Service::initRMEvents (const stdair::Date_T & *iStartDate,* const stdair::Date_T & *iEndDate*)

Initialise the RM events for the inventory.

Parameters

const	stdair::Date_T& Parameters for the start date.
const	stdair::Date_T& Parameters for the end date.

Definition at line 920 of file AIRINV_Service.cpp.

Referenced by AIRINV::AIRINV_Master_Service::initSnapshotAndRMEvents().

22.3.3.7 void AIRINV::AIRINV_Service::calculateAvailability (stdair::TravelSolutionStruct & *ioTravelSolution*)

Compute the availability for the given travel solution.

Definition at line 952 of file AIRINV_Service.cpp.

Referenced by AIRINV::AIRINV_Master_Service::calculateAvailability().

22.3.3.8 bool AIRINV::AIRINV_Service::sell (const std::string & *iSegmentDateKey,* const stdair::ClassCode_T & *iClassCode,* const stdair::PartySize_T & *iPartySize*)

Register a booking.

Parameters

const	std::string& Key for the segment on which the sale is made
const	stdair::ClassCode_T& Class code where the sale is made
const	stdair::PartySize_T& Party size

Returns

bool Whether or not the sale was successfull

Definition at line 978 of file AIRINV_Service.cpp.

Referenced by AIRINV::AIRINV_Master_Service::sell(), and sell().

22.3.3.9 bool AIRINV::AIRINV_Service::sell (const stdair::BookingClassID_T & *iClassID,* const stdair::PartySize_T & *iPartySize*)

Register a booking.

Parameters

const	stdair::BookingClassID_T&
const	stdair::PartySize_T& Party size

Returns

bool Whether or not the sale was successfull

Definition at line 1019 of file AIRINV_Service.cpp.

References sell().

22.3.3.10 bool AIRINV::AIRINV_Service::cancel (const std::string & *iSegmentDateKey,* const stdair::ClassCode_T & *iClassCode,* const stdair::PartySize_T & *iPartySize*)

Register a cancellation.

Parameters

const	std::string& Key for the segment on which the cancellation is made
const	stdair::ClassCode_T& Class code where the sale is made
const	stdair::PartySize_T& Party size

Returns

bool Whether or not the cancellation was successfull

Definition at line 1036 of file AIRINV_Service.cpp.

Referenced by AIRINV::AIRINV_Master_Service::cancel(), and cancel().

22.3.3.11 bool AIRINV::AIRINV_Service::cancel (const stdair::BookingClassID_T & *iClassID,* const stdair::PartySize_T & *iPartySize*)

Register a cancellation.

Parameters

const	stdair::BookingClassID_T&
const	stdair::PartySize_T& Party size

Returns

bool Whether or not the cancellation was successfull

Definition at line 1080 of file AIRINV_Service.cpp.

References cancel().

22.3.3.12 void AIRINV::AIRINV_Service::takeSnapshots (const stdair::AirlineCode_T & *iAirlineCode,* const stdair::DateTime_T & *iSnapshotTime*)

Take inventory snapshots.

Definition at line 1097 of file AIRINV_Service.cpp.

Referenced by AIRINV::AIRINV_Master_Service::takeSnapshots().

22.3.3.13 void AIRINV::AIRINV_Service::optimise (const stdair::AirlineCode_T & *iAirlineCode,* const stdair::KeyDescription_T & *iFDDescription,* const stdair::DateTime_T & *iRMEventTime*)

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

Definition at line 1124 of file AIRINV_Service.cpp.

Referenced by AIRINV::AIRINV_Master_Service::optimise().

22.3.3.14 std::string AIRINV::AIRINV_Service::jsonHandler (const stdair::JSONString & *iJSONString*) const

Dispatch the JSon command string to the right JSon Service, according to the JSon command type.

Parameters

const	stdair::JSONString& Input string which contained the JSon command string.

Returns

std::string Output string in which the asking objects are logged/dumped in a JSon format.

Definition at line 646 of file AIRINV_Service.cpp.

References csvDisplay(), jsonExportFlightDateList(), jsonExportFlightDateObjects(), and list().

Referenced by AIRINV::AIRINV_Master_Service::jsonHandler().

22.3.3.15 std::string AIRINV::AIRINV_Service::jsonExportFlightDateList (const stdair::AirlineCode_T & *iAirlineCode* = `"all"`, const stdair::FlightNumber_T & *iFlightNumber* = `0`) const

Recursively dump, in the returned string and in JSON format, the flight-date list corresponding to the parameters given as input.

Parameters

const	AirlineCode& Airline for which the flight-dates should be displayed. If set to "all" (default), all the inventories will be displayed.
const	FlightNumber_T& Flight number for which all the departure dates should be displayed. If set to 0 (the default), all the flight numbers will be displayed.

Definition at line 778 of file AIRINV_Service.cpp.

Referenced by AIRINV::AIRINV_Master_Service::jsonExportFlightDateList(), and jsonHandler().

22.3.3.16 std::string AIRINV::AIRINV_Service::jsonExportFlightDateObjects (const stdair::AirlineCode_T & *iAirlineCode,* const stdair::FlightNumber_T & *iFlightNumber,* const stdair::Date_T & *iDepartureDate*) const

Recursively dump, in the returned string and in JSON format, the flight-date corresponding to the parameters given as input.

Parameters

const	stdair::AirlineCode_T& Airline code of the flight to dump.
const	stdair::FlightNumber_T& Flight number of the flight to dump.
const	stdair::Date_T& Departure date of the flight to dump.

Returns

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

Definition at line 801 of file AIRINV_Service.cpp.

Referenced by AIRINV::AIRINV_Master_Service::jsonExportFlightDateObjects(), and jsonHandler().

22.3.3.17 std::string AIRINV::AIRINV_Service::list (const stdair::AirlineCode_T & *iAirlineCode* = `"all"`, const stdair::FlightNumber_T & *iFlightNumber* = `0`) const

Display the list of flight-dates (contained within the BOM tree) corresponding to the parameters given as input.

Parameters

const	AirlineCode& Airline for which the flight-dates should be displayed. If set to "all" (the default), all the inventories will be displayed.
const	FlightNumber_T& Flight number for which all the departure dates should be displayed. If set to 0 (the default), all the flight numbers will be displayed.

Returns

std::string Output string in which the BOM tree is logged/dumped.

Definition at line 826 of file AIRINV_Service.cpp.

Referenced by jsonHandler(), and AIRINV::AIRINV_Master_Service::list().

22.3.3.18 bool AIRINV::AIRINV_Service::check (const stdair::AirlineCode_T & *iAirlineCode,* const stdair::FlightNumber_T & *iFlightNumber,* const stdair::Date_T & *iDepartureDate*) const

Check whether the given flight-date is a valid one.

Parameters

const	stdair::AirlineCode_T& Airline code of the flight to check.
const	stdair::FlightNumber_T& Flight number of the flight to check.
const	stdair::Date_T& Departure date of the flight to check.

Returns

bool Whether or not the given flight date is valid.

Definition at line 850 of file AIRINV_Service.cpp.

Referenced by AIRINV::AIRINV_Master_Service::check().

22.3.3.19 std::string AIRINV::AIRINV_Service::csvDisplay ( ) const

Recursively display (dump in the returned string) the objects of the BOM tree.

Returns

std::string Output string in which the BOM tree is logged/dumped.

Definition at line 874 of file AIRINV_Service.cpp.

Referenced by AIRINV::AIRINV_Master_Service::csvDisplay(), and jsonHandler().

22.3.3.20 std::string AIRINV::AIRINV_Service::csvDisplay (const stdair::AirlineCode_T & *iAirlineCode,* const stdair::FlightNumber_T & *iFlightNumber,* const stdair::Date_T & *iDepartureDate*) const

Recursively display (dump in the returned string) the flight-date corresponding to the parameters given as input.

Parameters

const	stdair::AirlineCode_T& Airline code of the flight to display
const	stdair::FlightNumber_T& Flight number of the flight to display.
const	stdair::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 896 of file AIRINV_Service.cpp.

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

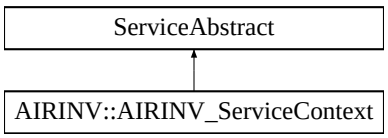
- airinv/AIRINV_Service.hpp
- airinv/service/AIRINV_Service.cpp

22.4 AIRINV::AIRINV_ServiceContext Class Reference

Class holding the context of the AirInv services.

`#include <airinv/service/AIRINV_ServiceContext.hpp>`

Inheritance diagram for AIRINV::AIRINV_ServiceContext:



Friends

- class AIRINV_Service
- class FacAirinvServiceContext

22.4.1 Detailed Description

Class holding the context of the AirInv services.

Definition at line 28 of file AIRINV_ServiceContext.hpp.

22.4.2 Friends And Related Function Documentation

22.4.2.1 friend class AIRINV_Service [friend]

The AIRINV_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 34 of file AIRINV_ServiceContext.hpp.

22.4.2.2 friend class FacAirinvServiceContext [friend]

Definition at line 35 of file AIRINV_ServiceContext.hpp.

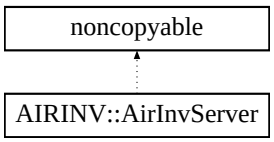
The documentation for this class was generated from the following files:

- airinv/service/AIRINV_ServiceContext.hpp
- airinv/service/AIRINV_ServiceContext.cpp

22.5 AIRINV::AirInvServer Class Reference

`#include <airinv/server/AirInvServer.hpp>`

Inheritance diagram for AIRINV::AirInvServer:



Public Member Functions

- AirInvServer (const std::string &address, const std::string &port, const stdair::AirlineCode_T &iAirlineCode, std::size_t thread_pool_size)
- ~AirInvServer ()
- void run ()
- void stop ()

22.5.1 Detailed Description

The top-level class of the AirInv server.

Definition at line 23 of file AirInvServer.hpp.

22.5.2 Constructor & Destructor Documentation

22.5.2.1 AIRINV::AirInvServer::AirInvServer (const std::string & *address*, const std::string & *port*, const stdair::AirlineCode_T & *iAirlineCode*, std::size_t *thread_pool_size*)

Constructor.

Construct the server to listen on the specified TCP address and port, and serve up files from the given directory.

Definition at line 20 of file AirInvServer_ASIO.cpp.

22.5.2.2 AIRINV::AirInvServer::~AirInvServer ( )

Destructor.

Definition at line 46 of file AirInvServer_ASIO.cpp.

22.5.3 Member Function Documentation

22.5.3.1 void AIRINV::AirInvServer::run ( )

Run the server's io_service loop.

Definition at line 50 of file AirInvServer_ASIO.cpp.

Referenced by main().

22.5.3.2 void AIRINV::AirInvServer::stop ( )

Stop the server.

Definition at line 69 of file AirInvServer_ASIO.cpp.

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

- airinv/server/AirInvServer.hpp
- airinv/server/AirInvServer_ASIO.cpp

22.6 AIRINV::BomAbstract Class Reference

`#include <airinv/bom/BomAbstract.hpp>`

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 std::string describeKey () const =0
- virtual std::string describeShortKey () const =0

Protected Member Functions

- BomAbstract ()
- BomAbstract (const BomAbstract &)
- virtual ~BomAbstract ()

Friends

- class FacBomAbstract

22.6.1 Detailed Description

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

Definition at line 14 of file BomAbstract.hpp.

22.6.2 Constructor & Destructor Documentation

22.6.2.1 AIRINV::BomAbstract::BomAbstract ( ) [inline], [protected]

Protected Default Constructor to ensure this class is abtract.

Definition at line 40 of file BomAbstract.hpp.

22.6.2.2 AIRINV::BomAbstract::BomAbstract (const BomAbstract &) [inline], [protected]

Definition at line 41 of file BomAbstract.hpp.

22.6.2.3 virtual AIRINV::BomAbstract::~BomAbstract ( ) [inline], [protected], [virtual]

Destructor.

Definition at line 44 of file BomAbstract.hpp.

22.6.3 Member Function Documentation

22.6.3.1 virtual void AIRINV::BomAbstract::toStream (std::ostream & *ioOut*) const [pure virtual]

Dump a Business Object into an output stream.

Parameters

ostream&	the output stream.

22.6.3.2 virtual void AIRINV::BomAbstract::fromStream (std::istream & *ioIn*) [pure virtual]

Read a Business Object from an input stream.

Parameters

istream&	the input stream.

Referenced by operator>>().

22.6.3.3 virtual std::string AIRINV::BomAbstract::toString ( ) const [pure virtual]

Get the serialised version of the Business Object.

22.6.3.4 virtual std::string AIRINV::BomAbstract::describeKey ( ) const [pure virtual]

Get a string describing the whole key (differentiating two objects at any level).

22.6.3.5 virtual std::string AIRINV::BomAbstract::describeShortKey ( ) const [pure virtual]

Get a string describing the short key (differentiating two objects at the same level).

22.6.4 Friends And Related Function Documentation

22.6.4.1 friend class FacBomAbstract [friend]

Definition at line 15 of file BomAbstract.hpp.

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

- airinv/bom/BomAbstract.hpp

22.7 stdair::BomPropertyTree Struct Reference

`#include <airinv/server/BomPropertyTree.hpp>`

Public Member Functions

- void load (const std::string &iBomTree)
- std::string save () const

Public Attributes

- stdair::AirlineCode_T _airlineCode
- stdair::FlightNumber_T _flightNumber
- stdair::Date_T _departureDate
- std::set< stdair::AirportCode_T > _airportCodeList

22.7.1 Detailed Description

Structure representing a list of airports.

Definition at line 19 of file BomPropertyTree.hpp.

22.7.2 Member Function Documentation

22.7.2.1 void stdair::BomPropertyTree::load (const std::string & *iBomTree*)

Update the current BOM tree (∗this) with the parsed stream, which is JSON formatted.

Definition at line 17 of file BomPropertyTree.cpp.

References _airlineCode, _departureDate, and _flightNumber.

22.7.2.2 std::string stdair::BomPropertyTree::save ( ) const

Dump the BOM tree (∗this) into the stream with a JSON format.

Definition at line 60 of file BomPropertyTree.cpp.

References _airlineCode, _airportCodeList, _departureDate, and _flightNumber.

22.7.3 Member Data Documentation

22.7.3.1 stdair::AirlineCode_T stdair::BomPropertyTree::_airlineCode

Airline code.

Definition at line 33 of file BomPropertyTree.hpp.

Referenced by load(), and save().

22.7.3.2 stdair::FlightNumber_T stdair::BomPropertyTree::_flightNumber

Flight number.

Definition at line 36 of file BomPropertyTree.hpp.

Referenced by load(), and save().

22.7.3.3 stdair::Date_T stdair::BomPropertyTree::_departureDate

Departure date.

Definition at line 39 of file BomPropertyTree.hpp.

Referenced by load(), and save().

22.7.3.4 std::set<stdair::AirportCode_T> stdair::BomPropertyTree::_airportCodeList

Just to have a list, for now.

Definition at line 42 of file BomPropertyTree.hpp.

Referenced by save().

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

- airinv/server/BomPropertyTree.hpp
- airinv/server/BomPropertyTree.cpp

22.8 AIRINV::BomRootHelper Class Reference

`#include <airinv/bom/BomRootHelper.hpp>`

Static Public Member Functions

- static void fillFromRouting (const stdair::BomRoot &)

22.8.1 Detailed Description

Class representing the actual business functions for an airline bom root.

Definition at line 16 of file BomRootHelper.hpp.

22.8.2 Member Function Documentation

22.8.2.1 void AIRINV::BomRootHelper::fillFromRouting (const stdair::BomRoot & *iBomRoot*) [static]

Fill the attributes derived from the routing legs (e.g., board and off dates).

Definition at line 16 of file BomRootHelper.cpp.

Referenced by AIRINV::InventoryManager::createDirectAccesses().

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

- airinv/bom/BomRootHelper.hpp
- airinv/bom/BomRootHelper.cpp

22.9 AIRINV::BookingClassHelper Class Reference

`#include <airinv/bom/BookingClassHelper.hpp>`

22.9.1 Detailed Description

Class representing the actual business functions for an airline booking class.

Definition at line 19 of file BookingClassHelper.hpp.

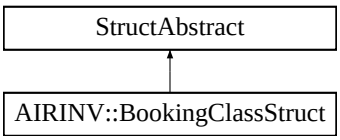
The documentation for this class was generated from the following file:

- airinv/bom/BookingClassHelper.hpp

22.10 AIRINV::BookingClassStruct Struct Reference

`#include <airinv/bom/BookingClassStruct.hpp>`

Inheritance diagram for AIRINV::BookingClassStruct:



Public Member Functions

- stdair::ClassCode_T getFullSubclassCode () const
- void fill (stdair::BookingClass &) const
- const std::string describe () const
- BookingClassStruct ()

Public Attributes

- stdair::ClassCode_T _classCode
- stdair::SubclassCode_T _subclassCode
- stdair::ClassCode_T _parentClassCode
- stdair::SubclassCode_T _parentSubclassCode
- stdair::AuthorizationLevel_T _cumulatedProtection
- stdair::AuthorizationLevel_T _protection
- stdair::NbOfSeats_T _nego
- stdair::OverbookingRate_T _noShowPercentage
- stdair::OverbookingRate_T _overbookingPercentage
- stdair::NbOfBookings_T _nbOfBookings
- stdair::NbOfBookings_T _nbOfGroupBookings
- stdair::NbOfBookings_T _nbOfPendingGroupBookings
- stdair::NbOfBookings_T _nbOfStaffBookings
- stdair::NbOfBookings_T _nbOfWLBookings
- stdair::NbOfBookings_T _etb
- stdair::Availability_T _netClassAvailability
- stdair::Availability_T _segmentAvailability
- stdair::Availability_T _netRevenueAvailability

22.10.1 Detailed Description

Utility Structure for the parsing of BookingClass structures.

Definition at line 24 of file BookingClassStruct.hpp.

22.10.2 Constructor & Destructor Documentation

22.10.2.1 AIRINV::BookingClassStruct::BookingClassStruct ( )

Default Constructor.

Definition at line 16 of file BookingClassStruct.cpp.

22.10.3 Member Function Documentation

22.10.3.1 stdair::ClassCode_T AIRINV::BookingClassStruct::getFullSubclassCode ( ) const

Returns the concatenation of the class and subclass codes.

Definition at line 20 of file BookingClassStruct.cpp.

References _classCode, and _subclassCode.

22.10.3.2 void AIRINV::BookingClassStruct::fill (stdair::BookingClass & *ioBookingClass*) const

Fill the BookingClass objects with the attributes of the BookingClassStruct.

Definition at line 44 of file BookingClassStruct.cpp.

22.10.3.3 const std::string AIRINV::BookingClassStruct::describe ( ) const

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

Definition at line 27 of file BookingClassStruct.cpp.

References _classCode, _cumulatedProtection, _etb, _nbOfBookings, _nbOfGroupBookings, _nbOfPendingGroupBookings, _nbOfStaffBookings, _nbOfWLBookings, _nego, _netClassAvailability, _netRevenueAvailability, _noShowPercentage, _overbookingPercentage, _parentClassCode, _parentSubclassCode, _protection, _segmentAvailability, and _subclassCode.

Referenced by AIRINV::FareFamilyStruct::describe().

22.10.4 Member Data Documentation

22.10.4.1 stdair::ClassCode_T AIRINV::BookingClassStruct::_classCode

Definition at line 26 of file BookingClassStruct.hpp.

Referenced by describe(), getFullSubclassCode(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), and AIRINV::InventoryParserHelper::storeFClasses::operator()().

22.10.4.2 stdair::SubclassCode_T AIRINV::BookingClassStruct::_subclassCode

Definition at line 27 of file BookingClassStruct.hpp.

Referenced by describe(), getFullSubclassCode(), and AIRINV::InventoryParserHelper::storeSubclassCode::operator()().

22.10.4.3 stdair::ClassCode_T AIRINV::BookingClassStruct::_parentClassCode

Definition at line 28 of file BookingClassStruct.hpp.

Referenced by describe(), and AIRINV::InventoryParserHelper::storeParentClassCode::operator()().

22.10.4.4 stdair::SubclassCode_T AIRINV::BookingClassStruct::_parentSubclassCode

Definition at line 29 of file BookingClassStruct.hpp.

Referenced by describe(), and AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()().

22.10.4.5 stdair::AuthorizationLevel_T AIRINV::BookingClassStruct::_cumulatedProtection

Definition at line 30 of file BookingClassStruct.hpp.

Referenced by describe(), and AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()().

22.10.4.6 stdair::AuthorizationLevel_T AIRINV::BookingClassStruct::_protection

Definition at line 31 of file BookingClassStruct.hpp.

Referenced by describe(), and AIRINV::InventoryParserHelper::storeProtection::operator()().

22.10.4.7 stdair::NbOfSeats_T AIRINV::BookingClassStruct::_nego

Definition at line 32 of file BookingClassStruct.hpp.

Referenced by describe(), and AIRINV::InventoryParserHelper::storeNego::operator()().

22.10.4.8 stdair::OverbookingRate_T AIRINV::BookingClassStruct::_noShowPercentage

Definition at line 33 of file BookingClassStruct.hpp.

Referenced by describe(), and AIRINV::InventoryParserHelper::storeNoShow::operator()().

22.10.4.9 stdair::OverbookingRate_T AIRINV::BookingClassStruct::_overbookingPercentage

Definition at line 34 of file BookingClassStruct.hpp.

Referenced by describe(), and AIRINV::InventoryParserHelper::storeOverbooking::operator()().

22.10.4.10 stdair::NbOfBookings_T AIRINV::BookingClassStruct::_nbOfBookings

Definition at line 35 of file BookingClassStruct.hpp.

Referenced by describe(), and AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()().

22.10.4.11 stdair::NbOfBookings_T AIRINV::BookingClassStruct::_nbOfGroupBookings

Definition at line 36 of file BookingClassStruct.hpp.

Referenced by describe(), and AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()().

22.10.4.12 stdair::NbOfBookings_T AIRINV::BookingClassStruct::_nbOfPendingGroupBookings

Definition at line 37 of file BookingClassStruct.hpp.

Referenced by describe(), and AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()().

22.10.4.13 stdair::NbOfBookings_T AIRINV::BookingClassStruct::_nbOfStaffBookings

Definition at line 38 of file BookingClassStruct.hpp.

Referenced by describe(), and AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()().

22.10.4.14 stdair::NbOfBookings_T AIRINV::BookingClassStruct::_nbOfWLBookings

Definition at line 39 of file BookingClassStruct.hpp.

Referenced by describe(), and AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()().

22.10.4.15 stdair::NbOfBookings_T AIRINV::BookingClassStruct::_etb

Definition at line 40 of file BookingClassStruct.hpp.

Referenced by describe(), and AIRINV::InventoryParserHelper::storeClassETB::operator()().

22.10.4.16 stdair::Availability_T AIRINV::BookingClassStruct::_netClassAvailability

Definition at line 41 of file BookingClassStruct.hpp.

Referenced by describe(), and AIRINV::InventoryParserHelper::storeClassAvailability::operator()().

22.10.4.17 stdair::Availability_T AIRINV::BookingClassStruct::_segmentAvailability

Definition at line 42 of file BookingClassStruct.hpp.

Referenced by describe(), and AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()().

22.10.4.18 stdair::Availability_T AIRINV::BookingClassStruct::_netRevenueAvailability

Definition at line 43 of file BookingClassStruct.hpp.

Referenced by describe(), and AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()().

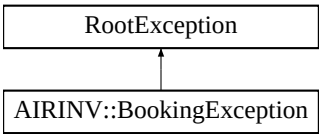
The documentation for this struct was generated from the following files:

- airinv/bom/BookingClassStruct.hpp
- airinv/bom/BookingClassStruct.cpp

22.11 AIRINV::BookingException Class Reference

`#include <airinv/AIRINV_Types.hpp>`

Inheritance diagram for AIRINV::BookingException:



22.11.1 Detailed Description

Specific exception related to bookings made against the inventory.

Definition at line 166 of file AIRINV_Types.hpp.

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

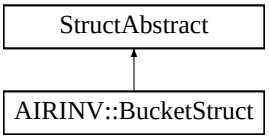
- airinv/AIRINV_Types.hpp

22.12 AIRINV::BucketStruct Struct Reference

Utility Structure for the parsing of Bucket structures.

`#include <airinv/bom/BucketStruct.hpp>`

Inheritance diagram for AIRINV::BucketStruct:



Public Member Functions

- void fill (stdair::Bucket &) const
- const std::string describe () const
- BucketStruct ()

Public Attributes

- stdair::Yield_T _yieldRangeUpperValue
- stdair::CabinCapacity_T _availability
- stdair::NbOfSeats_T _nbOfSeats
- stdair::SeatIndex_T _seatIndex

22.12.1 Detailed Description

Utility Structure for the parsing of Bucket structures.

Definition at line 26 of file BucketStruct.hpp.

22.12.2 Constructor & Destructor Documentation

22.12.2.1 AIRINV::BucketStruct::BucketStruct ( )

Default Constructor.

Definition at line 16 of file BucketStruct.cpp.

22.12.3 Member Function Documentation

22.12.3.1 void AIRINV::BucketStruct::fill (stdair::Bucket & *ioBucket*) const

Fill the Bucket objects with the attributes of the BucketStruct.

Definition at line 29 of file BucketStruct.cpp.

References _availability, _nbOfSeats, and _yieldRangeUpperValue.

22.12.3.2 const std::string AIRINV::BucketStruct::describe ( ) const

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

Definition at line 20 of file BucketStruct.cpp.

References _availability, _nbOfSeats, _seatIndex, and _yieldRangeUpperValue.

Referenced by AIRINV::LegCabinStruct::describe().

22.12.4 Member Data Documentation

22.12.4.1 stdair::Yield_T AIRINV::BucketStruct::_yieldRangeUpperValue

Definition at line 28 of file BucketStruct.hpp.

Referenced by describe(), fill(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), and AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()().

22.12.4.2 stdair::CabinCapacity_T AIRINV::BucketStruct::_availability

Definition at line 29 of file BucketStruct.hpp.

Referenced by describe(), fill(), and AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()().

22.12.4.3 stdair::NbOfSeats_T AIRINV::BucketStruct::_nbOfSeats

Definition at line 30 of file BucketStruct.hpp.

Referenced by describe(), and fill().

22.12.4.4 stdair::SeatIndex_T AIRINV::BucketStruct::_seatIndex

Definition at line 31 of file BucketStruct.hpp.

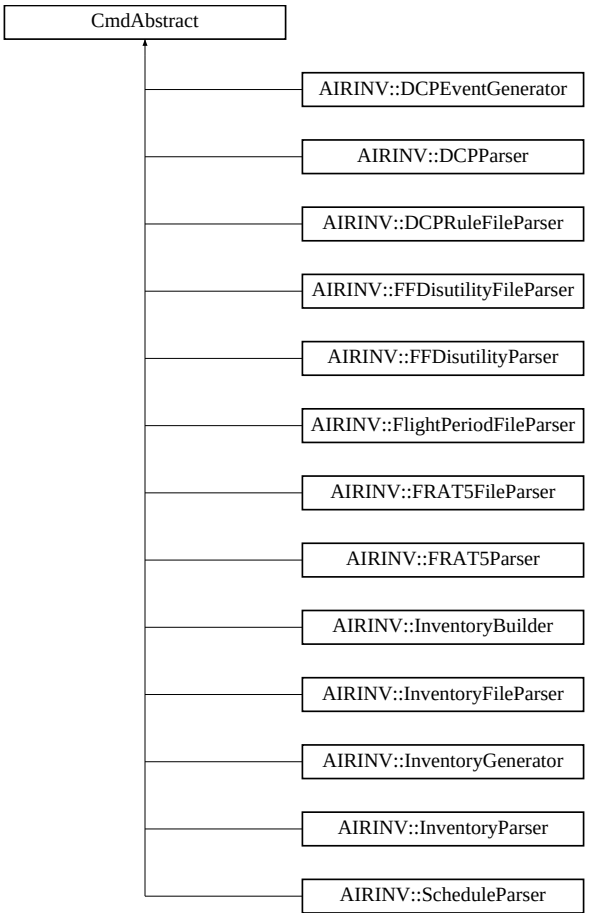
Referenced by describe(), and AIRINV::InventoryParserHelper::storeSeatIndex::operator()().

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

- airinv/bom/BucketStruct.hpp
- airinv/bom/BucketStruct.cpp

22.13 CmdAbstract Class Reference

Inheritance diagram for CmdAbstract:



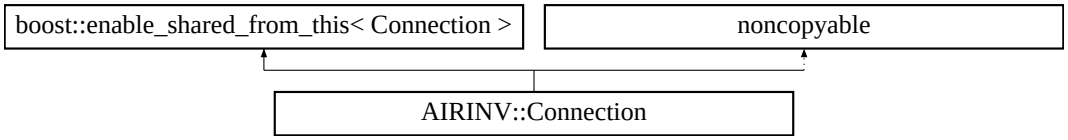
The documentation for this class was generated from the following file:

- airinv/command/FFDisutilityParser.hpp

22.14 AIRINV::Connection Class Reference

`#include <airinv/server/Connection.hpp>`

Inheritance diagram for AIRINV::Connection:



Public Member Functions

- Connection (boost::asio::io_service &, RequestHandler &)
- boost::asio::ip::tcp::socket & socket ()
- void start ()

22.14.1 Detailed Description

Represents a single connection from a client.

Definition at line 25 of file Connection.hpp.

22.14.2 Constructor & Destructor Documentation

22.14.2.1 AIRINV::Connection::Connection (boost::asio::io_service & *ioService,* RequestHandler & *ioHandler*)

Constructor.

Construct a connection with the given io_service.

Definition at line 16 of file Connection.cpp.

22.14.3 Member Function Documentation

22.14.3.1 boost::asio::ip::tcp::socket & AIRINV::Connection::socket ( )

Get the socket associated with the connection.

Definition at line 22 of file Connection.cpp.

22.14.3.2 void AIRINV::Connection::start ( )

Start the first asynchronous operation for the connection.

Definition at line 27 of file Connection.cpp.

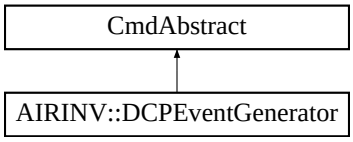
The documentation for this class was generated from the following files:

- airinv/server/Connection.hpp
- airinv/server/Connection.cpp

22.15 AIRINV::DCPEventGenerator Class Reference

`#include <airinv/command/vault/DCPEventGenerator.hpp>`

Inheritance diagram for AIRINV::DCPEventGenerator:



Friends

- class DCPFileParser
- struct DCPParserHelper::doEndDCP
- class DCPParser

22.15.1 Detailed Description

Class handling the generation / instantiation of the DCP BOM.

Definition at line 27 of file DCPEventGenerator.hpp.

22.15.2 Friends And Related Function Documentation

22.15.2.1 friend class DCPFileParser [friend]

Definition at line 31 of file DCPEventGenerator.hpp.

22.15.2.2 friend struct DCPParserHelper::doEndDCP [friend]

Definition at line 32 of file DCPEventGenerator.hpp.

22.15.2.3 friend class DCPParser [friend]

Definition at line 33 of file DCPEventGenerator.hpp.

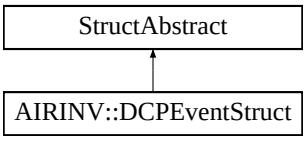
The documentation for this class was generated from the following files:

- airinv/command/vault/DCPEventGenerator.hpp
- airinv/command/vault/DCPEventGenerator.cpp

22.16 AIRINV::DCPEventStruct Struct Reference

`#include <airinv/bom/DCPEventStruct.hpp>`

Inheritance diagram for AIRINV::DCPEventStruct:



Public Member Functions

- DCPEventStruct ()
- stdair::Date_T getDate () const
- stdair::Duration_T getTime () const
- const std::string describe () const

- const unsigned int getAirlineListSize () const
- const unsigned int getClassCodeListSize () const
- const stdair::AirlineCode_T & getFirstAirlineCode () const
- void beginAirline ()
- bool hasNotReachedEndAirline () const
- stdair::AirlineCode_T getCurrentAirlineCode () const
- void iterateAirline ()
- const std::string & getFirstClassCode () const
- void beginClassCode ()
- bool hasNotReachedEndClassCode () const
- std::string getCurrentClassCode () const
- void iterateClassCode ()

Public Attributes

- stdair::year_t _itYear
- stdair::month_t _itMonth
- stdair::day_t _itDay
- stdair::hour_t _itHours
- stdair::minute_t _itMinutes
- stdair::second_t _itSeconds
- stdair::AirlineCodeList_T::iterator _itCurrentAirlineCode
- stdair::ClassList_StringList_T::iterator _itCurrentClassCode
- stdair::AirportCode_T _origin
- stdair::AirportCode_T _destination
- stdair::Date_T _dateRangeStart
- stdair::Date_T _dateRangeEnd
- stdair::Duration_T _timeRangeStart
- stdair::Duration_T _timeRangeEnd
- stdair::CabinCode_T _cabinCode
- stdair::CityCode_T _pos
- stdair::ChannelLabel_T _channel
- stdair::DayDuration_T _advancePurchase
- stdair::SaturdayStay_T _saturdayStay
- stdair::ChangeFees_T _changeFees
- stdair::NonRefundable_T _nonRefundable
- stdair::DayDuration_T _minimumStay
- stdair::PriceValue_T _DCP
- stdair::AirlineCode_T _airlineCode
- stdair::ClassCode_T _classCode
- stdair::AirlineCodeList_T _airlineCodeList
- stdair::ClassList_StringList_T _classCodeList

22.16.1 Detailed Description

Utility Structure for the parsing of Flight-Period structures.

Definition at line 21 of file DCPEventStruct.hpp.

22.16.2 Constructor & Destructor Documentation

22.16.2.1 AIRINV::DCPEventStruct::DCPEventStruct ( )

Default constructor.

Definition at line 18 of file DCPEventStruct.cpp.

22.16.3 Member Function Documentation

22.16.3.1 stdair::Date_T AIRINV::DCPEventStruct::getDate ( ) const

Get the date from the staging details.

Definition at line 38 of file DCPEventStruct.cpp.

References _itDay, _itMonth, and _itYear.

22.16.3.2 stdair::Duration_T AIRINV::DCPEventStruct::getTime ( ) const

Get the time from the staging details.

Definition at line 44 of file DCPEventStruct.cpp.

References _itHours, _itMinutes, and _itSeconds.

22.16.3.3 const std::string AIRINV::DCPEventStruct::describe ( ) const

Display of the structure.

Definition at line 53 of file DCPEventStruct.cpp.

References _advancePurchase, _airlineCodeList, _cabinCode, _changeFees, _channel, _classCodeList, _dateRangeEnd, _dateRangeStart, _DCP, _destination, _minimumStay, _nonRefundable, _origin, _pos, _saturdayStay, _timeRangeEnd, and _timeRangeStart.

22.16.3.4 const unsigned int AIRINV::DCPEventStruct::getAirlineListSize ( ) const [inline]

Get the size of the airline code list.

Definition at line 37 of file DCPEventStruct.hpp.

References _airlineCodeList.

22.16.3.5 const unsigned int AIRINV::DCPEventStruct::getClassCodeListSize ( ) const [inline]

Get the size of the class code list.

Definition at line 42 of file DCPEventStruct.hpp.

References _classCodeList.

22.16.3.6 const stdair::AirlineCode_T & AIRINV::DCPEventStruct::getFirstAirlineCode ( ) const

Get the first airline code.

Definition at line 87 of file DCPEventStruct.cpp.

References _airlineCodeList.

22.16.3.7 void AIRINV::DCPEventStruct::beginAirline ( )

Initialise the internal iterators on airline code: The current iterator is set on the first airline code, the next iterator is set on the second one.

Definition at line 95 of file DCPEventStruct.cpp.

References _airlineCodeList, and _itCurrentAirlineCode.

22.16.3.8 bool AIRINV::DCPEventStruct::hasNotReachedEndAirline ( ) const

States whether or not the end of the (airline code) list has been reached.

Definition at line 100 of file DCPEventStruct.cpp.

References _airlineCodeList, and _itCurrentAirlineCode.

22.16.3.9 stdair::AirlineCode_T AIRINV::DCPEventStruct::getCurrentAirlineCode ( ) const

Get the current element (airline code).

Definition at line 106 of file DCPEventStruct.cpp.

References _airlineCodeList, and _itCurrentAirlineCode.

22.16.3.10 void AIRINV::DCPEventStruct::iterateAirline ( )

Iterate for one element (airline code): increment both internal iterators on Buckets.

Definition at line 112 of file DCPEventStruct.cpp.

References _classCodeList, and _itCurrentAirlineCode.

22.16.3.11 const std::string & AIRINV::DCPEventStruct::getFirstClassCode ( ) const

Get the first class code list as a string.

Definition at line 119 of file DCPEventStruct.cpp.

References _classCodeList.

22.16.3.12 void AIRINV::DCPEventStruct::beginClassCode ( )

Initialise the internal iterators on class code: The current iterator is set on the first class code, the next iterator is set on the second one.

Definition at line 127 of file DCPEventStruct.cpp.

References _classCodeList, and _itCurrentClassCode.

22.16.3.13 bool AIRINV::DCPEventStruct::hasNotReachedEndClassCode ( ) const

States whether or not the end of the (class code) list has been reached.

Definition at line 132 of file DCPEventStruct.cpp.

References _classCodeList, and _itCurrentClassCode.

22.16.3.14 std::string AIRINV::DCPEventStruct::getCurrentClassCode ( ) const

Get the current element (class code).

Definition at line 138 of file DCPEventStruct.cpp.

References _classCodeList, and _itCurrentClassCode.

22.16.3.15 void AIRINV::DCPEventStruct::iterateClassCode ( )

Iterate for one element (classCode): increment both internal iterators on Buckets.

Definition at line 145 of file DCPEventStruct.cpp.

References _classCodeList, and _itCurrentClassCode.

22.16.4 Member Data Documentation

22.16.4.1 stdair::year_t AIRINV::DCPEventStruct::_itYear

Staging Date.

Definition at line 87 of file DCPEventStruct.hpp.

Referenced by getDate().

22.16.4.2 stdair::month_t AIRINV::DCPEventStruct::_itMonth

Definition at line 88 of file DCPEventStruct.hpp.

Referenced by getDate().

22.16.4.3 stdair::day_t AIRINV::DCPEventStruct::_itDay

Definition at line 89 of file DCPEventStruct.hpp.

Referenced by getDate().

22.16.4.4 stdair::hour_t AIRINV::DCPEventStruct::_itHours

Staging Time.

Definition at line 93 of file DCPEventStruct.hpp.

Referenced by getTime().

22.16.4.5 stdair::minute_t AIRINV::DCPEventStruct::_itMinutes

Definition at line 94 of file DCPEventStruct.hpp.

Referenced by getTime().

22.16.4.6 stdair::second_t AIRINV::DCPEventStruct::_itSeconds

Definition at line 95 of file DCPEventStruct.hpp.

Referenced by getTime().

22.16.4.7 stdair::AirlineCodeList_T::iterator AIRINV::DCPEventStruct::_itCurrentAirlineCode

Iterator for the current airline code list.

Definition at line 98 of file DCPEventStruct.hpp.

Referenced by beginAirline(), getCurrentAirlineCode(), hasNotReachedEndAirline(), and iterateAirline().

22.16.4.8 stdair::ClassList_StringList_T::iterator AIRINV::DCPEventStruct::_itCurrentClassCode

Iterator for the current class code.

Definition at line 101 of file DCPEventStruct.hpp.

Referenced by beginClassCode(), getCurrentClassCode(), hasNotReachedEndClassCode(), and iterateClassCode().

22.16.4.9 stdair::AirportCode_T AIRINV::DCPEventStruct::_origin

Origin.

Definition at line 104 of file DCPEventStruct.hpp.

Referenced by describe().

22.16.4.10 stdair::AirportCode_T AIRINV::DCPEventStruct::_destination

Destination.

Definition at line 107 of file DCPEventStruct.hpp.

Referenced by describe().

22.16.4.11 stdair::Date_T AIRINV::DCPEventStruct::_dateRangeStart

Start Range date available for this DCP event.

Definition at line 110 of file DCPEventStruct.hpp.

Referenced by describe().

22.16.4.12 stdair::Date_T AIRINV::DCPEventStruct::_dateRangeEnd

Start Range date available for this DCP event.

Definition at line 113 of file DCPEventStruct.hpp.

Referenced by describe().

22.16.4.13 stdair::Duration_T AIRINV::DCPEventStruct::_timeRangeStart

Start time from the time range available for this DCP event.

Definition at line 116 of file DCPEventStruct.hpp.

Referenced by describe().

22.16.4.14 stdair::Duration_T AIRINV::DCPEventStruct::_timeRangeEnd

End time from the time range available for this DCP event.

Definition at line 119 of file DCPEventStruct.hpp.

Referenced by describe().

22.16.4.15 stdair::CabinCode_T AIRINV::DCPEventStruct::_cabinCode

Cabin code.

Definition at line 122 of file DCPEventStruct.hpp.

Referenced by describe().

22.16.4.16 stdair::CityCode_T AIRINV::DCPEventStruct::_pos

Point-of-sale.

Definition at line 125 of file DCPEventStruct.hpp.

Referenced by describe().

22.16.4.17 stdair::ChannelLabel_T AIRINV::DCPEventStruct::_channel

Channel distribution.

Definition at line 128 of file DCPEventStruct.hpp.

Referenced by describe().

22.16.4.18 stdair::DayDuration_T AIRINV::DCPEventStruct::_advancePurchase

Number of days that the ticket is sold before the flightDate.

Definition at line 131 of file DCPEventStruct.hpp.

Referenced by describe().

22.16.4.19 stdair::SaturdayStay_T AIRINV::DCPEventStruct::_saturdayStay

Boolean saying whether a saturday is considered during the stay .

Definition at line 134 of file DCPEventStruct.hpp.

Referenced by describe().

22.16.4.20 stdair::ChangeFees_T AIRINV::DCPEventStruct::_changeFees

Boolean saying whether the change fees option is requested or not.

Definition at line 137 of file DCPEventStruct.hpp.

Referenced by describe().

22.16.4.21 stdair::NonRefundable_T AIRINV::DCPEventStruct::_nonRefundable

Boolean saying whether the refundable option is requested or not.

Definition at line 140 of file DCPEventStruct.hpp.

Referenced by describe().

22.16.4.22 stdair::DayDuration_T AIRINV::DCPEventStruct::_minimumStay

Number of days that the customer spent into the destination city.

Definition at line 143 of file DCPEventStruct.hpp.

Referenced by describe().

22.16.4.23 stdair::PriceValue_T AIRINV::DCPEventStruct::_DCP

Price.

Definition at line 146 of file DCPEventStruct.hpp.

Referenced by describe().

22.16.4.24 stdair::AirlineCode_T AIRINV::DCPEventStruct::_airlineCode

Airline code

Definition at line 149 of file DCPEventStruct.hpp.

22.16.4.25 stdair::ClassCode_T AIRINV::DCPEventStruct::_classCode

Code

Definition at line 152 of file DCPEventStruct.hpp.

22.16.4.26 stdair::AirlineCodeList_T AIRINV::DCPEventStruct::_airlineCodeList

Airline Code List

Definition at line 155 of file DCPEventStruct.hpp.

Referenced by beginAirline(), describe(), getAirlineListSize(), getCurrentAirlineCode(), getFirstAirlineCode(), and hasNotReachedEndAirline().

22.16.4.27 stdair::ClassList_StringList_T AIRINV::DCPEventStruct::_classCodeList

Numbers of different Airline Codes Class Code List

Definition at line 161 of file DCPEventStruct.hpp.

Referenced by beginClassCode(), describe(), getClassCodeListSize(), getCurrentClassCode(), getFirstClassCode(), hasNotReachedEndClassCode(), iterateAirline(), and iterateClassCode().

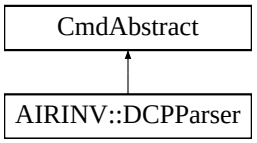
The documentation for this struct was generated from the following files:

- airinv/bom/DCPEventStruct.hpp
- airinv/bom/DCPEventStruct.cpp

22.17 AIRINV::DCPParser Class Reference

`#include <airinv/command/vault/DCPParser.hpp>`

Inheritance diagram for AIRINV::DCPParser:



Static Public Member Functions

- static void DCPRuleGeneration (const stdair::Filename_T &, stdair::BomRoot &)

22.17.1 Detailed Description

Class wrapping the parser entry point.

Definition at line 19 of file DCPParser.hpp.

22.17.2 Member Function Documentation

22.17.2.1 void AIRINV::DCPParser::DCPRuleGeneration (const stdair::Filename_T & *iFilename*, stdair::BomRoot & *ioBomRoot*) [static]

Parses the CSV file describing the DCPs for the simulator, and generates the event structures accordingly.

Parameters

const	stdair::Filename_T& The file-name of the CSV-formatted DCP input file.
stdair::BomRoot&	Root of the BOM tree.

Definition at line 16 of file DCPParser.cpp.

References AIRINV::DCPRuleFileParser::generateDCPRules().

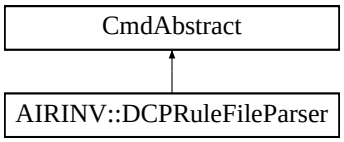
The documentation for this class was generated from the following files:

- airinv/command/vault/DCPParser.hpp
- airinv/command/vault/DCPParser.cpp

22.18 AIRINV::DCPRuleFileParser Class Reference

`#include <airinv/command/vault/DCPParserHelper.hpp>`

Inheritance diagram for AIRINV::DCPRuleFileParser:



Public Member Functions

- DCPRuleFileParser (stdair::BomRoot &ioBomRoot, const stdair::Filename_T &iFilename)
- bool generateDCPRules ()

22.18.1 Detailed Description

Class wrapping the initialisation and entry point of the parser.

The seemingly redundancy is used to force the instantiation of the actual parser, which is a templatised Boost Spirit grammar. Hence, the actual parser is instantiated within that class object code.

Definition at line 337 of file DCPParserHelper.hpp.

22.18.2 Constructor & Destructor Documentation

22.18.2.1 AIRINV::DCPRuleFileParser::DCPRuleFileParser (stdair::BomRoot & *ioBomRoot,* const stdair::Filename_T & *iFilename*)

Constructor.

Definition at line 572 of file DCPParserHelper.cpp.

22.18.3 Member Function Documentation

22.18.3.1 bool AIRINV::DCPRuleFileParser::generateDCPRules ( )

Parse the input file and generate the Inventories.

Definition at line 593 of file DCPParserHelper.cpp.

Referenced by AIRINV::DCPParser::DCPRuleGeneration().

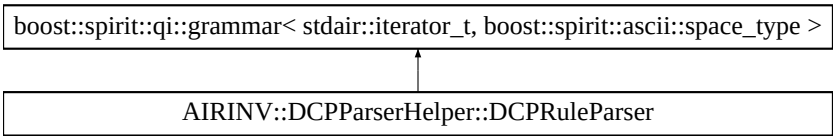
The documentation for this class was generated from the following files:

- airinv/command/vault/DCPParserHelper.hpp
- airinv/command/vault/DCPParserHelper.cpp

22.19 AIRINV::DCPParserHelper::DCPRuleParser Struct Reference

`#include <airinv/command/vault/DCPParserHelper.hpp>`

Inheritance diagram for AIRINV::DCPParserHelper::DCPRuleParser:



Public Member Functions

- DCPRuleParser (stdair::BomRoot &, DCPRuleStruct &)

Public Attributes

- boost::spirit::qi::rule < stdair::iterator_t, boost::spirit::ascii::space_type > start
- boost::spirit::qi::rule < stdair::iterator_t, boost::spirit::ascii::space_type > comments
- boost::spirit::qi::rule < stdair::iterator_t, boost::spirit::ascii::space_type > DCP_rule
- boost::spirit::qi::rule < stdair::iterator_t, boost::spirit::ascii::space_type > DCP_rule_end
- boost::spirit::qi::rule < stdair::iterator_t, boost::spirit::ascii::space_type > DCP_key
- boost::spirit::qi::rule < stdair::iterator_t, boost::spirit::ascii::space_type > DCP_id
- boost::spirit::qi::rule < stdair::iterator_t, boost::spirit::ascii::space_type > origin
- boost::spirit::qi::rule < stdair::iterator_t, boost::spirit::ascii::space_type > destination
- boost::spirit::qi::rule < stdair::iterator_t, boost::spirit::ascii::space_type > dateRangeStart
- boost::spirit::qi::rule < stdair::iterator_t, boost::spirit::ascii::space_type > dateRangeEnd
- boost::spirit::qi::rule < stdair::iterator_t, boost::spirit::ascii::space_type > date
- boost::spirit::qi::rule < stdair::iterator_t, boost::spirit::ascii::space_type > timeRangeStart
- boost::spirit::qi::rule < stdair::iterator_t, boost::spirit::ascii::space_type > timeRangeEnd
- boost::spirit::qi::rule < stdair::iterator_t, boost::spirit::ascii::space_type > time
- boost::spirit::qi::rule < stdair::iterator_t, boost::spirit::ascii::space_type > position
- boost::spirit::qi::rule < stdair::iterator_t, boost::spirit::ascii::space_type > cabinCode
- boost::spirit::qi::rule < stdair::iterator_t, boost::spirit::ascii::space_type > channel
- boost::spirit::qi::rule < stdair::iterator_t, boost::spirit::ascii::space_type > advancePurchase

- boost::spirit::qi::rule
  < stdair::iterator_t,
  boost::spirit::ascii::space_type > saturdayStay
- boost::spirit::qi::rule
  < stdair::iterator_t,
  boost::spirit::ascii::space_type > changeFees
- boost::spirit::qi::rule
  < stdair::iterator_t,
  boost::spirit::ascii::space_type > nonRefundable
- boost::spirit::qi::rule
  < stdair::iterator_t,
  boost::spirit::ascii::space_type > minimumStay
- boost::spirit::qi::rule
  < stdair::iterator_t,
  boost::spirit::ascii::space_type > DCP
- boost::spirit::qi::rule
  < stdair::iterator_t,
  boost::spirit::ascii::space_type > segment
- boost::spirit::qi::rule
  < stdair::iterator_t,
  boost::spirit::ascii::space_type > list_class
- stdair::BomRoot & _bomRoot
- DCPRuleStruct & _DCPRule

22.19.1 Detailed Description

DCP: DCPID; OriginCity; DestinationCity; DateRangeStart; DateRangeEnd; DepartureTimeRangeStart; DepartureTimeRangeEnd; POS; AdvancePurchase; SaturdayNight; ChangeFees; NonRefundable; MinimumStay; Price; AirlineCode; Class;

DCPID OriginCity (3-char airport code) DestinationCity (3-char airport code) DateRangeStart (yyyy-mm-dd) DateRangeEnd (yyyy-mm-dd) DepartureTimeRangeStart (hh:mm) DepartureTimeRangeEnd (hh:mm) POS (3-char position city) Cabin Code (1-char cabin code) Channel (D=direct, I=indirect, N=oNline, F=oFfline) AdvancePurchase SaturdayNight (T=True, F=False) ChangeFees (T=True, F=False) NonRefundable (T=True, F=False) MinimumStay Price AirlineCode (2-char airline code) ClassList (List of 1-char class code)

Grammar: Demand ::= PrefDepDate ';' Origin ';' Destination ';' PassengerType ';' DemandParams ';' PosDist ';' ChannelDist ';' TripDist ';' StayDist ';' FfDist ';' PrefDepTimeDist ';' minWTP ';' TimeValueDist ';' DtdDist EndOfDemand PrefDepDate ::= date PassengerType ::= 'T' | 'F' DemandParams ::= DemandMean ';' DemandStdDev PosDist ::= PosPair (',' PosPair)* PosPair ::= PosCode ':' PosShare PosCode ::= AirportCode | "row" PosShare ::= real ChannelDist ::= ChannelPair (',' ChannelPair)* ChannelPair ::= Channel_Code ':' ChannelShare ChannelCode ::= "DF" | "DN" | "IF" | "IN" ChannelShare ::= real TripDist ::= TripPair (',' TripPair)* TripPair ::= TripCode ':' TripShare TripCode ::= "RO" | "RI" | "OW" TripShare ::= real StayDist ::= StayPair (',' StayPair)* StayPair ::= [0;3]-digit-integer ':' stay_share StayShare ::= real FFDist ::= FF_Pair (',' FF_Pair)* FFPair ::= FFCode ':' FFShare FFCode ::= 'P' | 'G' | 'S' | 'M' | 'N' FFShare ::= real PrefDepTimeDist ::= PrefDepTimePair (',' PrefDepTimePair)* PrefDepTimePair ::= time ':' PrefDepTimeShare PrefDepTimeShare ::= real minWTP ::= real TimeValueDist ::= TimeValuePair (',' TimeValuePair)* TimeValuePair ::= [0;2]-digit-integer ':' TimeValueShare TimeValueShare ::= real DTDDist ::= DTDPair (',' DTDPair)* DTDPair ::= real ':' DTDShare DTDShare ::= real EndOfDemand ::= ';' Grammar for the DCP-Rule parser.

Definition at line 304 of file DCPParserHelper.hpp.

22.19.2 Constructor & Destructor Documentation

22.19.2.1 AIRINV::DCPParserHelper::DCPRuleParser::DCPRuleParser (stdair::BomRoot & *ioBomRoot,* DCPRuleStruct & *ioDCPRule*)

Definition at line 453 of file DCPParserHelper.cpp.

References _bomRoot, _DCPRule, advancePurchase, cabinCode, changeFees, channel, comments, date, dateRangeEnd, dateRangeStart, AIRINV::DCPParserHelper::day_p, DCP, DCP_id, DCP_key, DCP_rule, DCP_rule_end, destination, AIRINV::DCPParserHelper::hour_p, list_class, minimumStay, AIRINV::DCPParserHelper::minute_p, AIRINV::DCPParserHelper::month_p, nonRefundable, origin, position, saturdayStay, AIRINV::DCPParserHelper::second_p, segment, start, time, timeRangeEnd, timeRangeStart, AIRINV::DCPParserHelper::uint1_4_p, and AIRINV::DCPParserHelper::year_p.

22.19.3 Member Data Documentation

22.19.3.1 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::start

Definition at line 313 of file DCPParserHelper.hpp.

Referenced by DCPRuleParser().

22.19.3.2 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::comments

Definition at line 313 of file DCPParserHelper.hpp.

Referenced by DCPRuleParser().

22.19.3.3 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::DCP_rule

Definition at line 313 of file DCPParserHelper.hpp.

Referenced by DCPRuleParser().

22.19.3.4 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::DCP_rule_end

Definition at line 313 of file DCPParserHelper.hpp.

Referenced by DCPRuleParser().

22.19.3.5 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::DCP_key

Definition at line 313 of file DCPParserHelper.hpp.

Referenced by DCPRuleParser().

22.19.3.6 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::DCP_id

Definition at line 313 of file DCPParserHelper.hpp.

Referenced by DCPRuleParser().

22.19.3.7 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::origin

Definition at line 313 of file DCPParserHelper.hpp.

Referenced by DCPRuleParser().

22.19.3.8 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::destination

Definition at line 313 of file DCPParserHelper.hpp.

Referenced by DCPRuleParser().

22.19.3.9 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::dateRangeStart

Definition at line 313 of file DCPParserHelper.hpp.

Referenced by DCPRuleParser().

22.19.3.10 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::dateRangeEnd

Definition at line 313 of file DCPParserHelper.hpp.

Referenced by DCPRuleParser().

22.19.3.11 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::date

Definition at line 313 of file DCPParserHelper.hpp.

Referenced by DCPRuleParser().

22.19.3.12 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::timeRangeStart

Definition at line 313 of file DCPParserHelper.hpp.

Referenced by DCPRuleParser().

22.19.3.13 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::timeRangeEnd

Definition at line 313 of file DCPParserHelper.hpp.

Referenced by DCPRuleParser().

22.19.3.14 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::time

Definition at line 313 of file DCPParserHelper.hpp.

Referenced by DCPRuleParser().

22.19.3.15 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::position

Definition at line 313 of file DCPParserHelper.hpp.

Referenced by DCPRuleParser().

22.19.3.16 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::cabinCode

Definition at line 313 of file DCPParserHelper.hpp.

Referenced by DCPRuleParser().

22.19.3.17 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::channel

Definition at line 313 of file DCPParserHelper.hpp.

Referenced by DCPRuleParser().

22.19.3.18 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::advancePurchase

Definition at line 313 of file DCPParserHelper.hpp.

Referenced by DCPRuleParser().

22.19.3.19 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::saturdayStay

Definition at line 313 of file DCPParserHelper.hpp.

Referenced by DCPRuleParser().

22.19.3.20 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::changeFees

Definition at line 313 of file DCPParserHelper.hpp.

Referenced by DCPRuleParser().

22.19.3.21 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::nonRefundable

Definition at line 313 of file DCPParserHelper.hpp.

Referenced by DCPRuleParser().

22.19.3.22 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::minimumStay

Definition at line 313 of file DCPParserHelper.hpp.

Referenced by DCPRuleParser().

22.19.3.23 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::DCP

Definition at line 313 of file DCPParserHelper.hpp.

Referenced by DCPRuleParser().

22.19.3.24 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::segment

Definition at line 313 of file DCPParserHelper.hpp.

Referenced by DCPRuleParser().

22.19.3.25 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRINV::DCPParserHelper::DCPRuleParser::list_class

Definition at line 313 of file DCPParserHelper.hpp.

Referenced by DCPRuleParser().

22.19.3.26 stdair::BomRoot& AIRINV::DCPParserHelper::DCPRuleParser::_bomRoot

Definition at line 320 of file DCPParserHelper.hpp.

Referenced by DCPRuleParser().

22.19.3.27 DCPRuleStruct& AIRINV::DCPParserHelper::DCPRuleParser::_DCPRule

Definition at line 321 of file DCPParserHelper.hpp.

Referenced by DCPRuleParser().

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

- airinv/command/vault/DCPParserHelper.hpp
- airinv/command/vault/DCPParserHelper.cpp

22.20 AIRINV::DefaultMap Struct Reference

`#include <airinv/basic/BasConst_Curves.hpp>`

Static Public Member Functions

- static FRAT5Curve_T createPickupFRAT5Curve ()

22.20.1 Detailed Description

Default PoS probability mass.

Definition at line 16 of file BasConst_Curves.hpp.

22.20.2 Member Function Documentation

22.20.2.1 FRAT5Curve_T AIRINV::DefaultMap::createPickupFRAT5Curve ( ) [static]

Definition at line 16 of file BasConst.cpp.

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

- airinv/basic/BasConst_Curves.hpp
- airinv/basic/BasConst.cpp

22.21 AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT > Struct Template Reference

`#include <airinv/command/ScheduleParserHelper.hpp>`

Public Member Functions

- definition (FlightPeriodParser const &self)
- boost::spirit::classic::rule < ScannerT > const & start () const

Public Attributes

- boost::spirit::classic::rule < ScannerT > flight_period_list
- boost::spirit::classic::rule < ScannerT > not_to_be_parsed
- boost::spirit::classic::rule < ScannerT > flight_period
- boost::spirit::classic::rule < ScannerT > flight_period_end

- boost::spirit::classic::rule < ScannerT > flight_key
- boost::spirit::classic::rule < ScannerT > airline_code
- boost::spirit::classic::rule < ScannerT > flight_number
- boost::spirit::classic::rule < ScannerT > date
- boost::spirit::classic::rule < ScannerT > dow
- boost::spirit::classic::rule < ScannerT > time
- boost::spirit::classic::rule < ScannerT > date_offset
- boost::spirit::classic::rule < ScannerT > leg
- boost::spirit::classic::rule < ScannerT > leg_key
- boost::spirit::classic::rule < ScannerT > operating_leg_details
- boost::spirit::classic::rule < ScannerT > leg_details
- boost::spirit::classic::rule < ScannerT > leg_cabin_details
- boost::spirit::classic::rule < ScannerT > segment_section
- boost::spirit::classic::rule < ScannerT > segment_key
- boost::spirit::classic::rule < ScannerT > full_segment_cabin_details
- boost::spirit::classic::rule < ScannerT > segment_cabin_details
- boost::spirit::classic::rule < ScannerT > full_family_cabin_details
- boost::spirit::classic::rule < ScannerT > family_cabin_details
- boost::spirit::classic::rule < ScannerT > generic_segment
- boost::spirit::classic::rule < ScannerT > specific_segment_list

22.21.1 Detailed Description

template<typename ScannerT>struct AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >

Definition at line 288 of file ScheduleParserHelper.hpp.

22.21.2 Constructor & Destructor Documentation

22.21.2.1 template<typename ScannerT > AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::definition (FlightPeriodParser const & *self*)

Definition at line 538 of file ScheduleParserHelper.cpp.

References AIRINV::ScheduleParserHelper::airline_code_p(), AIRINV::ScheduleParserHelper::airport_p(), AIRINV::ScheduleParserHelper::cabin_code_p(), AIRINV::ScheduleParserHelper::class_code_list_p(), AIRINV::ScheduleParserHelper::day_p(), AIRINV::ScheduleParserHelper::dow_p(), AIRINV::ScheduleParserHelper::family_code_p, AIRINV::ScheduleParserHelper::flight_number_p(), AIRINV::ScheduleParserHelper::hours_p(), AIRINV::ScheduleParserHelper::int1_p, AIRINV::ScheduleParserHelper::key_p(), AIRINV::ScheduleParserHelper::minutes_p(), AIRINV::ScheduleParserHelper::month_p(), AIRINV::ScheduleParserHelper::seconds_p(), and AIRINV::ScheduleParserHelper::year_p().

22.21.3 Member Function Documentation

22.21.3.1 template<typename ScannerT > bsc::rule< ScannerT > const & AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::start ( ) const

Entry point of the parser.

Definition at line 692 of file ScheduleParserHelper.cpp.

22.21.4 Member Data Documentation

22.21.4.1 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::flight_period_list

Definition at line 292 of file ScheduleParserHelper.hpp.

22.21.4.2 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::not_to_be_parsed

Definition at line 292 of file ScheduleParserHelper.hpp.

22.21.4.3 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::flight_period

Definition at line 292 of file ScheduleParserHelper.hpp.

22.21.4.4 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::flight_period_end

Definition at line 292 of file ScheduleParserHelper.hpp.

22.21.4.5 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::flight_key

Definition at line 292 of file ScheduleParserHelper.hpp.

22.21.4.6 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::airline_code

Definition at line 292 of file ScheduleParserHelper.hpp.

22.21.4.7 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::flight_number

Definition at line 292 of file ScheduleParserHelper.hpp.

22.21.4.8 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >::date

Definition at line 292 of file ScheduleParserHelper.hpp.

22.21.4.9 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::-FlightPeriodParser::definition< ScannerT >::dow

Definition at line 292 of file ScheduleParserHelper.hpp.

22.21.4.10 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::-FlightPeriodParser::definition< ScannerT >::time

Definition at line 292 of file ScheduleParserHelper.hpp.

22.21.4.11 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::-FlightPeriodParser::definition< ScannerT >::date_offset

Definition at line 292 of file ScheduleParserHelper.hpp.

22.21.4.12 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::-FlightPeriodParser::definition< ScannerT >::leg

Definition at line 292 of file ScheduleParserHelper.hpp.

22.21.4.13 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::-FlightPeriodParser::definition< ScannerT >::leg_key

Definition at line 292 of file ScheduleParserHelper.hpp.

22.21.4.14 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::-FlightPeriodParser::definition< ScannerT >::operating_leg_details

Definition at line 292 of file ScheduleParserHelper.hpp.

22.21.4.15 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::-FlightPeriodParser::definition< ScannerT >::leg_details

Definition at line 292 of file ScheduleParserHelper.hpp.

22.21.4.16 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::-FlightPeriodParser::definition< ScannerT >::leg_cabin_details

Definition at line 292 of file ScheduleParserHelper.hpp.

22.21.4.17 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::-FlightPeriodParser::definition< ScannerT >::segment_section

Definition at line 292 of file ScheduleParserHelper.hpp.

22.21.4.18 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::-FlightPeriodParser::definition< ScannerT >::segment_key

Definition at line 292 of file ScheduleParserHelper.hpp.

22.21.4.19 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::-FlightPeriodParser::definition< ScannerT >::full_segment_cabin_details

Definition at line 292 of file ScheduleParserHelper.hpp.

22.21.4.20 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::-FlightPeriodParser::definition< ScannerT >::segment_cabin_details

Definition at line 292 of file ScheduleParserHelper.hpp.

22.21.4.21 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::-FlightPeriodParser::definition< ScannerT >::full_family_cabin_details

Definition at line 292 of file ScheduleParserHelper.hpp.

22.21.4.22 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::-FlightPeriodParser::definition< ScannerT >::family_cabin_details

Definition at line 292 of file ScheduleParserHelper.hpp.

22.21.4.23 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::-FlightPeriodParser::definition< ScannerT >::generic_segment

Definition at line 292 of file ScheduleParserHelper.hpp.

22.21.4.24 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::ScheduleParserHelper::-FlightPeriodParser::definition< ScannerT >::specific_segment_list

Definition at line 292 of file ScheduleParserHelper.hpp.

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

- airinv/command/ScheduleParserHelper.hpp
- airinv/command/ScheduleParserHelper.cpp

22.22 AIRINV::FFDisutilityParserHelper::FFDisutilityParser::definition< ScannerT > Struct Template Reference

`#include <airinv/command/FFDisutilityParserHelper.hpp>`

Public Member Functions

- definition (FFDisutilityParser const &self)
- boost::spirit::classic::rule < ScannerT > const & start () const

Public Attributes

- boost::spirit::classic::rule < ScannerT > curve_list
- boost::spirit::classic::rule < ScannerT > not_to_be_parsed
- boost::spirit::classic::rule < ScannerT > curve
- boost::spirit::classic::rule < ScannerT > key
- boost::spirit::classic::rule < ScannerT > map
- boost::spirit::classic::rule < ScannerT > value_pair
- boost::spirit::classic::rule < ScannerT > curve_end

22.22.1 Detailed Description

template<typename ScannerT>struct AIRINV::FFDisutilityParserHelper::FFDisutilityParser::definition< ScannerT >

Definition at line 95 of file FFDisutilityParserHelper.hpp.

22.22.2 Constructor & Destructor Documentation

22.22.2.1 template<typename ScannerT > AIRINV::FFDisutilityParserHelper::FFDisutilityParser::definition< ScannerT >::definition (FFDisutilityParser const & *self*)

Definition at line 123 of file FFDisutilityParserHelper.cpp.

References AIRINV::FFDisutilityParserHelper::key_p().

22.22.3 Member Function Documentation

22.22.3.1 template<typename ScannerT > bsc::rule< ScannerT > const & AIRINV::FFDisutilityParserHelper::FFDisutilityParser::definition< ScannerT >::start ( ) const

Entry point of the parser.

Definition at line 163 of file FFDisutilityParserHelper.cpp.

22.22.4 Member Data Documentation

22.22.4.1 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::FFDisutilityParserHelper::FFDisutilityParser::definition< ScannerT >::curve_list

Definition at line 99 of file FFDisutilityParserHelper.hpp.

22.22.4.2 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::FFDisutilityParserHelper::FFDisutilityParser::definition< ScannerT >::not_to_be_parsed

Definition at line 99 of file FFDisutilityParserHelper.hpp.

22.22.4.3 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::FFDisutilityParserHelper::FFDisutilityParser::definition< ScannerT >::curve

Definition at line 99 of file FFDisutilityParserHelper.hpp.

22.22.4.4 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::FFDisutilityParserHelper::FFDisutilityParser::definition< ScannerT >::key

Definition at line 99 of file FFDisutilityParserHelper.hpp.

22.22.4.5 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::FFDisutilityParserHelper::FFDisutilityParser::definition< ScannerT >::map

Definition at line 99 of file FFDisutilityParserHelper.hpp.

22.22.4.6 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::FFDisutilityParserHelper::FFDisutilityParser::definition< ScannerT >::value_pair

Definition at line 99 of file FFDisutilityParserHelper.hpp.

22.22.4.7 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::FFDisutilityParserHelper::FFDisutilityParser::definition< ScannerT >::curve_end

Definition at line 99 of file FFDisutilityParserHelper.hpp.

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

- airinv/command/FFDisutilityParserHelper.hpp
- airinv/command/FFDisutilityParserHelper.cpp

22.23 AIRINV::FRAT5ParserHelper::FRAT5Parser::definition< ScannerT > Struct Template Reference

`#include <airinv/command/FRAT5ParserHelper.hpp>`

Public Member Functions

- definition (FRAT5Parser const &self)
- boost::spirit::classic::rule < ScannerT > const & start () const

Public Attributes

- boost::spirit::classic::rule < ScannerT > curve_list
- boost::spirit::classic::rule < ScannerT > not_to_be_parsed
- boost::spirit::classic::rule < ScannerT > curve
- boost::spirit::classic::rule < ScannerT > key
- boost::spirit::classic::rule < ScannerT > map
- boost::spirit::classic::rule < ScannerT > value_pair
- boost::spirit::classic::rule < ScannerT > curve_end

22.23.1 Detailed Description

template<typename ScannerT>struct AIRINV::FRAT5ParserHelper::FRAT5Parser::definition< ScannerT >

Definition at line 95 of file FRAT5ParserHelper.hpp.

22.23.2 Constructor & Destructor Documentation

22.23.2.1 template<typename ScannerT > AIRINV::FRAT5ParserHelper::FRAT5Parser::definition< ScannerT >::definition (FRAT5Parser const & *self*)

Definition at line 124 of file FRAT5ParserHelper.cpp.

References AIRINV::FRAT5ParserHelper::key_p().

22.23.3 Member Function Documentation

22.23.3.1 template<typename ScannerT > bsc::rule< ScannerT > const & AIRINV::FRAT5ParserHelper::FRAT5-Parser::definition< ScannerT >::start ( ) const

Entry point of the parser.

Definition at line 164 of file FRAT5ParserHelper.cpp.

22.23.4 Member Data Documentation

22.23.4.1 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::FRAT5ParserHelper::FRAT5Parser::definition< ScannerT >::curve_list

Definition at line 99 of file FRAT5ParserHelper.hpp.

22.23.4.2 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::FRAT5ParserHelper::FRAT5Parser::definition< ScannerT >::not_to_be_parsed

Definition at line 99 of file FRAT5ParserHelper.hpp.

22.23.4.3 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::FRAT5ParserHelper::FRAT5Parser::definition< ScannerT >::curve

Definition at line 99 of file FRAT5ParserHelper.hpp.

22.23.4.4 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::FRAT5ParserHelper::FRAT5Parser::definition< ScannerT >::key

Definition at line 99 of file FRAT5ParserHelper.hpp.

22.23.4.5 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::FRAT5ParserHelper::FRAT5Parser::definition< ScannerT >::map

Definition at line 99 of file FRAT5ParserHelper.hpp.

22.23.4.6 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::FRAT5ParserHelper::FRAT5Parser::definition< ScannerT >::value_pair

Definition at line 99 of file FRAT5ParserHelper.hpp.

22.23.4.7 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::FRAT5ParserHelper::FRAT5Parser::definition< ScannerT >::curve_end

Definition at line 99 of file FRAT5ParserHelper.hpp.

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

- airinv/command/FRAT5ParserHelper.hpp
- airinv/command/FRAT5ParserHelper.cpp

22.24 AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT > Struct Template Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Public Member Functions

- definition (InventoryParser const &self)
- boost::spirit::classic::rule
  < ScannerT > const & start () const

Public Attributes

- boost::spirit::classic::rule
  < ScannerT > flight_date_list

- boost::spirit::classic::rule < ScannerT > not_to_be_parsed
- boost::spirit::classic::rule < ScannerT > flight_date
- boost::spirit::classic::rule < ScannerT > flight_date_end
- boost::spirit::classic::rule < ScannerT > flight_key
- boost::spirit::classic::rule < ScannerT > airline_code
- boost::spirit::classic::rule < ScannerT > flight_number
- boost::spirit::classic::rule < ScannerT > flight_type_code
- boost::spirit::classic::rule < ScannerT > flight_visibility_code
- boost::spirit::classic::rule < ScannerT > date
- boost::spirit::classic::rule < ScannerT > leg_list
- boost::spirit::classic::rule < ScannerT > leg
- boost::spirit::classic::rule < ScannerT > operating_leg_details
- boost::spirit::classic::rule < ScannerT > leg_key
- boost::spirit::classic::rule < ScannerT > leg_details
- boost::spirit::classic::rule < ScannerT > leg_cabin_list
- boost::spirit::classic::rule < ScannerT > leg_cabin_details
- boost::spirit::classic::rule < ScannerT > bucket_list
- boost::spirit::classic::rule < ScannerT > bucket_details
- boost::spirit::classic::rule < ScannerT > time
- boost::spirit::classic::rule < ScannerT > segment_list
- boost::spirit::classic::rule < ScannerT > segment
- boost::spirit::classic::rule < ScannerT > segment_key
- boost::spirit::classic::rule < ScannerT > full_segment_cabin_details
- boost::spirit::classic::rule < ScannerT > segment_cabin_list
- boost::spirit::classic::rule < ScannerT > segment_cabin_key
- boost::spirit::classic::rule < ScannerT > segment_cabin_details
- boost::spirit::classic::rule < ScannerT > class_list
- boost::spirit::classic::rule < ScannerT > class_key

- boost::spirit::classic::rule < ScannerT > parent_subclass_code
- boost::spirit::classic::rule < ScannerT > class_protection
- boost::spirit::classic::rule < ScannerT > class_nego
- boost::spirit::classic::rule < ScannerT > class_details
- boost::spirit::classic::rule < ScannerT > family_cabin_list
- boost::spirit::classic::rule < ScannerT > family_cabin_details

22.24.1 Detailed Description

template<typename ScannerT>struct AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >

Definition at line 476 of file InventoryParserHelper.hpp.

22.24.2 Constructor & Destructor Documentation

22.24.2.1 template<typename ScannerT > AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::definition (InventoryParser const & *self*)

Definition at line 904 of file InventoryParserHelper.cpp.

References AIRINV::InventoryParserHelper::airline_code_p(), AIRINV::InventoryParserHelper::airport_p(), AIRINV::InventoryParserHelper::cabin_code_p(), AIRINV::InventoryParserHelper::class_code_list_p(), AIRINV::InventoryParserHelper::class_code_p(), AIRINV::InventoryParserHelper::day_p(), AIRINV::InventoryParserHelper::family_code_p, AIRINV::InventoryParserHelper::flight_number_p(), AIRINV::InventoryParserHelper::hours_p(), AIRINV::InventoryParserHelper::minutes_p(), AIRINV::InventoryParserHelper::month_p(), AIRINV::InventoryParserHelper::seconds_p(), AIRINV::InventoryParserHelper::uint1_2_p, AIRINV::InventoryParserHelper::uint1_3_p, and AIRINV::InventoryParserHelper::year_p().

22.24.3 Member Function Documentation

22.24.3.1 template<typename ScannerT > bsc::rule< ScannerT > const & AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::start ( ) const

Entry point of the parser.

Definition at line 1118 of file InventoryParserHelper.cpp.

22.24.4 Member Data Documentation

22.24.4.1 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::flight_date_list

Definition at line 480 of file InventoryParserHelper.hpp.

22.24.4.2 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::not_to_be_parsed

Definition at line 480 of file InventoryParserHelper.hpp.

22.24.4.3 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::flight_date

Definition at line 480 of file InventoryParserHelper.hpp.

22.24.4.4 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::flight_date_end

Definition at line 480 of file InventoryParserHelper.hpp.

22.24.4.5 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::flight_key

Definition at line 480 of file InventoryParserHelper.hpp.

22.24.4.6 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::airline_code

Definition at line 480 of file InventoryParserHelper.hpp.

22.24.4.7 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::flight_number

Definition at line 480 of file InventoryParserHelper.hpp.

22.24.4.8 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::flight_type_code

Definition at line 480 of file InventoryParserHelper.hpp.

22.24.4.9 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::flight_visibility_code

Definition at line 480 of file InventoryParserHelper.hpp.

22.24.4.10 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::date

Definition at line 480 of file InventoryParserHelper.hpp.

22.24.4.11 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::leg_list

Definition at line 480 of file InventoryParserHelper.hpp.

22.24.4.12 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::leg

Definition at line 480 of file InventoryParserHelper.hpp.

22.24.4.13 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::operating_leg_details

Definition at line 480 of file InventoryParserHelper.hpp.

22.24.4.14 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::leg_key

Definition at line 480 of file InventoryParserHelper.hpp.

22.24.4.15 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::leg_details

Definition at line 480 of file InventoryParserHelper.hpp.

22.24.4.16 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::leg_cabin_list

Definition at line 480 of file InventoryParserHelper.hpp.

22.24.4.17 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::leg_cabin_details

Definition at line 480 of file InventoryParserHelper.hpp.

22.24.4.18 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::bucket_list

Definition at line 480 of file InventoryParserHelper.hpp.

22.24.4.19 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::bucket_details

Definition at line 480 of file InventoryParserHelper.hpp.

22.24.4.20 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::time

Definition at line 480 of file InventoryParserHelper.hpp.

22.24.4.21 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::segment_list

Definition at line 480 of file InventoryParserHelper.hpp.

22.24.4.22 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::segment

Definition at line 480 of file InventoryParserHelper.hpp.

22.24.4.23 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::segment_key

Definition at line 480 of file InventoryParserHelper.hpp.

22.24.4.24 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::full_segment_cabin_details

Definition at line 480 of file InventoryParserHelper.hpp.

22.24.4.25 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::segment_cabin_list

Definition at line 480 of file InventoryParserHelper.hpp.

22.24.4.26 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >::segment_cabin_key

Definition at line 480 of file InventoryParserHelper.hpp.

22.24.4.27 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::-InventoryParser::definition< ScannerT >::segment_cabin_details

Definition at line 480 of file InventoryParserHelper.hpp.

22.24.4.28 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::-InventoryParser::definition< ScannerT >::class_list

Definition at line 480 of file InventoryParserHelper.hpp.

22.24.4.29 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::-InventoryParser::definition< ScannerT >::class_key

Definition at line 480 of file InventoryParserHelper.hpp.

22.24.4.30 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::-InventoryParser::definition< ScannerT >::parent_subclass_code

Definition at line 480 of file InventoryParserHelper.hpp.

22.24.4.31 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::-InventoryParser::definition< ScannerT >::class_protection

Definition at line 480 of file InventoryParserHelper.hpp.

22.24.4.32 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::-InventoryParser::definition< ScannerT >::class_nego

Definition at line 480 of file InventoryParserHelper.hpp.

22.24.4.33 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::-InventoryParser::definition< ScannerT >::class_details

Definition at line 480 of file InventoryParserHelper.hpp.

22.24.4.34 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::-InventoryParser::definition< ScannerT >::family_cabin_list

Definition at line 480 of file InventoryParserHelper.hpp.

22.24.4.35 template<typename ScannerT > boost::spirit::classic::rule<ScannerT> AIRINV::InventoryParserHelper::-InventoryParser::definition< ScannerT >::family_cabin_details

Definition at line 480 of file InventoryParserHelper.hpp.

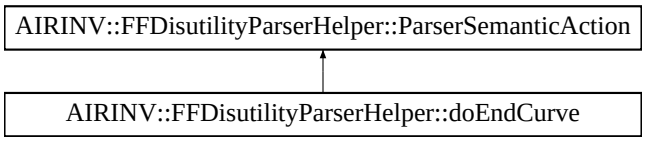
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.25 AIRINV::FFDisutilityParserHelper::doEndCurve Struct Reference

`#include <airinv/command/FFDisutilityParserHelper.hpp>`

Inheritance diagram for AIRINV::FFDisutilityParserHelper::doEndCurve:



Public Member Functions

- doEndCurve (stdair::BomRoot &, FFDisutilityStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- stdair::BomRoot & _bomRoot
- FFDisutilityStruct & _ffDisutility

22.25.1 Detailed Description

Mark the end of the FFDisutility curve parsing.

Definition at line 61 of file FFDisutilityParserHelper.hpp.

22.25.2 Constructor & Destructor Documentation

22.25.2.1 AIRINV::FFDisutilityParserHelper::doEndCurve::doEndCurve (stdair::BomRoot & *ioBomRoot,* FFDisutilityStruct & *ioFFDisutility*)

Actor Constructor.

Definition at line 79 of file FFDisutilityParserHelper.cpp.

22.25.3 Member Function Documentation

22.25.3.1 void AIRINV::FFDisutilityParserHelper::doEndCurve::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 87 of file FFDisutilityParserHelper.cpp.

References _bomRoot, AIRINV::FFDisutilityStruct::_curve, AIRINV::FFDisutilityParserHelper::ParserSemantic-Action::_ffDisutility, AIRINV::FFDisutilityStruct::_key, and AIRINV::FFDisutilityStruct::describe().

22.25.4 Member Data Documentation

22.25.4.1 stdair::BomRoot& AIRINV::FFDisutilityParserHelper::doEndCurve::_bomRoot

Actor Specific Context.

Definition at line 67 of file FFDisutilityParserHelper.hpp.

Referenced by operator()().

22.25.4.2 FFDisutilityStruct& AIRINV::FFDisutilityParserHelper::ParserSemanticAction::_ffDisutility [inherited]

Actor Context.

Definition at line 33 of file FFDisutilityParserHelper.hpp.

Referenced by AIRINV::FFDisutilityParserHelper::storeCurveKey::operator()(), AIRINV::FFDisutilityParserHelper::storeDTD::operator()(), AIRINV::FFDisutilityParserHelper::storeFFDisutilityValue::operator()(), and operator()().

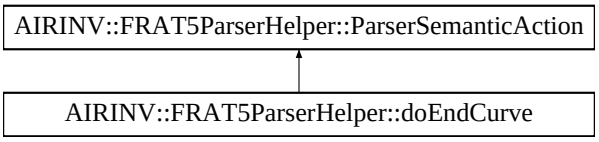
The documentation for this struct was generated from the following files:

- airinv/command/FFDisutilityParserHelper.hpp
- airinv/command/FFDisutilityParserHelper.cpp

22.26 AIRINV::FRAT5ParserHelper::doEndCurve Struct Reference

`#include <airinv/command/FRAT5ParserHelper.hpp>`

Inheritance diagram for AIRINV::FRAT5ParserHelper::doEndCurve:



Public Member Functions

- doEndCurve (stdair::BomRoot &, FRAT5Struct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- stdair::BomRoot & _bomRoot
- FRAT5Struct & _frat5

22.26.1 Detailed Description

Mark the end of the FRAT5 curve parsing.

Definition at line 61 of file FRAT5ParserHelper.hpp.

22.26.2 Constructor & Destructor Documentation

22.26.2.1 AIRINV::FRAT5ParserHelper::doEndCurve::doEndCurve (stdair::BomRoot & *ioBomRoot,* FRAT5Struct & *ioFRAT5*)

Actor Constructor.

Definition at line 80 of file FRAT5ParserHelper.cpp.

22.26.3 Member Function Documentation

22.26.3.1 void AIRINV::FRAT5ParserHelper::doEndCurve::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 88 of file FRAT5ParserHelper.cpp.

References _bomRoot, AIRINV::FRAT5Struct::_curve, AIRINV::FRAT5ParserHelper::ParserSemanticAction::_frat5, AIRINV::FRAT5Struct::_key, and AIRINV::FRAT5Struct::describe().

22.26.4 Member Data Documentation

22.26.4.1 stdair::BomRoot& AIRINV::FRAT5ParserHelper::doEndCurve::_bomRoot

Actor Specific Context.

Definition at line 67 of file FRAT5ParserHelper.hpp.

Referenced by operator()().

22.26.4.2 FRAT5Struct& AIRINV::FRAT5ParserHelper::ParserSemanticAction::_frat5 [inherited]

Actor Context.

Definition at line 33 of file FRAT5ParserHelper.hpp.

Referenced by AIRINV::FRAT5ParserHelper::storeCurveKey::operator()(), AIRINV::FRAT5ParserHelper::storeDTD::operator()(), AIRINV::FRAT5ParserHelper::storeFRAT5Value::operator()(), and operator()().

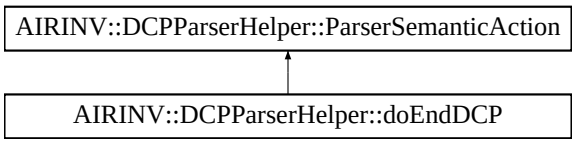
The documentation for this struct was generated from the following files:

- airinv/command/FRAT5ParserHelper.hpp
- airinv/command/FRAT5ParserHelper.cpp

22.27 AIRINV::DCPParserHelper::doEndDCP Struct Reference

`#include <airinv/command/vault/DCPParserHelper.hpp>`

Inheritance diagram for AIRINV::DCPParserHelper::doEndDCP:



Public Member Functions

- doEndDCP (stdair::BomRoot &, DCPRuleStruct &)
- void operator() (boost::spirit::qi::unused_type, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- stdair::BomRoot & _bomRoot
- DCPRuleStruct & _DCPRule

22.27.1 Detailed Description

Mark the end of the DCP-rule parsing.

Definition at line 218 of file DCPParserHelper.hpp.

22.27.2 Constructor & Destructor Documentation

22.27.2.1 AIRINV::DCPParserHelper::doEndDCP::doEndDCP (stdair::BomRoot & *ioBomRoot*, DCPRuleStruct & *ioDCPRule*)

Actor Constructor.

Definition at line 399 of file DCPParserHelper.cpp.

22.27.3 Member Function Documentation

22.27.3.1 void AIRINV::DCPParserHelper::doEndDCP::operator() (boost::spirit::qi::unused_type , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 406 of file DCPParserHelper.cpp.

References _bomRoot, and AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule.

22.27.4 Member Data Documentation

22.27.4.1 stdair::BomRoot& AIRINV::DCPParserHelper::doEndDCP::_bomRoot

Actor Specific Context.

Definition at line 226 of file DCPParserHelper.hpp.

Referenced by operator()().

22.27.4.2 DCPRuleStruct& AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule [inherited]

Actor Context.

Definition at line 34 of file DCPParserHelper.hpp.

Referenced by AIRINV::DCPParserHelper::storeDCPId::operator()(), AIRINV::DCPParserHelper::storeOrigin::operator()(), AIRINV::DCPParserHelper::storeDestination::operator()(), AIRINV::DCPParserHelper::storeDateRangeStart::operator()(), AIRINV::DCPParserHelper::storeDateRangeEnd::operator()(), AIRINV::DCPParserHelper::storeStartRangeTime::operator()(), AIRINV::DCPParserHelper::storeEndRangeTime::operator()(), AIRINV::DCPParserHelper::storePOS::operator()(), AIRINV::DCPParserHelper::storeCabinCode::operator()(), AIRINV::DCPParserHelper::storeChannel::operator()(), AIRINV::DCPParserHelper::storeAdvancePurchase::operator()(), AIRINV::DCPParserHelper::storeSaturdayStay::operator()(), AIRINV::DCPParserHelper::storeChangeFees::operator()(), AIRINV::DCPParserHelper::storeNonRefundable::operator()(), AIRINV::DCPParserHelper::storeMinimumStay::operator()(), AIRINV::DCPParserHelper::storeDCP::operator()(), AIRINV::DCPParserHelper::storeAirlineCode::operator()(), AIRINV::DCPParserHelper::storeClass::operator()(), and operator()().

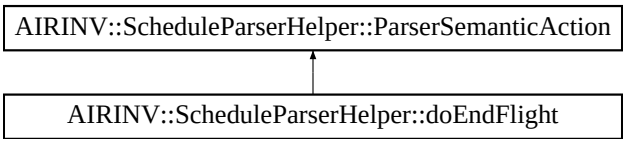
The documentation for this struct was generated from the following files:

- airinv/command/vault/DCPParserHelper.hpp
- airinv/command/vault/DCPParserHelper.cpp

22.28 AIRINV::ScheduleParserHelper::doEndFlight Struct Reference

`#include <airinv/command/ScheduleParserHelper.hpp>`

Inheritance diagram for AIRINV::ScheduleParserHelper::doEndFlight:



Public Member Functions

- doEndFlight (stdair::BomRoot &, FlightPeriodStruct &)

- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- stdair::BomRoot & _bomRoot
- FlightPeriodStruct & _flightPeriod

22.28.1 Detailed Description

Mark the end of the flight-period parsing.

Definition at line 224 of file ScheduleParserHelper.hpp.

22.28.2 Constructor & Destructor Documentation

22.28.2.1 AIRINV::ScheduleParserHelper::doEndFlight::doEndFlight (stdair::BomRoot & *ioBomRoot,* FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 436 of file ScheduleParserHelper.cpp.

22.28.3 Member Function Documentation

22.28.3.1 void AIRINV::ScheduleParserHelper::doEndFlight::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 444 of file ScheduleParserHelper.cpp.

References _bomRoot, AIRINV::LegStruct::_cabinList, AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod, AIRINV::FlightPeriodStruct::_itLeg, AIRINV::FlightPeriodStruct::_legAlreadyDefined, AIRINV::FlightPeriodStruct::_legList, and AIRINV::FlightPeriodStruct::describe().

22.28.4 Member Data Documentation

22.28.4.1 stdair::BomRoot& AIRINV::ScheduleParserHelper::doEndFlight::_bomRoot

Actor Specific Context.

Definition at line 230 of file ScheduleParserHelper.hpp.

Referenced by operator()().

22.28.4.2 FlightPeriodStruct& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRINV::ScheduleParserHelper::storeAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRINV::ScheduleParserHelper::storeDow::operator()(), AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeBoardingTime::operator()(), AIRINV::ScheduleParserHelper::storeOffTime::operator()(), AIRINV::ScheduleParserHelper::storeElapsedTime::operator()(), AIRINV::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeCapacity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator()(),

AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeClasses::operator()(), AIRINV::ScheduleParserHelper::storeFamilyCode::operator()(), AIRINV::ScheduleParserHelper::storeFRAT5CurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFFDisutilityCurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFClasses::operator()(), and operator()().

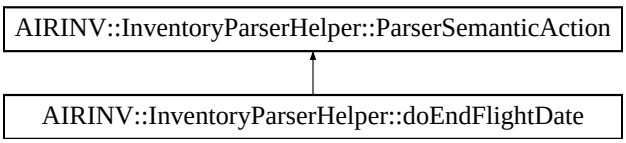
The documentation for this struct was generated from the following files:

- airinv/command/ScheduleParserHelper.hpp
- airinv/command/ScheduleParserHelper.cpp

22.29 AIRINV::InventoryParserHelper::doEndFlightDate Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::doEndFlightDate:



Public Member Functions

- doEndFlightDate (stdair::BomRoot &, FlightDateStruct &, unsigned int &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- stdair::BomRoot & _bomRoot
- unsigned int & _nbOfFlights
- FlightDateStruct & _flightDate

22.29.1 Detailed Description

Mark the end of the inventory parsing.

Definition at line 441 of file InventoryParserHelper.hpp.

22.29.2 Constructor & Destructor Documentation

22.29.2.1 AIRINV::InventoryParserHelper::doEndFlightDate::doEndFlightDate (stdair::BomRoot & *ioBomRoot*, FlightDateStruct & *ioFlightDate*, unsigned int & *ioNbOfFlights*)

Actor Constructor.

Definition at line 778 of file InventoryParserHelper.cpp.

22.29.3 Member Function Documentation

22.29.3.1 void AIRINV::InventoryParserHelper::doEndFlightDate::operator() (iterator_t *iStr*, iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 787 of file InventoryParserHelper.cpp.

References _bomRoot, AIRINV::SegmentStruct::_cabinList, AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::FlightDateStruct::_itSegment, _nbOfFlights, and AIRINV::FlightDateStruct::_segmentList.

22.29.4 Member Data Documentation

22.29.4.1 stdair::BomRoot& AIRINV::InventoryParserHelper::doEndFlightDate::_bomRoot

Actor Specific Context.

Definition at line 448 of file InventoryParserHelper.hpp.

Referenced by operator()().

22.29.4.2 unsigned int& AIRINV::InventoryParserHelper::doEndFlightDate::_nbOfFlights

Definition at line 449 of file InventoryParserHelper.hpp.

Referenced by operator()().

22.29.4.3 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and operator()().

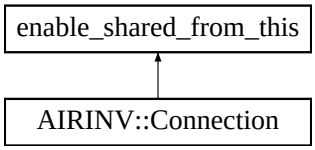
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp

- airinv/command/InventoryParserHelper.cpp

22.30 enable_shared_from_this Class Reference

Inheritance diagram for enable_shared_from_this:



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

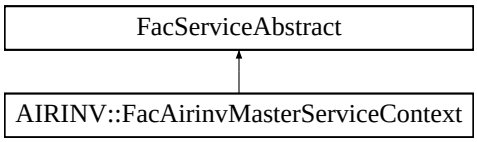
- airinv/server/Connection.hpp

22.31 AIRINV::FacAirinvMasterServiceContext Class Reference

Factory for Bucket.

`#include <airinv/factory/FacAirinvMasterServiceContext.hpp>`

Inheritance diagram for AIRINV::FacAirinvMasterServiceContext:



Public Member Functions

- ~FacAirinvMasterServiceContext ()
- AIRINV_Master_ServiceContext & create ()

Static Public Member Functions

- static FacAirinvMasterServiceContext & instance ()

Protected Member Functions

- FacAirinvMasterServiceContext ()

22.31.1 Detailed Description

Factory for Bucket.

Definition at line 20 of file FacAirinvMasterServiceContext.hpp.

22.31.2 Constructor & Destructor Documentation

22.31.2.1 AIRINV::FacAirinvMasterServiceContext::~FacAirinvMasterServiceContext ( )

Destructor.

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

Definition at line 17 of file FacAirinvMasterServiceContext.cpp.

22.31.2.2 AIRINV::FacAirinvMasterServiceContext::FacAirinvMasterServiceContext ( ) [inline],[protected]

Default Constructor.

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

Definition at line 44 of file FacAirinvMasterServiceContext.hpp.

Referenced by instance().

22.31.3 Member Function Documentation

22.31.3.1 FacAirinvMasterServiceContext & AIRINV::FacAirinvMasterServiceContext::instance ( ) [static]

Provide the unique instance.

The singleton is instantiated when first used

Returns

FacAirinvMasterServiceContext&

Definition at line 22 of file FacAirinvMasterServiceContext.cpp.

References FacAirinvMasterServiceContext().

22.31.3.2 AIRINV_Master_ServiceContext & AIRINV::FacAirinvMasterServiceContext::create ( )

Create a new AIRINV_Master_ServiceContext object.

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

Returns

AIRINV_Master_ServiceContext& The newly created object.

Definition at line 34 of file FacAirinvMasterServiceContext.cpp.

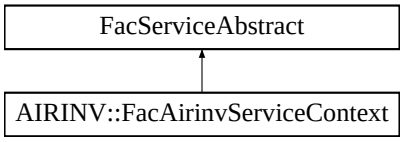
The documentation for this class was generated from the following files:

- airinv/factory/FacAirinvMasterServiceContext.hpp
- airinv/factory/FacAirinvMasterServiceContext.cpp

22.32 AIRINV::FacAirinvServiceContext Class Reference

`#include <airinv/factory/FacAirinvServiceContext.hpp>`

Inheritance diagram for AIRINV::FacAirinvServiceContext:



Public Member Functions

- ~FacAirinvServiceContext ()
- AIRINV_ServiceContext & create ()

Static Public Member Functions

- static FacAirinvServiceContext & instance ()

Protected Member Functions

- FacAirinvServiceContext ()

22.32.1 Detailed Description

Factory for Bucket.

Definition at line 18 of file FacAirinvServiceContext.hpp.

22.32.2 Constructor & Destructor Documentation

22.32.2.1 AIRINV::FacAirinvServiceContext::~FacAirinvServiceContext ( )

Destructor.

The Destruction put the _instance to NULL in order to be clean for the next FacAirinvServiceContext::instance()

Definition at line 17 of file FacAirinvServiceContext.cpp.

22.32.2.2 AIRINV::FacAirinvServiceContext::FacAirinvServiceContext ( ) [inline],[protected]

Default Constructor.

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

Definition at line 42 of file FacAirinvServiceContext.hpp.

Referenced by instance().

22.32.3 Member Function Documentation

22.32.3.1 FacAirinvServiceContext & AIRINV::FacAirinvServiceContext::instance ( ) [static]

Provide the unique instance.

The singleton is instantiated when first used

Returns

FacAirinvServiceContext&

Definition at line 22 of file FacAirinvServiceContext.cpp.

References FacAirinvServiceContext().

22.32.3.2 AIRINV_ServiceContext & AIRINV::FacAirinvServiceContext::create ( )

Create a new AIRINV_ServiceContext object.

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

Returns

AIRINV_ServiceContext& The newly created object.

Definition at line 34 of file FacAirinvServiceContext.cpp.

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

- airinv/factory/FacAirinvServiceContext.hpp
- airinv/factory/FacAirinvServiceContext.cpp

22.33 AIRINV::FacBomAbstract Class Reference

`#include <airinv/factory/FacBomAbstract.hpp>`

Public Types

- typedef std::vector < BomAbstract * > BomPool_T

Static Public Member Functions

- static std::size_t getID (const BomAbstract *)
- static std::size_t getID (const BomAbstract &)
- static std::string getIDString (const BomAbstract *)
- static std::string getIDString (const BomAbstract &)

Protected Member Functions

- FacBomAbstract ()
- FacBomAbstract (const FacBomAbstract &)
- virtual ~FacBomAbstract ()

Protected Attributes

- BomPool_T _pool

Friends

- class FacSupervisor

22.33.1 Detailed Description

Base class for Factory layer.

Definition at line 17 of file FacBomAbstract.hpp.

22.33.2 Member Typedef Documentation

22.33.2.1 typedef std::vector<BomAbstract*> AIRINV::FacBomAbstract::BomPool_T

Define the list (pool) of Bom objects.

Definition at line 22 of file FacBomAbstract.hpp.

22.33.3 Constructor & Destructor Documentation

22.33.3.1 AIRINV::FacBomAbstract::FacBomAbstract ( ) [inline], [protected]

Default Constructor.

This constructor is protected to ensure the class is abstract.

Definition at line 41 of file FacBomAbstract.hpp.

22.33.3.2 AIRINV::FacBomAbstract::FacBomAbstract (const FacBomAbstract &) [inline], [protected]

Definition at line 42 of file FacBomAbstract.hpp.

22.33.3.3 AIRINV::FacBomAbstract::~FacBomAbstract ( ) [protected], [virtual]

Destructor.

Definition at line 16 of file FacBomAbstract.cpp.

22.33.4 Member Function Documentation

22.33.4.1 std::size_t AIRINV::FacBomAbstract::getID (const BomAbstract * *iBomAbstract_ptr*) [static]

Return the ID corresponding to the given object pointer.

Definition at line 35 of file FacBomAbstract.cpp.

Referenced by getID(), and getIDString().

22.33.4.2 std::size_t AIRINV::FacBomAbstract::getID (const BomAbstract & *iBomAbstract*) [static]

Return the ID corresponding to the given object reference.

Definition at line 43 of file FacBomAbstract.cpp.

References getID().

22.33.4.3 std::string AIRINV::FacBomAbstract::getIDString (const BomAbstract * *iBomAbstract_ptr*) [static]

Return the ID, as a string, corresponding to the given object pointer.

Definition at line 48 of file FacBomAbstract.cpp.

References getID().

Referenced by getIDString().

22.33.4.4 std::string AIRINV::FacBomAbstract::getIDString (const BomAbstract & *iBomAbstract*) [static]

Return the ID, as a string, corresponding to the given object reference.

Definition at line 56 of file FacBomAbstract.cpp.

References getIDString().

22.33.5 Friends And Related Function Documentation

22.33.5.1 friend class FacSupervisor [friend]

Definition at line 18 of file FacBomAbstract.hpp.

22.33.6 Member Data Documentation

22.33.6.1 BomPool_T AIRINV::FacBomAbstract::_pool [protected]

List of instantiated Business Objects

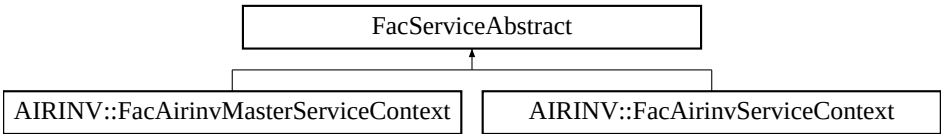
Definition at line 53 of file FacBomAbstract.hpp.

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

- airinv/factory/FacBomAbstract.hpp
- airinv/factory/FacBomAbstract.cpp

22.34 FacServiceAbstract Class Reference

Inheritance diagram for FacServiceAbstract:



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

- airinv/factory/FacAirinvMasterServiceContext.hpp

22.35 AIRINV::FacServiceAbstract Class Reference

`#include <airinv/factory/FacServiceAbstract.hpp>`

Public Types

- typedef std::vector
  < ServiceAbstract * > ServicePool_T

Public Member Functions

- virtual ~FacServiceAbstract ()
- void clean ()

Protected Member Functions

- FacServiceAbstract ()

Protected Attributes

- ServicePool_T _pool

22.35.1 Detailed Description

Base class for the (Service) Factory layer.

Definition at line 16 of file FacServiceAbstract.hpp.

22.35.2 Member Typedef Documentation

22.35.2.1 typedef std::vector<ServiceAbstract*> AIRINV::FacServiceAbstract::ServicePool_T

Define the list (pool) of Service objects.

Definition at line 20 of file FacServiceAbstract.hpp.

22.35.3 Constructor & Destructor Documentation

22.35.3.1 AIRINV::FacServiceAbstract::~FacServiceAbstract ( ) [virtual]

Destructor.

Definition at line 13 of file FacServiceAbstract.cpp.

References clean().

22.35.3.2 AIRINV::FacServiceAbstract::FacServiceAbstract ( ) [inline], [protected]

Default Constructor.

This constructor is protected to ensure the class is abstract.

Definition at line 31 of file FacServiceAbstract.hpp.

22.35.4 Member Function Documentation

22.35.4.1 void AIRINV::FacServiceAbstract::clean ( )

Destroyed all the object instantiated by this factory.

Definition at line 18 of file FacServiceAbstract.cpp.

References _pool.

Referenced by ~FacServiceAbstract().

22.35.5 Member Data Documentation

22.35.5.1 ServicePool_T AIRINV::FacServiceAbstract::_pool [protected]

List of instantiated Business Objects

Definition at line 34 of file FacServiceAbstract.hpp.

Referenced by clean().

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

- airinv/factory/FacServiceAbstract.hpp
- airinv/factory/FacServiceAbstract.cpp

22.36 AIRINV::FacSupervisor Class Reference

`#include <airinv/factory/FacSupervisor.hpp>`

Public Types

- typedef std::vector < FacBomAbstract * > BomFactoryPool_T
- typedef std::vector < FacServiceAbstract * > ServiceFactoryPool_T

Public Member Functions

- void registerBomFactory (FacBomAbstract *)
- void registerServiceFactory (FacServiceAbstract *)
- void cleanBomLayer ()
- void cleanServiceLayer ()
- ~FacSupervisor ()

Static Public Member Functions

- static FacSupervisor & instance ()
- static void cleanFactory ()

Protected Member Functions

- FacSupervisor ()
- FacSupervisor (const FacSupervisor &)

22.36.1 Detailed Description

Singleton class to register and clean all Factories.

Definition at line 17 of file FacSupervisor.hpp.

22.36.2 Member Typedef Documentation

22.36.2.1 typedef std::vector<FacBomAbstract*> AIRINV::FacSupervisor::BomFactoryPool_T

Define the pool (list) of factories.

Definition at line 21 of file FacSupervisor.hpp.

22.36.2.2 typedef std::vector<FacServiceAbstract*> AIRINV::FacSupervisor::ServiceFactoryPool_T

Definition at line 22 of file FacSupervisor.hpp.

22.36.3 Constructor & Destructor Documentation

22.36.3.1 AIRINV::FacSupervisor::~FacSupervisor ( )

Destructor

The static instance is deleted (and reset to NULL) by the static cleanFactory() method.

Definition at line 41 of file FacSupervisor.cpp.

References cleanBomLayer(), and cleanServiceLayer().

22.36.3.2 AIRINV::FacSupervisor::FacSupervisor ( ) [protected]

Default Constructor.

This constructor is protected to ensure the singleton pattern.

Definition at line 16 of file FacSupervisor.cpp.

Referenced by instance().

22.36.3.3 AIRINV::FacSupervisor::FacSupervisor (const FacSupervisor &) [inline], [protected]

Definition at line 66 of file FacSupervisor.hpp.

22.36.4 Member Function Documentation

22.36.4.1 FacSupervisor & AIRINV::FacSupervisor::instance ( ) [static]

Provides the unique instance.

The singleton is instantiated when first used.

Returns

FacSupervisor&

Definition at line 20 of file FacSupervisor.cpp.

References FacSupervisor().

22.36.4.2 void AIRINV::FacSupervisor::registerBomFactory (FacBomAbstract * *ioFacBomAbstract_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

FacAbstract&	the concrete Factory to register.

Definition at line 30 of file FacSupervisor.cpp.

22.36.4.3 void AIRINV::FacSupervisor::registerServiceFactory (FacServiceAbstract * *ioFacServiceAbstract_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

FacService-Abstract&	the concrete Factory to register.

Definition at line 36 of file FacSupervisor.cpp.

22.36.4.4 void AIRINV::FacSupervisor::cleanBomLayer ( )

Clean all created object.

Call the clean method of all the instantiated factories for the Bom layer.

Definition at line 47 of file FacSupervisor.cpp.

Referenced by cleanFactory(), and ~FacSupervisor().

22.36.4.5 void AIRINV::FacSupervisor::cleanServiceLayer ( )

Clean all Service created object.

Call the clean method of all the instantiated factories for the Service layer.

Definition at line 61 of file FacSupervisor.cpp.

Referenced by cleanFactory(), and ~FacSupervisor().

22.36.4.6 void AIRINV::FacSupervisor::cleanFactory ( ) [static]

Clean the static instance.

The singleton is deleted.

Definition at line 75 of file FacSupervisor.cpp.

References cleanBomLayer(), and cleanServiceLayer().

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

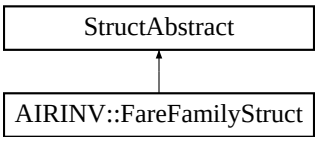
- airinv/factory/FacSupervisor.hpp
- airinv/factory/FacSupervisor.cpp

22.37 AIRINV::FareFamilyStruct Struct Reference

Utility Structure for the parsing of fare family details.

`#include <airinv/bom/FareFamilyStruct.hpp>`

Inheritance diagram for AIRINV::FareFamilyStruct:



Public Member Functions

- FareFamilyStruct ()
- FareFamilyStruct (const stdair::FamilyCode_T &, const stdair::CurveKey_T &, const stdair::CurveKey_T &, const stdair::ClassList_String_T &)
- void fill (stdair::FareFamily &) const
- const std::string describe () const

Public Attributes

- stdair::FamilyCode_T _familyCode

- stdair::CurveKey_T _frat5CurveKey
- stdair::CurveKey_T _ffDisutilityCurveKey
- stdair::ClassList_String_T _classes
- BookingClassStructList_T _classList

22.37.1 Detailed Description

Utility Structure for the parsing of fare family details.

Definition at line 26 of file FareFamilyStruct.hpp.

22.37.2 Constructor & Destructor Documentation

22.37.2.1 AIRINV::FareFamilyStruct::FareFamilyStruct ( )

Default constructor.

Definition at line 16 of file FareFamilyStruct.cpp.

22.37.2.2 AIRINV::FareFamilyStruct::FareFamilyStruct (const stdair::FamilyCode_T & *iFamilyCode,* const stdair::CurveKey_T & *iFRAT5Key,* const stdair::CurveKey_T & *iFFDisutilityKey,* const stdair::ClassList_String_T & *iClasses*)

Main constructor.

Definition at line 23 of file FareFamilyStruct.cpp.

22.37.3 Member Function Documentation

22.37.3.1 void AIRINV::FareFamilyStruct::fill (stdair::FareFamily & *ioFareFamily*) const

Fill the FareFamily objects with the attributes of the FareFamilyStruct.

Definition at line 52 of file FareFamilyStruct.cpp.

22.37.3.2 const std::string AIRINV::FareFamilyStruct::describe ( ) const

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

Definition at line 32 of file FareFamilyStruct.cpp.

References _classes, _classList, _familyCode, _ffDisutilityCurveKey, _frat5CurveKey, and AIRINV::BookingClassStruct::describe().

Referenced by AIRINV::SegmentCabinStruct::describe().

22.37.4 Member Data Documentation

22.37.4.1 stdair::FamilyCode_T AIRINV::FareFamilyStruct::_familyCode

Definition at line 28 of file FareFamilyStruct.hpp.

Referenced by describe(), AIRINV::ScheduleParserHelper::storeFamilyCode::operator()(), AIRINV::ScheduleParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::storeFamilyCode::operator()().

22.37.4.2 stdair::CurveKey_T AIRINV::FareFamilyStruct::_frat5CurveKey

Definition at line 29 of file FareFamilyStruct.hpp.

Referenced by describe(), AIRINV::ScheduleParserHelper::storeFRAT5CurveKey::operator()(), and AIRINV::ScheduleParserHelper::storeFClasses::operator()().

22.37.4.3 stdair::CurveKey_T AIRINV::FareFamilyStruct::_ffDisutilityCurveKey

Definition at line 30 of file FareFamilyStruct.hpp.

Referenced by describe(), AIRINV::ScheduleParserHelper::storeFFDisutilityCurveKey::operator()(), and AIRINV::ScheduleParserHelper::storeFClasses::operator()().

22.37.4.4 stdair::ClassList_String_T AIRINV::FareFamilyStruct::_classes

Definition at line 31 of file FareFamilyStruct.hpp.

Referenced by describe(), AIRINV::ScheduleParserHelper::storeClasses::operator()(), and AIRINV::InventoryParserHelper::storeFClasses::operator()().

22.37.4.5 BookingClassStructList_T AIRINV::FareFamilyStruct::_classList

Definition at line 32 of file FareFamilyStruct.hpp.

Referenced by describe(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), and AIRINV::InventoryParserHelper::storeFClasses::operator()().

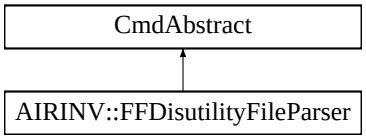
The documentation for this struct was generated from the following files:

- airinv/bom/FareFamilyStruct.hpp
- airinv/bom/FareFamilyStruct.cpp

22.38 AIRINV::FFDisutilityFileParser Class Reference

`#include <airinv/command/FFDisutilityParserHelper.hpp>`

Inheritance diagram for AIRINV::FFDisutilityFileParser:



Public Member Functions

- FFDisutilityFileParser (stdair::BomRoot &ioBomRoot, const stdair::Filename_T &iFilename)
- bool generateFFDisutilityCurves ()

22.38.1 Detailed Description

Short Description

Detailed Description. Class wrapping the initialisation and entry point of the parser.

The seemingly redundancy is used to force the instantiation of the actual parser, which is a templatised Boost Spirit grammar. Hence, the actual parser is instantiated within that class object code.

Definition at line 126 of file FFDisutilityParserHelper.hpp.

22.38.2 Constructor & Destructor Documentation

22.38.2.1 AIRINV::FFDisutilityFileParser::FFDisutilityFileParser (stdair::BomRoot & *ioBomRoot,* const stdair::Filename_T & *iFilename*)

Constructor.

Definition at line 177 of file FFDisutilityParserHelper.cpp.

22.38.3 Member Function Documentation

22.38.3.1 bool AIRINV::FFDisutilityFileParser::generateFFDisutilityCurves ( )

Parse the input file and generate the FFDisutility curves.

Definition at line 201 of file FFDisutilityParserHelper.cpp.

Referenced by AIRINV::FFDisutilityParser::parse().

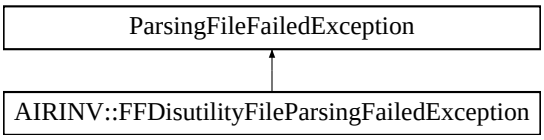
The documentation for this class was generated from the following files:

- airinv/command/FFDisutilityParserHelper.hpp
- airinv/command/FFDisutilityParserHelper.cpp

22.39 AIRINV::FFDisutilityFileParsingFailedException Class Reference

`#include <airinv/AIRINV_Types.hpp>`

Inheritance diagram for AIRINV::FFDisutilityFileParsingFailedException:



Public Member Functions

- FFDisutilityFileParsingFailedException (const std::string &iWhat)

22.39.1 Detailed Description

The FFDisutility input file can not be parsed.

Definition at line 80 of file AIRINV_Types.hpp.

22.39.2 Constructor & Destructor Documentation

22.39.2.1 AIRINV::FFDisutilityFileParsingFailedException::FFDisutilityFileParsingFailedException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 86 of file AIRINV_Types.hpp.

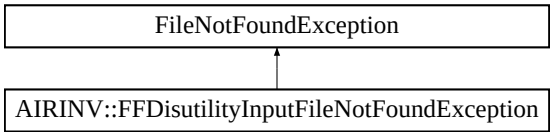
The documentation for this class was generated from the following file:

- airinv/AIRINV_Types.hpp

22.40 AIRINV::FFDisutilityInputFileNotFoundException Class Reference

`#include <airinv/AIRINV_Types.hpp>`

Inheritance diagram for AIRINV::FFDisutilityInputFileNotFoundException:



Public Member Functions

- FFDisutilityInputFileNotFoundException (const std::string &iWhat)

22.40.1 Detailed Description

The FF disutility input file can not be found or opened.

Definition at line 142 of file AIRINV_Types.hpp.

22.40.2 Constructor & Destructor Documentation

22.40.2.1 AIRINV::FFDisutilityInputFileNotFoundException::FFDisutilityInputFileNotFoundException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 147 of file AIRINV_Types.hpp.

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

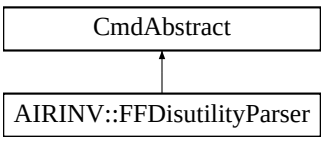
- airinv/AIRINV_Types.hpp

22.41 AIRINV::FFDisutilityParser Class Reference

Class wrapping the parser entry point.

`#include <airinv/command/FFDisutilityParser.hpp>`

Inheritance diagram for AIRINV::FFDisutilityParser:



Static Public Member Functions

- static void parse (const stdair::FFDisutilityFilePath &iFFDisutilityInputFilename, stdair::BomRoot &)

22.41.1 Detailed Description

Class wrapping the parser entry point.

Definition at line 22 of file FFDisutilityParser.hpp.

22.41.2 Member Function Documentation

22.41.2.1 void AIRINV::FFDisutilityParser::parse (const stdair::FFDisutilityFilePath & *iFFDisutilityInputFilename*, stdair::BomRoot & *ioBomRoot*) [static]

Parse the CSV file describing the FFDisutility curves for the simulator, and generates the curves accordingly.

Parameters

const	stdair::Filename_T& The file-name of the CSV-formatted FFDisutility curve input file.
stdair::BomRoot&	Root of the BOM tree.

Definition at line 20 of file FFDisutilityParser.cpp.

References AIRINV::FFDisutilityFileParser::generateFFDisutilityCurves().

Referenced by AIRINV::AIRINV_Service::parseAndLoad().

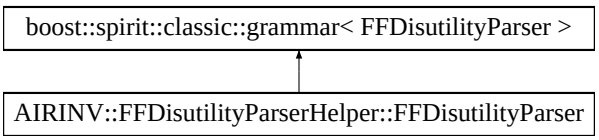
The documentation for this class was generated from the following files:

- airinv/command/FFDisutilityParser.hpp
- airinv/command/FFDisutilityParser.cpp

22.42 AIRINV::FFDisutilityParserHelper::FFDisutilityParser Struct Reference

`#include <airinv/command/FFDisutilityParserHelper.hpp>`

Inheritance diagram for AIRINV::FFDisutilityParserHelper::FFDisutilityParser:



Classes

- struct definition

Public Member Functions

- FFDisutilityParser (stdair::BomRoot &, FFDisutilityStruct &)

Public Attributes

- stdair::BomRoot & _bomRoot
- FFDisutilityStruct & _ffDisutility

22.42.1 Detailed Description

CurveKey; FFDisutilityCurve; D1; 63:0.0050; 56:0.0049; 49:0.0047; 42:0.0045; 35:0.0043; 31:0.0040; 27:0.0037; 23:0.0034; 19:0.0030; 16:0.0026; 13:0.0022; 10:0.0017; 7:0.0013; 5:0.0012; 3:0.0011; 1:0.0010;

Grammar: Curve ::= Key ';' FFDisutilityMap EndOfCurve Key ::= string FFDisutilityMap ::= DTDValuePaire (';' DTDValuePaire)* DTDValuePaire ::= DTD ':' FFDisutilityValue EndOfCurve ::= ';' Grammar for the FFDisutility curve parser.

Definition at line 89 of file FFDisutilityParserHelper.hpp.

22.42.2 Constructor & Destructor Documentation

22.42.2.1 AIRINV::FFDisutilityParserHelper::FFDisutilityParser::FFDisutilityParser (stdair::BomRoot & *ioBomRoot*, FFDisutilityStruct & *ioFFDisutility*)

Definition at line 114 of file FFDisutilityParserHelper.cpp.

22.42.3 Member Data Documentation

22.42.3.1 stdair::BomRoot& AIRINV::FFDisutilityParserHelper::FFDisutilityParser::_bomRoot

Definition at line 107 of file FFDisutilityParserHelper.hpp.

22.42.3.2 FFDisutilityStruct& AIRINV::FFDisutilityParserHelper::FFDisutilityParser::_ffDisutility

Definition at line 108 of file FFDisutilityParserHelper.hpp.

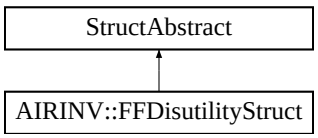
The documentation for this struct was generated from the following files:

- airinv/command/FFDisutilityParserHelper.hpp
- airinv/command/FFDisutilityParserHelper.cpp

22.43 AIRINV::FFDisutilityStruct Struct Reference

`#include <airinv/bom/FFDisutilityStruct.hpp>`

Inheritance diagram for AIRINV::FFDisutilityStruct:



Public Member Functions

- const std::string describe () const
- FFDisutilityStruct ()
- ~FFDisutilityStruct ()

Public Attributes

- std::string _key
- stdair::FFDisutilityCurve_T _curve
- stdair::DTD_T _dtd

22.43.1 Detailed Description

Utility Structure for the parsing of FFDisutility structures.

Definition at line 15 of file FFDisutilityStruct.hpp.

22.43.2 Constructor & Destructor Documentation

22.43.2.1 AIRINV::FFDisutilityStruct::FFDisutilityStruct ( )

Default constructor.

Definition at line 15 of file FFDisutilityStruct.cpp.

22.43.2.2 AIRINV::FFDisutilityStruct::~FFDisutilityStruct ( )

Destructor

Definition at line 19 of file FFDisutilityStruct.cpp.

22.43.3 Member Function Documentation

22.43.3.1 const std::string AIRINV::FFDisutilityStruct::describe ( ) const

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

Definition at line 23 of file FFDisutilityStruct.cpp.

References _curve, and _key.

Referenced by AIRINV::FFDisutilityParserHelper::doEndCurve::operator()().

22.43.4 Member Data Documentation

22.43.4.1 std::string AIRINV::FFDisutilityStruct::_key

Curve key.

Definition at line 37 of file FFDisutilityStruct.hpp.

Referenced by describe(), AIRINV::FFDisutilityParserHelper::storeCurveKey::operator()(), and AIRINV::FFDisutilityParserHelper::doEndCurve::operator()().

22.43.4.2 stdair::FFDisutilityCurve_T AIRINV::FFDisutilityStruct::_curve

Curve.

Definition at line 40 of file FFDisutilityStruct.hpp.

Referenced by describe(), AIRINV::FFDisutilityParserHelper::storeFFDisutilityValue::operator()(), and AIRINV::FFDisutilityParserHelper::doEndCurve::operator()().

22.43.4.3 stdair::DTD_T AIRINV::FFDisutilityStruct::_dtd

DTD.

Definition at line 45 of file FFDisutilityStruct.hpp.

Referenced by AIRINV::FFDisutilityParserHelper::storeDTD::operator()(), and AIRINV::FFDisutilityParserHelper::storeFFDisutilityValue::operator()().

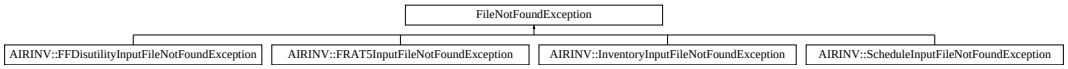
The documentation for this struct was generated from the following files:

- airinv/bom/FFDisutilityStruct.hpp

- airinv/bom/FFDisutilityStruct.cpp

22.44 FileNotFoundException Class Reference

Inheritance diagram for FileNotFoundException:



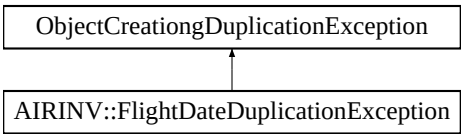
The documentation for this class was generated from the following file:

- airinv/AIRINV_Types.hpp

22.45 AIRINV::FlightDateDuplicationException Class Reference

`#include <airinv/AIRINV_Types.hpp>`

Inheritance diagram for AIRINV::FlightDateDuplicationException:



Public Member Functions

- FlightDateDuplicationException (const std::string &iWhat)

22.45.1 Detailed Description

Duplicated flight date object.

Definition at line 154 of file AIRINV_Types.hpp.

22.45.2 Constructor & Destructor Documentation

22.45.2.1 AIRINV::FlightDateDuplicationException::FlightDateDuplicationException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 159 of file AIRINV_Types.hpp.

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

- airinv/AIRINV_Types.hpp

22.46 AIRINV::FlightDateHelper Class Reference

`#include <airinv/bom/FlightDateHelper.hpp>`

Static Public Member Functions

- static void fillFromRouting (const stdair::FlightDate &)
- static void updateAvailability (const stdair::FlightDate &, const stdair::SegmentCabin &, const stdair::PartySize_T &iNbOfBookings)
- static void updateAvailabilityPool (const stdair::FlightDate &, const stdair::CabinCode_T &)
- static void recalculateAvailability (const stdair::FlightDate &, const stdair::CabinCode_T &)
- static void updateBookingControls (stdair::FlightDate &)
- static void recalculateAvailability (const stdair::FlightDate &)

22.46.1 Detailed Description

Class representing the actual business functions for an airline flight-date.

Definition at line 19 of file FlightDateHelper.hpp.

22.46.2 Member Function Documentation

22.46.2.1 void AIRINV::FlightDateHelper::fillFromRouting (const stdair::FlightDate & *iFlightDate*) [static]

Fill the attributes derived from the routing legs (e.g., board and off dates).

Definition at line 51 of file FlightDateHelper.cpp.

22.46.2.2 void AIRINV::FlightDateHelper::updateAvailability (const stdair::FlightDate & *iFlightDate,* const stdair::SegmentCabin & *iSegmentCabin,* const stdair::PartySize_T & *iNbOfBookings*) [static]

Update the availability of all classes after a reservation.

Definition at line 67 of file FlightDateHelper.cpp.

References recalculateAvailability(), and updateAvailabilityPool().

Referenced by AIRINV::SegmentCabinHelper::updateFromReservation().

22.46.2.3 void AIRINV::FlightDateHelper::updateAvailabilityPool (const stdair::FlightDate & *iFlightDate,* const stdair::CabinCode_T & *iCabinCode*) [static]

Update the availability pool of all the segment-cabins after a reservation.

Definition at line 92 of file FlightDateHelper.cpp.

Referenced by updateAvailability().

22.46.2.4 void AIRINV::FlightDateHelper::recalculateAvailability (const stdair::FlightDate & *iFlightDate,* const stdair::CabinCode_T & *iCabinCode*) [static]

Recalculate the availability of all the segment-cabins after a reservation.

Definition at line 127 of file FlightDateHelper.cpp.

References AIRINV::SegmentCabinHelper::updateAvailabilities().

Referenced by updateAvailability().

22.46.2.5 void AIRINV::FlightDateHelper::updateBookingControls (stdair::FlightDate & *ioFlightDate*) [static]

Update booking controls after optimisation.

Definition at line 22 of file FlightDateHelper.cpp.

References AIRINV::SegmentCabinHelper::buildPseudoBidPriceVector(), and AIRINV::SegmentCabinHelper::updateBookingControlsUsingPseudoBidPriceVector().

22.46.2.6 void AIRINV::FlightDateHelper::recalculateAvailability (const stdair::FlightDate & *iFlightDate*) [static]

Recalculate the availability of all the segment-cabins after an optimisation.

Definition at line 145 of file FlightDateHelper.cpp.

References AIRINV::SegmentCabinHelper::updateAvailabilities().

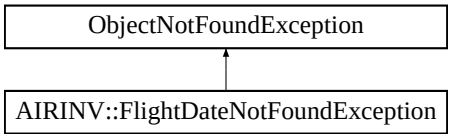
The documentation for this class was generated from the following files:

- airinv/bom/FlightDateHelper.hpp
- airinv/bom/FlightDateHelper.cpp

22.47 AIRINV::FlightDateNotFoundException Class Reference

`#include <airinv/AIRINV_Types.hpp>`

Inheritance diagram for AIRINV::FlightDateNotFoundException:



Public Member Functions

- FlightDateNotFoundException (const std::string &iWhat)

22.47.1 Detailed Description

Flight Date not found

Definition at line 184 of file AIRINV_Types.hpp.

22.47.2 Constructor & Destructor Documentation

22.47.2.1 AIRINV::FlightDateNotFoundException::FlightDateNotFoundException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 189 of file AIRINV_Types.hpp.

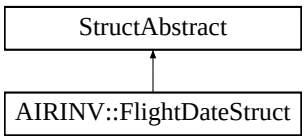
The documentation for this class was generated from the following file:

- airinv/AIRINV_Types.hpp

22.48 AIRINV::FlightDateStruct Struct Reference

`#include <airinv/bom/FlightDateStruct.hpp>`

Inheritance diagram for AIRINV::FlightDateStruct:



Public Member Functions

- stdair::Date_T getDate () const
- stdair::Duration_T getTime () const
- const std::string describe () const
- void addAirport (const stdair::AirportCode_T &)
- void buildSegments ()
- void addSegmentCabin (const SegmentStruct &, const SegmentCabinStruct &)
- void addSegmentCabin (const SegmentCabinStruct &)
- void addFareFamily (const SegmentStruct &, const SegmentCabinStruct &, const FareFamilyStruct &)
- void addFareFamily (const SegmentCabinStruct &, const FareFamilyStruct &)
- FlightDateStruct ()

Public Attributes

- stdair::AirlineCode_T _airlineCode
- stdair::FlightNumber_T _flightNumber
- stdair::Date_T _flightDate
- FlightTypeCode _flightTypeCode
- FlightVisibilityCode _flightVisibilityCode
- LegStructList_T _legList
- SegmentStructList_T _segmentList
- unsigned int _itYear
- unsigned int _itMonth
- unsigned int _itDay
- int _dateOffSet
- long _itHours
- long _itMinutes
- long _itSeconds
- AirportList_T _airportList
- AirportOrderedList_T _airportOrderedList
- bool _legAlreadyDefined
- LegStruct _itLeg
- LegCabinStruct _itLegCabin
- BucketStruct _itBucket
- bool _areSegmentDefinitionsSpecific
- SegmentStruct _itSegment
- SegmentCabinStruct _itSegmentCabin
- BookingClassStruct _itBookingClass

22.48.1 Detailed Description

Utility Structure for the parsing of Flight-Date structures.

Definition at line 27 of file FlightDateStruct.hpp.

22.48.2 Constructor & Destructor Documentation

22.48.2.1 AIRINV::FlightDateStruct::FlightDateStruct ( )

Constructor.

Definition at line 17 of file FlightDateStruct.cpp.

22.48.3 Member Function Documentation

22.48.3.1 stdair::Date_T AIRINV::FlightDateStruct::getDate ( ) const

Set the date from the staging details.

Definition at line 25 of file FlightDateStruct.cpp.

References _itDay, _itMonth, and _itYear.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), and AIRINV::InventoryParserHelper::storeOffDate::operator()().

22.48.3.2 stdair::Duration_T AIRINV::FlightDateStruct::getTime ( ) const

Set the time from the staging details.

Definition at line 30 of file FlightDateStruct.cpp.

References _itHours, _itMinutes, and _itSeconds.

Referenced by AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), and AIRINV::InventoryParserHelper::storeOffTime::operator()().

22.48.3.3 const std::string AIRINV::FlightDateStruct::describe ( ) const

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

Definition at line 37 of file FlightDateStruct.cpp.

References _airlineCode, _flightDate, _flightNumber, _flightTypeCode, _flightVisibilityCode, _legList, _segmentList, AIRINV::SegmentStruct::describe(), AIRINV::LegStruct::describe(), AIRINV::FlightVisibilityCode::getCode(), and AIRINV::FlightVisibilityCode::NORMAL.

22.48.3.4 void AIRINV::FlightDateStruct::addAirport (const stdair::AirportCode_T & *iAirport*)

Add the given airport to the internal lists (if not already existing).

Definition at line 67 of file FlightDateStruct.cpp.

References _airportList, and _airportOrderedList.

Referenced by AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), and AIRINV::InventoryParserHelper::storeLegOffPoint::operator()().

22.48.3.5 void AIRINV::FlightDateStruct::buildSegments ( )

Build the list of SegmentStruct objects.

Definition at line 83 of file FlightDateStruct.cpp.

References _airportList, _airportOrderedList, AIRINV::SegmentStruct::_boardingPoint, AIRINV::SegmentStruct::_offPoint, and _segmentList.

22.48.3.6 void AIRINV::FlightDateStruct::addSegmentCabin (const SegmentStruct & *iSegment,* const SegmentCabinStruct & *iCabin*)

Add, to the Segment structure whose key corresponds to the given (board point, off point) pair, the specific segment cabin details (mainly, the list of the class codes).

Note that the Segment structure is retrieved from the internal list, already filled by a previous step (the buildSegments() method).

Definition at line 116 of file FlightDateStruct.cpp.

References AIRINV::SegmentStruct::_boardingPoint, AIRINV::SegmentStruct::_cabinList, AIRINV::SegmentStruct::_offPoint, and _segmentList.

22.48.3.7 void AIRINV::FlightDateStruct::addSegmentCabin (const SegmentCabinStruct & *iCabin*)

Add, to all the Segment structures, the general segment cabin details (mainly, the list of the class codes).

Note that the Segment structures are stored within the internal list, already filled by a previous step (the buildSegments() method).

Definition at line 153 of file FlightDateStruct.cpp.

References AIRINV::SegmentStruct::_cabinList, and _segmentList.

22.48.3.8 void AIRINV::FlightDateStruct::addFareFamily (const SegmentStruct & *iSegment,* const SegmentCabinStruct & *iCabin,* const FareFamilyStruct & *iFareFamily*)

Add, to the SegmentCabin structure whose key corresponds to the given cabin code, the specific segment fare family details (mainly, the list of the class codes).

Note that the SegmentCabin structure is retrieved from the internal list, already filled by a previous step (the buildSegmentCabins() method).

Definition at line 167 of file FlightDateStruct.cpp.

References AIRINV::SegmentStruct::_boardingPoint, AIRINV::SegmentCabinStruct::_cabinCode, AIRINV::SegmentStruct::_cabinList, AIRINV::SegmentCabinStruct::_fareFamilies, AIRINV::SegmentStruct::_offPoint, and _segmentList.

22.48.3.9 void AIRINV::FlightDateStruct::addFareFamily (const SegmentCabinStruct & *iCabin,* const FareFamilyStruct & *iFareFamily*)

Add, to all the Segment structures, the general fare family sets (list of fare families).

Note that the SegmentCabin structures are stored within the internal list, already filled by a previous step (the buildSegmentCabins() method).

Definition at line 231 of file FlightDateStruct.cpp.

References AIRINV::SegmentCabinStruct::_cabinCode, AIRINV::SegmentStruct::_cabinList, AIRINV::SegmentCabinStruct::_fareFamilies, and _segmentList.

22.48.4 Member Data Documentation

22.48.4.1 stdair::AirlineCode_T AIRINV::FlightDateStruct::_airlineCode

Definition at line 81 of file FlightDateStruct.hpp.

Referenced by describe(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), and AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()().

22.48.4.2 stdair::FlightNumber_T AIRINV::FlightDateStruct::_flightNumber

Definition at line 82 of file FlightDateStruct.hpp.

Referenced by describe(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), and AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()().

22.48.4.3 stdair::Date_T AIRINV::FlightDateStruct::_flightDate

Definition at line 83 of file FlightDateStruct.hpp.

Referenced by describe(), AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), and AIRINV::InventoryParserHelper::storeFlightDate::operator()().

22.48.4.4 FlightTypeCode AIRINV::FlightDateStruct::_flightTypeCode

Definition at line 84 of file FlightDateStruct.hpp.

Referenced by describe(), and AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()().

22.48.4.5 FlightVisibilityCode AIRINV::FlightDateStruct::_flightVisibilityCode

Definition at line 85 of file FlightDateStruct.hpp.

Referenced by describe(), and AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()().

22.48.4.6 LegStructList_T AIRINV::FlightDateStruct::_legList

Definition at line 86 of file FlightDateStruct.hpp.

Referenced by describe(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), and AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()().

22.48.4.7 SegmentStructList_T AIRINV::FlightDateStruct::_segmentList

Definition at line 87 of file FlightDateStruct.hpp.

Referenced by addFareFamily(), addSegmentCabin(), buildSegments(), describe(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

22.48.4.8 unsigned int AIRINV::FlightDateStruct::_itYear

Staging Date.

Definition at line 90 of file FlightDateStruct.hpp.

Referenced by getDate().

22.48.4.9 unsigned int AIRINV::FlightDateStruct::_itMonth

Definition at line 91 of file FlightDateStruct.hpp.

Referenced by getDate().

22.48.4.10 unsigned int AIRINV::FlightDateStruct::_itDay

Definition at line 92 of file FlightDateStruct.hpp.

Referenced by getDate().

22.48.4.11 int AIRINV::FlightDateStruct::_dateOffSet

Definition at line 93 of file FlightDateStruct.hpp.

Referenced by AIRINV::InventoryParserHelper::storeBoardingTime::operator()().

22.48.4.12 long AIRINV::FlightDateStruct::_itHours

Staging Time.

Definition at line 96 of file FlightDateStruct.hpp.

Referenced by getTime().

22.48.4.13 long AIRINV::FlightDateStruct::_itMinutes

Definition at line 97 of file FlightDateStruct.hpp.

Referenced by getTime().

22.48.4.14 long AIRINV::FlightDateStruct::_itSeconds

Definition at line 98 of file FlightDateStruct.hpp.

Referenced by getTime(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), and AIRINV::InventoryParserHelper::storeOffTime::operator()().

22.48.4.15 AirportList_T AIRINV::FlightDateStruct::_airportList

Staging Airport List (helper to derive the list of Segment structures).

Definition at line 102 of file FlightDateStruct.hpp.

Referenced by addAirport(), and buildSegments().

22.48.4.16 AirportOrderedList_T AIRINV::FlightDateStruct::_airportOrderedList

Definition at line 103 of file FlightDateStruct.hpp.

Referenced by addAirport(), and buildSegments().

22.48.4.17 bool AIRINV::FlightDateStruct::_legAlreadyDefined

Staging Leg (resp. Cabin) structure, gathering the result of the iteration on one leg (resp. cabin).

Definition at line 107 of file FlightDateStruct.hpp.

22.48.4.18 LegStruct AIRINV::FlightDateStruct::_itLeg

Definition at line 108 of file FlightDateStruct.hpp.

Referenced by AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), and AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()().

22.48.4.19 LegCabinStruct AIRINV::FlightDateStruct::_itLegCabin

Definition at line 109 of file FlightDateStruct.hpp.

Referenced by AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), and AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()().

22.48.4.20 BucketStruct AIRINV::FlightDateStruct::_itBucket

Definition at line 110 of file FlightDateStruct.hpp.

Referenced by AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), and AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()().

22.48.4.21 bool AIRINV::FlightDateStruct::_areSegmentDefinitionsSpecific

Staging Segment-related attributes.

Definition at line 113 of file FlightDateStruct.hpp.

22.48.4.22 SegmentStruct AIRINV::FlightDateStruct::_itSegment

Definition at line 114 of file FlightDateStruct.hpp.

Referenced by AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

22.48.4.23 SegmentCabinStruct AIRINV::FlightDateStruct::_itSegmentCabin

Definition at line 115 of file FlightDateStruct.hpp.

Referenced by AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), and AIRINV::InventoryParserHelper::storeFClasses::operator()().

22.48.4.24 BookingClassStruct AIRINV::FlightDateStruct::_itBookingClass

Definition at line 116 of file FlightDateStruct.hpp.

Referenced by AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), and AIRINV::InventoryParserHelper::storeFClasses::operator()().

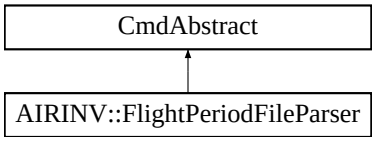
The documentation for this struct was generated from the following files:

- airinv/bom/FlightDateStruct.hpp
- airinv/bom/FlightDateStruct.cpp

22.49 AIRINV::FlightPeriodFileParser Class Reference

```
#include <airinv/command/ScheduleParserHelper.hpp>
```

Inheritance diagram for AIRINV::FlightPeriodFileParser:



Public Member Functions

- FlightPeriodFileParser (stdair::BomRoot &ioBomRoot, const stdair::Filename_T &iFilename)
- bool generateInventories ()

22.49.1 Detailed Description

Short Description

Detailed Description. Class wrapping the initialisation and entry point of the parser.

The seemingly redundancy is used to force the instantiation of the actual parser, which is a templatised Boost Spirit grammar. Hence, the actual parser is instantiated within that class object code.

Definition at line 325 of file ScheduleParserHelper.hpp.

22.49.2 Constructor & Destructor Documentation

22.49.2.1 AIRINV::FlightPeriodFileParser::FlightPeriodFileParser (stdair::BomRoot & *ioBomRoot,* const stdair::Filename_T & *iFilename*)

Constructor.

Definition at line 706 of file ScheduleParserHelper.cpp.

22.49.3 Member Function Documentation

22.49.3.1 bool AIRINV::FlightPeriodFileParser::generateInventories ( )

Parse the input file and generate the Inventories.

Definition at line 730 of file ScheduleParserHelper.cpp.

Referenced by AIRINV::ScheduleParser::generateInventories().

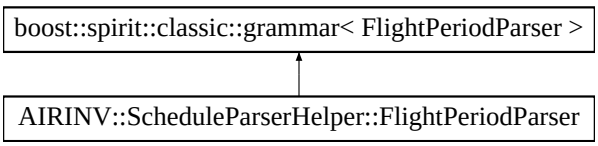
The documentation for this class was generated from the following files:

- airinv/command/ScheduleParserHelper.hpp
- airinv/command/ScheduleParserHelper.cpp

22.50 AIRINV::ScheduleParserHelper::FlightPeriodParser Struct Reference

`#include <airinv/command/ScheduleParserHelper.hpp>`

Inheritance diagram for AIRINV::ScheduleParserHelper::FlightPeriodParser:



Classes

- struct definition

Public Member Functions

- FlightPeriodParser (stdair::BomRoot &, FlightPeriodStruct &)

Public Attributes

- stdair::BomRoot & _bomRoot
- FlightPeriodStruct & _flightPeriod

22.50.1 Detailed Description

AirlineCode; FlightNumber; DateRangeStart; DateRangeEnd; DOW; (list) BoardingPoint; OffPoint; BoardingTime; DateOffset; OffTime; ElapsedTime; (list) CabinCode; Capacity; SegmentSpecificty (0 or 1); (list) (optional BoardingPoint; OffPoint); CabinCode; Classes BA; 9; 2007-04-20; 2007-04-30; 0000011; LHR; BKK; 22:00; +1; 15:15; 11:15; C; 12; M; 300; BKK; SYD; 18:10; +1; 06:05; 08:55; C; 20; M; 250; 0; C; CDIU; 1; CD; 2; IU; M; YHBKLMNOPQRSTVWX; 3; YHBKLMNOPQRSTVWX BA; 9; 2007-04-20; 2007-04-30; 1111100; LHR; SIN; 22:00; +1; 15:15; 11:15; C; 15; M; 310; SIN; SYD; 18:10; +1; 06:05; 08:55; C; 25; M; 260; 1; LHR; SIN; C; CDIU; 1; CDIU; M; YHBKLMNOPQRSTVWX; 2;YHBKLMNOPQRSTVWX SIN; SYD; C; CDIU; 1; CDIU; M; YHBKLMNOPQRSTVWX; 2;YHBKLMNOPQRSTVWX LHR; SYD; C; CDIU; 1; CDIU; M; YHBKLMNOPQRSTVWX; 2;YHBKLMNOPQRSTVWX

Grammar: DOW ::= int FlightKey ::= AirlineCode ';' FlightNumber ';' DateRangeStart ';' DateRangeEnd ';' DOW LegKey ::= BoardingPoint ';' OffPoint LegDetails ::= BoardingTime ['/' BoardingDateOffset] ';' OffTime ['/' BoardingDateOffset] ';' Elapsed LegCabinDetails ::= CabinCode ';' Capacity Leg ::= LegKey ';' LegDetails (';' CabinDetails)+ SegmentKey ::= BoardingPoint ';' OffPoint SegmentCabinDetails ::= CabinCode ';' Classes (';' FamilyCabinDetails)+ FamilyCabinDetails ::= FamilyCode ';' FRAT5Key ';' FFDisutilityKey ';" Classes FullSegmentCabinDetails::= (';' SegmentCabinDetails)+ GenericSegment ::= '0' (';' SegmentCabinDetails)+ SpecificSegments ::= '1' (';' SegmentKey ';' FullSegmentCabinDetails)+ SegmentSection ::= GenericSegment | SpecificSegments FlightPeriod ::= FlightKey (';' Leg)+ ';' SegmentSection ';' EndOfFlight EndOfFlight ::= ';' Grammar for the Flight-Period parser.

Definition at line 282 of file ScheduleParserHelper.hpp.

22.50.2 Constructor & Destructor Documentation

22.50.2.1 AIRINV::ScheduleParserHelper::FlightPeriodParser::FlightPeriodParser (stdair::BomRoot & *ioBomRoot*, FlightPeriodStruct & *ioFlightPeriod*)

Definition at line 529 of file ScheduleParserHelper.cpp.

22.50.3 Member Data Documentation

22.50.3.1 stdair::BomRoot& AIRINV::ScheduleParserHelper::FlightPeriodParser::_bomRoot

Definition at line 306 of file ScheduleParserHelper.hpp.

22.50.3.2 FlightPeriodStruct& AIRINV::ScheduleParserHelper::FlightPeriodParser::_flightPeriod

Definition at line 307 of file ScheduleParserHelper.hpp.

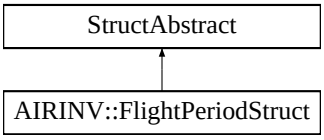
The documentation for this struct was generated from the following files:

- airinv/command/ScheduleParserHelper.hpp
- airinv/command/ScheduleParserHelper.cpp

22.51 AIRINV::FlightPeriodStruct Struct Reference

`#include <airinv/bom/FlightPeriodStruct.hpp>`

Inheritance diagram for AIRINV::FlightPeriodStruct:



Public Member Functions

- stdair::Date_T getDate () const
- stdair::Duration_T getTime () const
- const std::string describe () const
- void addAirport (const stdair::AirportCode_T &)
- void buildSegments ()
- void addSegmentCabin (const SegmentStruct &, const SegmentCabinStruct &)
- void addSegmentCabin (const SegmentCabinStruct &)
- void addFareFamily (const SegmentStruct &, const SegmentCabinStruct &, const FareFamilyStruct &)
- void addFareFamily (const SegmentCabinStruct &, const FareFamilyStruct &)
- FlightPeriodStruct ()

Public Attributes

- stdair::AirlineCode_T _airlineCode
- stdair::FlightNumber_T _flightNumber
- stdair::DatePeriod_T _dateRange
- stdair::DoWStruct _dow
- LegStructList_T _legList
- SegmentStructList_T _segmentList
- bool _legAlreadyDefined
- LegStruct _itLeg
- LegCabinStruct _itLegCabin
- stdair::Date_T _dateRangeStart
- stdair::Date_T _dateRangeEnd
- unsigned int _itYear
- unsigned int _itMonth
- unsigned int _itDay
- int _dateOffset
- long _itHours
- long _itMinutes
- long _itSeconds
- AirportList_T _airportList
- AirportOrderedList_T _airportOrderedList
- bool _areSegmentDefinitionsSpecific
- SegmentStruct _itSegment
- SegmentCabinStruct _itSegmentCabin

22.51.1 Detailed Description

Utility Structure for the parsing of Flight-Period structures.

Definition at line 24 of file FlightPeriodStruct.hpp.

22.51.2 Constructor & Destructor Documentation

22.51.2.1 AIRINV::FlightPeriodStruct::FlightPeriodStruct ( )

Constructor.

Definition at line 17 of file FlightPeriodStruct.cpp.

22.51.3 Member Function Documentation

22.51.3.1 stdair::Date_T AIRINV::FlightPeriodStruct::getDate ( ) const

Set the date from the staging details.

Definition at line 24 of file FlightPeriodStruct.cpp.

References _itDay, _itMonth, and _itYear.

Referenced by AIRINV::ScheduleParserHelper::storeDateRangeStart::operator()(), and AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator()().

22.51.3.2 stdair::Duration_T AIRINV::FlightPeriodStruct::getTime ( ) const

Set the time from the staging details.

Definition at line 29 of file FlightPeriodStruct.cpp.

References _itHours, _itMinutes, and _itSeconds.

Referenced by AIRINV::ScheduleParserHelper::storeBoardingTime::operator()(), AIRINV::ScheduleParserHelper::storeOffTime::operator()(), and AIRINV::ScheduleParserHelper::storeElapsedTime::operator()().

22.51.3.3 const std::string AIRINV::FlightPeriodStruct::describe ( ) const

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

Definition at line 36 of file FlightPeriodStruct.cpp.

References _airlineCode, _dateRange, _dow, _flightNumber, _legList, _segmentList, AIRINV::SegmentStruct::describe(), and AIRINV::LegStruct::describe().

Referenced by AIRINV::ScheduleParserHelper::doEndFlight::operator()().

22.51.3.4 void AIRINV::FlightPeriodStruct::addAirport (const stdair::AirportCode_T & *iAirport*)

Add the given airport to the internal lists (if not already existing).

Definition at line 62 of file FlightPeriodStruct.cpp.

References _airportList, and _airportOrderedList.

Referenced by AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator()(), and AIRINV::ScheduleParserHelper::storeLegOffPoint::operator()().

22.51.3.5 void AIRINV::FlightPeriodStruct::buildSegments ( )

Build the list of SegmentStruct objects.

Definition at line 78 of file FlightPeriodStruct.cpp.

References _airportList, _airportOrderedList, AIRINV::SegmentStruct::_boardingPoint, AIRINV::SegmentStruct::_offPoint, and _segmentList.

Referenced by AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator()().

22.51.3.6 void AIRINV::FlightPeriodStruct::addSegmentCabin (const SegmentStruct & *iSegment,* const SegmentCabinStruct & *iCabin*)

Add, to the Segment structure whose key corresponds to the given (board point, off point) pair, the specific segment cabin details (mainly, the list of the class codes).

Note that the Segment structure is retrieved from the internal list, already filled by a previous step (the buildSegments() method).

Definition at line 111 of file FlightPeriodStruct.cpp.

References AIRINV::SegmentStruct::_boardingPoint, AIRINV::SegmentStruct::_cabinList, AIRINV::SegmentStruct::_offPoint, and _segmentList.

Referenced by AIRINV::ScheduleParserHelper::storeClasses::operator()().

22.51.3.7 void AIRINV::FlightPeriodStruct::addSegmentCabin (const SegmentCabinStruct & *iCabin*)

Add, to all the Segment structures, the general segment cabin details (mainly, the list of the class codes).

Note that the Segment structures are stored within the internal list, already filled by a previous step (the buildSegments() method).

Definition at line 148 of file FlightPeriodStruct.cpp.

References AIRINV::SegmentStruct::_cabinList, and _segmentList.

22.51.3.8 void AIRINV::FlightPeriodStruct::addFareFamily (const SegmentStruct & *iSegment,* const SegmentCabinStruct & *iCabin,* const FareFamilyStruct & *iFareFamily*)

Add, to the SegmentCabin structure whose key corresponds to the given cabin code, the specific segment fare family details (mainly, the list of the class codes).

Note that the SegmentCabin structure is retrieved from the internal list, already filled by a previous step (the buildSegmentCabins() method).

Definition at line 161 of file FlightPeriodStruct.cpp.

References AIRINV::SegmentStruct::_boardingPoint, AIRINV::SegmentCabinStruct::_cabinCode, AIRINV::SegmentStruct::_cabinList, AIRINV::SegmentCabinStruct::_fareFamilies, AIRINV::SegmentStruct::_offPoint, and _segmentList.

Referenced by AIRINV::ScheduleParserHelper::storeFClasses::operator()().

22.51.3.9 void AIRINV::FlightPeriodStruct::addFareFamily (const SegmentCabinStruct & *iCabin,* const FareFamilyStruct & *iFareFamily*)

Add, to all the Segment structures, the general fare family sets (list of fare families).

Note that the SegmentCabin structures are stored within the internal list, already filled by a previous step (the buildSegmentCabins() method).

Definition at line 225 of file FlightPeriodStruct.cpp.

References AIRINV::SegmentCabinStruct::_cabinCode, AIRINV::SegmentStruct::_cabinList, AIRINV::SegmentCabinStruct::_fareFamilies, and _segmentList.

22.51.4 Member Data Documentation

22.51.4.1 stdair::AirlineCode_T AIRINV::FlightPeriodStruct::_airlineCode

Definition at line 80 of file FlightPeriodStruct.hpp.

Referenced by describe(), AIRINV::ScheduleParserHelper::storeAirlineCode::operator()(), and AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator()().

22.51.4.2 stdair::FlightNumber_T AIRINV::FlightPeriodStruct::_flightNumber

Definition at line 81 of file FlightPeriodStruct.hpp.

Referenced by describe(), AIRINV::ScheduleParserHelper::storeFlightNumber::operator()(), and AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator()().

22.51.4.3 stdair::DatePeriod_T AIRINV::FlightPeriodStruct::_dateRange

Definition at line 82 of file FlightPeriodStruct.hpp.

Referenced by describe(), and AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator()().

22.51.4.4 stdair::DoWStruct AIRINV::FlightPeriodStruct::_dow

Definition at line 83 of file FlightPeriodStruct.hpp.

Referenced by describe(), and AIRINV::ScheduleParserHelper::storeDow::operator()().

22.51.4.5 LegStructList_T AIRINV::FlightPeriodStruct::_legList

Definition at line 84 of file FlightPeriodStruct.hpp.

Referenced by describe(), AIRINV::ScheduleParserHelper::storeAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator()(), and AIRINV::ScheduleParserHelper::doEndFlight::operator()().

22.51.4.6 SegmentStructList_T AIRINV::FlightPeriodStruct::_segmentList

Definition at line 85 of file FlightPeriodStruct.hpp.

Referenced by addFareFamily(), addSegmentCabin(), buildSegments(), and describe().

22.51.4.7 bool AIRINV::FlightPeriodStruct::_legAlreadyDefined

Staging Leg (resp. Cabin) structure, gathering the result of the iteration on one leg (resp. cabin).

Definition at line 89 of file FlightPeriodStruct.hpp.

Referenced by AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator()(), and AIRINV::ScheduleParserHelper::doEndFlight::operator()().

22.51.4.8 LegStruct AIRINV::FlightPeriodStruct::_itLeg

Definition at line 90 of file FlightPeriodStruct.hpp.

Referenced by AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeBoardingTime::operator()(), AIRINV::ScheduleParserHelper::storeOffTime::operator()(), AIRINV::ScheduleParserHelper::storeElapsedTime::operator()(), AIRINV::ScheduleParserHelper::storeCapacity::operator()(), and AIRINV::ScheduleParserHelper::doEndFlight::operator()().

22.51.4.9 LegCabinStruct AIRINV::FlightPeriodStruct::_itLegCabin

Definition at line 91 of file FlightPeriodStruct.hpp.

Referenced by AIRINV::ScheduleParserHelper::storeLegCabinCode::operator()(), and AIRINV::ScheduleParserHelper::storeCapacity::operator()().

22.51.4.10 stdair::Date_T AIRINV::FlightPeriodStruct::_dateRangeStart

Staging Date.

Definition at line 94 of file FlightPeriodStruct.hpp.

Referenced by AIRINV::ScheduleParserHelper::storeDateRangeStart::operator()(), and AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator()().

22.51.4.11 stdair::Date_T AIRINV::FlightPeriodStruct::_dateRangeEnd

Definition at line 95 of file FlightPeriodStruct.hpp.

Referenced by AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator()().

22.51.4.12 unsigned int AIRINV::FlightPeriodStruct::_itYear

Definition at line 96 of file FlightPeriodStruct.hpp.

Referenced by getDate().

22.51.4.13 unsigned int AIRINV::FlightPeriodStruct::_itMonth

Definition at line 97 of file FlightPeriodStruct.hpp.

Referenced by getDate().

22.51.4.14 unsigned int AIRINV::FlightPeriodStruct::_itDay

Definition at line 98 of file FlightPeriodStruct.hpp.

Referenced by getDate().

22.51.4.15 int AIRINV::FlightPeriodStruct::_dateOffset

Definition at line 99 of file FlightPeriodStruct.hpp.

Referenced by AIRINV::ScheduleParserHelper::storeBoardingTime::operator()(), AIRINV::ScheduleParserHelper::storeOffTime::operator()(), and AIRINV::ScheduleParserHelper::storeElapsedTime::operator()().

22.51.4.16 long AIRINV::FlightPeriodStruct::_itHours

Staging Time.

Definition at line 102 of file FlightPeriodStruct.hpp.

Referenced by getTime().

22.51.4.17 long AIRINV::FlightPeriodStruct::_itMinutes

Definition at line 103 of file FlightPeriodStruct.hpp.

Referenced by getTime().

22.51.4.18 long AIRINV::FlightPeriodStruct::_itSeconds

Definition at line 104 of file FlightPeriodStruct.hpp.

Referenced by getTime(), AIRINV::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRINV::ScheduleParserHelper::storeBoardingTime::operator()(), AIRINV::ScheduleParserHelper::storeOffTime::operator()(), and AIRINV::ScheduleParserHelper::storeElapsedTime::operator()().

22.51.4.19 AirportList_T AIRINV::FlightPeriodStruct::_airportList

Staging Airport List (helper to derive the list of Segment structures).

Definition at line 108 of file FlightPeriodStruct.hpp.

Referenced by addAirport(), buildSegments(), and AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator()().

22.51.4.20 AirportOrderedList_T AIRINV::FlightPeriodStruct::_airportOrderedList

Definition at line 109 of file FlightPeriodStruct.hpp.

Referenced by addAirport(), buildSegments(), and AIRINV::ScheduleParserHelper::storeSegmentSpecificity-::operator()().

22.51.4.21 bool AIRINV::FlightPeriodStruct::_areSegmentDefinitionsSpecific

Staging Segment-related attributes.

Definition at line 112 of file FlightPeriodStruct.hpp.

Referenced by AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRINV::ScheduleParser-Helper::storeClasses::operator()(), and AIRINV::ScheduleParserHelper::storeFClasses::operator()().

22.51.4.22 SegmentStruct AIRINV::FlightPeriodStruct::_itSegment

Definition at line 113 of file FlightPeriodStruct.hpp.

Referenced by AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::Schedule-ParserHelper::storeSegmentOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeClasses::operator()(), and AIRINV::ScheduleParserHelper::storeFClasses::operator()().

22.51.4.23 SegmentCabinStruct AIRINV::FlightPeriodStruct::_itSegmentCabin

Definition at line 114 of file FlightPeriodStruct.hpp.

Referenced by AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRINV::ScheduleParser-Helper::storeClasses::operator()(), AIRINV::ScheduleParserHelper::storeFamilyCode::operator()(), AIRINV::-ScheduleParserHelper::storeFRAT5CurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFFDisutility-CurveKey::operator()(), and AIRINV::ScheduleParserHelper::storeFClasses::operator()().

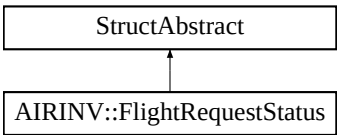
The documentation for this struct was generated from the following files:

- airinv/bom/FlightPeriodStruct.hpp
- airinv/bom/FlightPeriodStruct.cpp

22.52 AIRINV::FlightRequestStatus Struct Reference

`#include <airinv/FlightRequestStatus.hpp>`

Inheritance diagram for AIRINV::FlightRequestStatus:



Public Types

- enum EN_FlightRequestStatus { OK = 0, NOT_FOUND, INTERNAL_ERROR, LAST_VALUE }

Public Member Functions

- EN_FlightRequestStatus getCode () const
- const std::string describe () const
- FlightRequestStatus (const EN_FlightRequestStatus &)
- FlightRequestStatus (const std::string &iCode)

Static Public Member Functions

- static const std::string & getLabel (const EN_FlightRequestStatus &)
- static const std::string & getCodeLabel (const EN_FlightRequestStatus &)
- static std::string describeLabels ()

22.52.1 Detailed Description

Enumeration of flight type codes.

Definition at line 15 of file FlightRequestStatus.hpp.

22.52.2 Member Enumeration Documentation

22.52.2.1 enum AIRINV::FlightRequestStatus::EN_FlightRequestStatus

Enumerator:

OK

NOT_FOUND

INTERNAL_ERROR

LAST_VALUE

Definition at line 17 of file FlightRequestStatus.hpp.

22.52.3 Constructor & Destructor Documentation

22.52.3.1 AIRINV::FlightRequestStatus::FlightRequestStatus (const EN_FlightRequestStatus & *iFlightRequestStatus*)

Constructor.

Definition at line 25 of file FlightRequestStatus.cpp.

22.52.3.2 AIRINV::FlightRequestStatus::FlightRequestStatus (const std::string & *iCode*)

Constructor.

Definition at line 30 of file FlightRequestStatus.cpp.

References describeLabels(), INTERNAL_ERROR, LAST_VALUE, NOT_FOUND, and OK.

22.52.4 Member Function Documentation

22.52.4.1 const std::string & AIRINV::FlightRequestStatus::getLabel (const EN_FlightRequestStatus & *iCode*) [static]

Get the label as a string.

Definition at line 58 of file FlightRequestStatus.cpp.

22.52.4.2 const std::string & AIRINV::FlightRequestStatus::getCodeLabel (const EN_FlightRequestStatus & *iCode*) [static]

Get the label as a single char.

Definition at line 64 of file FlightRequestStatus.cpp.

22.52.4.3 std::string AIRINV::FlightRequestStatus::describeLabels ( ) [static]

List the labels.

Definition at line 69 of file FlightRequestStatus.cpp.

References LAST_VALUE.

Referenced by FlightRequestStatus().

22.52.4.4 FlightRequestStatus::EN_FlightRequestStatus AIRINV::FlightRequestStatus::getCode ( ) const

Get the enumerated value.

Definition at line 82 of file FlightRequestStatus.cpp.

22.52.4.5 const std::string AIRINV::FlightRequestStatus::describe ( ) const

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

Definition at line 87 of file FlightRequestStatus.cpp.

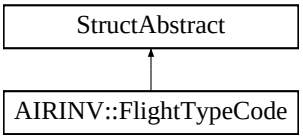
The documentation for this struct was generated from the following files:

- airinv/FlightRequestStatus.hpp
- airinv/basic/FlightRequestStatus.cpp

22.53 AIRINV::FlightTypeCode Struct Reference

`#include <airinv/basic/FlightTypeCode.hpp>`

Inheritance diagram for AIRINV::FlightTypeCode:



Public Types

- enum EN_FlightTypeCode { DOMESTIC = 0, INTERNATIONAL, GROUND_HANDLING, LAST_VALUE }

Public Member Functions

- EN_FlightTypeCode getCode () const
- const std::string describe () const
- FlightTypeCode (const EN_FlightTypeCode &)
- FlightTypeCode (const std::string &iCode)

Static Public Member Functions

- static const std::string & getLabel (const EN_FlightTypeCode &)
- static const std::string & getCodeLabel (const EN_FlightTypeCode &)
- static std::string describeLabels ()

22.53.1 Detailed Description

Enumeration of flight type codes.

Definition at line 15 of file FlightTypeCode.hpp.

22.53.2 Member Enumeration Documentation

22.53.2.1 enum AIRINV::FlightTypeCode::EN_FlightTypeCode

Enumerator:

DOMESTIC

INTERNATIONAL

GROUND_HANDLING

LAST_VALUE

Definition at line 17 of file FlightTypeCode.hpp.

22.53.3 Constructor & Destructor Documentation

22.53.3.1 AIRINV::FlightTypeCode::FlightTypeCode (const EN_FlightTypeCode & *iFlightTypeCode*)

Constructor.

Definition at line 24 of file FlightTypeCode.cpp.

22.53.3.2 AIRINV::FlightTypeCode::FlightTypeCode (const std::string & *iCode*)

Constructor.

Definition at line 29 of file FlightTypeCode.cpp.

References describeLabels(), DOMESTIC, GROUND_HANDLING, INTERNATIONAL, and LAST_VALUE.

22.53.4 Member Function Documentation

22.53.4.1 const std::string & AIRINV::FlightTypeCode::getLabel (const EN_FlightTypeCode & *iCode*) [static]

Get the label as a string.

Definition at line 54 of file FlightTypeCode.cpp.

22.53.4.2 const std::string & AIRINV::FlightTypeCode::getCodeLabel (const EN_FlightTypeCode & *iCode*) [static]

Get the label as a single char.

Definition at line 60 of file FlightTypeCode.cpp.

22.53.4.3 std::string AIRINV::FlightTypeCode::describeLabels ( ) [static]

List the labels.

Definition at line 65 of file FlightTypeCode.cpp.

References LAST_VALUE.

Referenced by FlightTypeCode().

22.53.4.4 FlightTypeCode::EN_FlightTypeCode AIRINV::FlightTypeCode::getCode ( ) const

Get the enumerated value.

Definition at line 77 of file FlightTypeCode.cpp.

Referenced by AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()().

22.53.4.5 const std::string AIRINV::FlightTypeCode::describe ( ) const

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

Definition at line 82 of file FlightTypeCode.cpp.

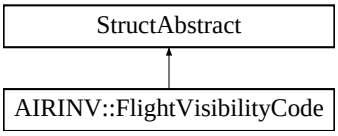
The documentation for this struct was generated from the following files:

- airinv/basic/FlightTypeCode.hpp
- airinv/basic/FlightTypeCode.cpp

22.54 AIRINV::FlightVisibilityCode Struct Reference

`#include <airinv/basic/FlightVisibilityCode.hpp>`

Inheritance diagram for AIRINV::FlightVisibilityCode:



Public Types

- enum EN_FlightVisibilityCode { NORMAL = 0, HIDDEN, PSEUDO, LAST_VALUE }

Public Member Functions

- EN_FlightVisibilityCode getCode () const
- const std::string describe () const
- FlightVisibilityCode (const EN_FlightVisibilityCode &)
- FlightVisibilityCode (const std::string &iCode)

Static Public Member Functions

- static const std::string & getLabel (const EN_FlightVisibilityCode &)
- static const std::string & getCodeLabel (const EN_FlightVisibilityCode &)
- static std::string describeLabels ()

22.54.1 Detailed Description

Enumeration of flight visibility codes.

Definition at line 15 of file FlightVisibilityCode.hpp.

22.54.2 Member Enumeration Documentation

22.54.2.1 enum AIRINV::FlightVisibilityCode::EN_FlightVisibilityCode

Enumerator:

NORMAL

HIDDEN

PSEUDO

LAST_VALUE

Definition at line 17 of file FlightVisibilityCode.hpp.

22.54.3 Constructor & Destructor Documentation

22.54.3.1 AIRINV::FlightVisibilityCode::FlightVisibilityCode (const EN_FlightVisibilityCode & *iFlightVisibilityCode*)

Constructor.

Definition at line 25 of file FlightVisibilityCode.cpp.

22.54.3.2 AIRINV::FlightVisibilityCode::FlightVisibilityCode (const std::string & *iCode*)

Constructor.

Definition at line 30 of file FlightVisibilityCode.cpp.

References describeLabels(), HIDDEN, LAST_VALUE, NORMAL, and PSEUDO.

22.54.4 Member Function Documentation

22.54.4.1 const std::string & AIRINV::FlightVisibilityCode::getLabel (const EN_FlightVisibilityCode & *iCode*) [static]

Get the label as a string.

Definition at line 57 of file FlightVisibilityCode.cpp.

22.54.4.2 const std::string & AIRINV::FlightVisibilityCode::getCodeLabel (const EN_FlightVisibilityCode & *iCode*) [static]

Get the label as a single char.

Definition at line 63 of file FlightVisibilityCode.cpp.

22.54.4.3 std::string AIRINV::FlightVisibilityCode::describeLabels ( ) [static]

List the labels.

Definition at line 68 of file FlightVisibilityCode.cpp.

References LAST_VALUE.

Referenced by FlightVisibilityCode().

22.54.4.4 FlightVisibilityCode::EN_FlightVisibilityCode AIRINV::FlightVisibilityCode::getCode ( ) const

Get the enumerated value.

Definition at line 81 of file FlightVisibilityCode.cpp.

Referenced by AIRINV::FlightDateStruct::describe(), and AIRINV::InventoryParserHelper::storeFlightVisibilityCode-::operator()().

22.54.4.5 const std::string AIRINV::FlightVisibilityCode::describe ( ) const

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

Definition at line 86 of file FlightVisibilityCode.cpp.

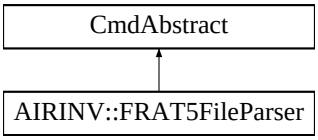
The documentation for this struct was generated from the following files:

- airinv/basic/FlightVisibilityCode.hpp
- airinv/basic/FlightVisibilityCode.cpp

22.55 AIRINV::FRAT5FileParser Class Reference

`#include <airinv/command/FRAT5ParserHelper.hpp>`

Inheritance diagram for AIRINV::FRAT5FileParser:



Public Member Functions

- FRAT5FileParser (stdair::BomRoot &ioBomRoot, const stdair::Filename_T &iFilename)
- bool generateFRAT5Curves ()

22.55.1 Detailed Description

Short Description

Detailed Description. Class wrapping the initialisation and entry point of the parser.

The seemingly redundancy is used to force the instantiation of the actual parser, which is a templatised Boost Spirit grammar. Hence, the actual parser is instantiated within that class object code.

Definition at line 126 of file FRAT5ParserHelper.hpp.

22.55.2 Constructor & Destructor Documentation

22.55.2.1 AIRINV::FRAT5FileParser::FRAT5FileParser (stdair::BomRoot & *ioBomRoot,* const stdair::Filename_T & *iFilename*)

Constructor.

Definition at line 178 of file FRAT5ParserHelper.cpp.

22.55.3 Member Function Documentation

22.55.3.1 bool AIRINV::FRAT5FileParser::generateFRAT5Curves ( )

Parse the input file and generate the FRAT5 curves.

Definition at line 202 of file FRAT5ParserHelper.cpp.

Referenced by AIRINV::FRAT5Parser::parse().

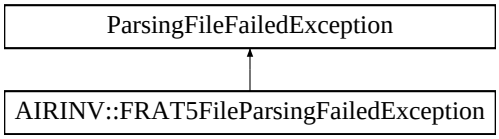
The documentation for this class was generated from the following files:

- airinv/command/FRAT5ParserHelper.hpp
- airinv/command/FRAT5ParserHelper.cpp

22.56 AIRINV::FRAT5FileParsingFailedException Class Reference

`#include <airinv/AIRINV_Types.hpp>`

Inheritance diagram for AIRINV::FRAT5FileParsingFailedException:



Public Member Functions

- FRAT5FileParsingFailedException (const std::string &iWhat)

22.56.1 Detailed Description

The FRAT5 input file can not be parsed.

Definition at line 67 of file AIRINV_Types.hpp.

22.56.2 Constructor & Destructor Documentation

22.56.2.1 AIRINV::FRAT5FileParsingFailedException::FRAT5FileParsingFailedException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 73 of file AIRINV_Types.hpp.

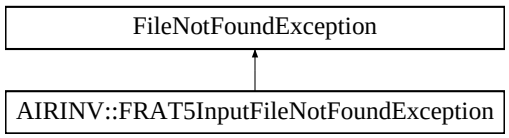
The documentation for this class was generated from the following file:

- airinv/AIRINV_Types.hpp

22.57 AIRINV::FRAT5InputFileNotFoundException Class Reference

`#include <airinv/AIRINV_Types.hpp>`

Inheritance diagram for AIRINV::FRAT5InputFileNotFoundException:



Public Member Functions

- FRAT5InputFileNotFoundException (const std::string &iWhat)

22.57.1 Detailed Description

The FRAT5 input file can not be found or opened.

Definition at line 130 of file AIRINV_Types.hpp.

22.57.2 Constructor & Destructor Documentation

22.57.2.1 AIRINV::FRAT5InputFileNotFoundException::FRAT5InputFileNotFoundException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 135 of file AIRINV_Types.hpp.

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

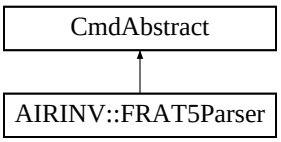
- airinv/AIRINV_Types.hpp

22.58 AIRINV::FRAT5Parser Class Reference

Class wrapping the parser entry point.

`#include <airinv/command/FRAT5Parser.hpp>`

Inheritance diagram for AIRINV::FRAT5Parser:



Static Public Member Functions

- static void parse (const stdair::FRAT5FilePath &iFRAT5InputFilename, stdair::BomRoot &)

22.58.1 Detailed Description

Class wrapping the parser entry point.

Definition at line 22 of file FRAT5Parser.hpp.

22.58.2 Member Function Documentation

22.58.2.1 void AIRINV::FRAT5Parser::parse (const stdair::FRAT5FilePath & *iFRAT5InputFilename,* stdair::BomRoot & *ioBomRoot*) [static]

Parse the CSV file describing the FRAT5 curves for the simulator, and generates the curves accordingly.

Parameters

const	stdair::Filename_T& The file-name of the CSV-formatted FRAT5 curve input file.
stdair::BomRoot&	Root of the BOM tree.

Definition at line 20 of file FRAT5Parser.cpp.

References AIRINV::FRAT5FileParser::generateFRAT5Curves().

Referenced by AIRINV::AIRINV_Service::parseAndLoad().

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

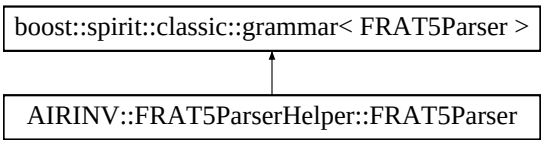
- airinv/command/FRAT5Parser.hpp

- airinv/command/FRAT5Parser.cpp

22.59 AIRINV::FRAT5ParserHelper::FRAT5Parser Struct Reference

`#include <airinv/command/FRAT5ParserHelper.hpp>`

Inheritance diagram for AIRINV::FRAT5ParserHelper::FRAT5Parser:



Classes

- struct definition

Public Member Functions

- FRAT5Parser (stdair::BomRoot &, FRAT5Struct &)

Public Attributes

- stdair::BomRoot & _bomRoot
- FRAT5Struct & _frat5

22.59.1 Detailed Description

CurveKey; FRAT5Curve; F1; 63:1.40; 56:1.45; 49:1.50; 42:1.55; 35:1.60; 31:1.70; 27:1.80; 23:2.00; 19:2.30; 16:2.-60; 13:3.00; 10:3.30; 7:3.40; 5:3.44; 3:3.47; 1:3.50;

Grammar: Curve ::= Key ';' FRAT5Map EndOfCurve Key ::= string FRAT5Map ::= DTDValuePaire (';' DTDValue-Paire)* DTDValuePaire ::= DTD ':' FRAT5Value EndOfCurve ::= ';' Grammar for the FRAT5 curve parser.

Definition at line 89 of file FRAT5ParserHelper.hpp.

22.59.2 Constructor & Destructor Documentation

22.59.2.1 AIRINV::FRAT5ParserHelper::FRAT5Parser::FRAT5Parser (stdair::BomRoot & *ioBomRoot,* FRAT5Struct & *ioFRAT5*)

Definition at line 115 of file FRAT5ParserHelper.cpp.

22.59.3 Member Data Documentation

22.59.3.1 stdair::BomRoot& AIRINV::FRAT5ParserHelper::FRAT5Parser::_bomRoot

Definition at line 107 of file FRAT5ParserHelper.hpp.

22.59.3.2 FRAT5Struct& AIRINV::FRAT5ParserHelper::FRAT5Parser::_frat5

Definition at line 108 of file FRAT5ParserHelper.hpp.

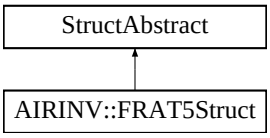
The documentation for this struct was generated from the following files:

- airinv/command/FRAT5ParserHelper.hpp
- airinv/command/FRAT5ParserHelper.cpp

22.60 AIRINV::FRAT5Struct Struct Reference

`#include <airinv/bom/FRAT5Struct.hpp>`

Inheritance diagram for AIRINV::FRAT5Struct:



Public Member Functions

- const std::string describe () const
- FRAT5Struct ()
- ~FRAT5Struct ()

Public Attributes

- std::string _key
- stdair::FRAT5Curve_T _curve
- stdair::DTD_T _dtd

22.60.1 Detailed Description

Utility Structure for the parsing of FRAT5 structures.

Definition at line 15 of file FRAT5Struct.hpp.

22.60.2 Constructor & Destructor Documentation

22.60.2.1 AIRINV::FRAT5Struct::FRAT5Struct ( )

Default constructor.

Definition at line 15 of file FRAT5Struct.cpp.

22.60.2.2 AIRINV::FRAT5Struct::~FRAT5Struct ( )

Destructor

Definition at line 19 of file FRAT5Struct.cpp.

22.60.3 Member Function Documentation

22.60.3.1 const std::string AIRINV::FRAT5Struct::describe ( ) const

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

Definition at line 23 of file FRAT5Struct.cpp.

References _curve, and _key.

Referenced by AIRINV::FRAT5ParserHelper::doEndCurve::operator()().

22.60.4 Member Data Documentation

22.60.4.1 std::string AIRINV::FRAT5Struct::_key

Curve key.

Definition at line 37 of file FRAT5Struct.hpp.

Referenced by describe(), AIRINV::FRAT5ParserHelper::storeCurveKey::operator()(), and AIRINV::FRAT5ParserHelper::doEndCurve::operator()().

22.60.4.2 stdair::FRAT5Curve_T AIRINV::FRAT5Struct::_curve

Curve.

Definition at line 40 of file FRAT5Struct.hpp.

Referenced by describe(), AIRINV::FRAT5ParserHelper::storeFRAT5Value::operator()(), and AIRINV::FRAT5ParserHelper::doEndCurve::operator()().

22.60.4.3 stdair::DTD_T AIRINV::FRAT5Struct::_dtd

DTD.

Definition at line 45 of file FRAT5Struct.hpp.

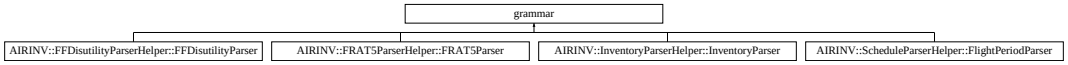
Referenced by AIRINV::FRAT5ParserHelper::storeDTD::operator()(), and AIRINV::FRAT5ParserHelper::storeFRAT5Value::operator()().

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

- airinv/bom/FRAT5Struct.hpp
- airinv/bom/FRAT5Struct.cpp

22.61 grammar Class Reference

Inheritance diagram for grammar:

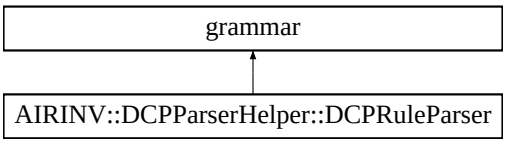


The documentation for this class was generated from the following file:

- airinv/command/FFDisutilityParserHelper.hpp

22.62 grammar Class Reference

Inheritance diagram for grammar:



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

- airinv/command/vault/DCPParserHelper.hpp

22.63 AIRINV::header Struct Reference

`#include <airinv/server/header.hpp>`

Public Attributes

- std::string name
- std::string value

22.63.1 Detailed Description

Header structure.

Definition at line 13 of file header.hpp.

22.63.2 Member Data Documentation

22.63.2.1 std::string AIRINV::header::name

Definition at line 14 of file header.hpp.

22.63.2.2 std::string AIRINV::header::value

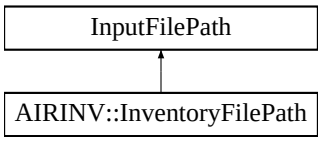
Definition at line 15 of file header.hpp.

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

- airinv/server/header.hpp

22.64 InputFilePath Class Reference

Inheritance diagram for InputFilePath:



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

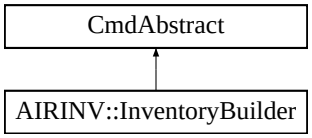
- airinv/AIRINV_Types.hpp

22.65 AIRINV::InventoryBuilder Class Reference

Class handling the generation / instantiation of the Inventory BOM.

`#include <airinv/command/InventoryBuilder.hpp>`

Inheritance diagram for AIRINV::InventoryBuilder:



Friends

- class AIRINV_Service
- struct InventoryParserHelper::doEndFlightDate

22.65.1 Detailed Description

Class handling the generation / instantiation of the Inventory BOM.

Definition at line 45 of file InventoryBuilder.hpp.

22.65.2 Friends And Related Function Documentation

22.65.2.1 friend class AIRINV_Service [friend]

Only the following class may use methods of InventoryBuilder. Indeed, as those methods build the BOM, it is not good to expose them publicly.

Definition at line 51 of file InventoryBuilder.hpp.

22.65.2.2 friend struct InventoryParserHelper::doEndFlightDate [friend]

Definition at line 52 of file InventoryBuilder.hpp.

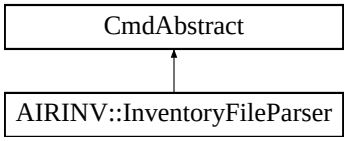
The documentation for this class was generated from the following files:

- airinv/command/InventoryBuilder.hpp
- airinv/command/InventoryBuilder.cpp

22.66 AIRINV::InventoryFileParser Class Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryFileParser:



Public Member Functions

- InventoryFileParser (stdair::BomRoot &, const stdair::Filename_T &iInventoryInputFilename)
- bool buildInventory ()

22.66.1 Detailed Description

Class wrapping the initialisation and entry point of the parser.

The seemingly redundancy is used to force the instantiation of the actual parser, which is a templatised Boost Spirit grammar. Hence, the actual parser is instantiated within that class object code.

Definition at line 516 of file InventoryParserHelper.hpp.

22.66.2 Constructor & Destructor Documentation

22.66.2.1 AIRINV::InventoryFileParser::InventoryFileParser (stdair::BomRoot & , const stdair::Filename_T & *iInventoryInputFilename*)

Constructor.

Definition at line 1132 of file InventoryParserHelper.cpp.

22.66.3 Member Function Documentation

22.66.3.1 bool AIRINV::InventoryFileParser::buildInventory ( )

Parse the inventory input file.

Definition at line 1156 of file InventoryParserHelper.cpp.

Referenced by AIRINV::InventoryParser::buildInventory().

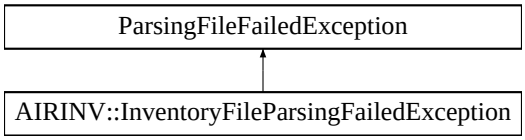
The documentation for this class was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.67 AIRINV::InventoryFileParsingFailedException Class Reference

```
#include <airinv/AIRINV_Types.hpp>
```

Inheritance diagram for AIRINV::InventoryFileParsingFailedException:



Public Member Functions

- InventoryFileParsingFailedException (const std::string &iWhat)

22.67.1 Detailed Description

The inventory input file can not be parsed.

Definition at line 28 of file AIRINV_Types.hpp.

22.67.2 Constructor & Destructor Documentation

22.67.2.1 AIRINV::InventoryFileParsingFailedException::InventoryFileParsingFailedException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 34 of file AIRINV_Types.hpp.

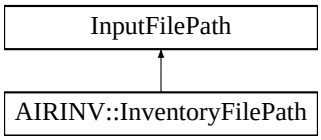
The documentation for this class was generated from the following file:

- airinv/AIRINV_Types.hpp

22.68 AIRINV::InventoryFilePath Class Reference

`#include <airinv/AIRINV_Types.hpp>`

Inheritance diagram for AIRINV::InventoryFilePath:



Public Member Functions

- InventoryFilePath (const stdair::Filename_T &iFilename)

22.68.1 Detailed Description

Inventory input file.

Definition at line 198 of file AIRINV_Types.hpp.

22.68.2 Constructor & Destructor Documentation

22.68.2.1 AIRINV::InventoryFilePath::InventoryFilePath (const stdair::Filename_T & *iFilename*) [inline], [explicit]

Constructor.

Definition at line 203 of file AIRINV_Types.hpp.

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

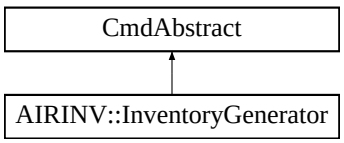
- airinv/AIRINV_Types.hpp

22.69 AIRINV::InventoryGenerator Class Reference

Class handling the generation / instantiation of the Inventory BOM.

`#include <airinv/command/InventoryGenerator.hpp>`

Inheritance diagram for AIRINV::InventoryGenerator:



Friends

- class FlightPeriodFileParser
- class FFFlightPeriodFileParser
- struct ScheduleParserHelper::doEndFlight
- class ScheduleParser

22.69.1 Detailed Description

Class handling the generation / instantiation of the Inventory BOM.

Definition at line 42 of file InventoryGenerator.hpp.

22.69.2 Friends And Related Function Documentation

22.69.2.1 friend class FlightPeriodFileParser [friend]

Only the following class may use methods of InventoryGenerator. Indeed, as those methods build the BOM, it is not good to expose them publicly.

Definition at line 48 of file InventoryGenerator.hpp.

22.69.2.2 friend class FFFlightPeriodFileParser [friend]

Definition at line 49 of file InventoryGenerator.hpp.

22.69.2.3 friend struct ScheduleParserHelper::doEndFlight [friend]

Definition at line 50 of file InventoryGenerator.hpp.

22.69.2.4 friend class ScheduleParser [friend]

Definition at line 51 of file InventoryGenerator.hpp.

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

- airinv/command/InventoryGenerator.hpp
- airinv/command/InventoryGenerator.cpp

22.70 AIRINV::InventoryHelper Class Reference

`#include <airinv/bom/InventoryHelper.hpp>`

Static Public Member Functions

- static void fillFromRouting (const stdair::Inventory &)
- static void calculateAvailability (const stdair::Inventory &, const std::string &, stdair::TravelSolutionStruct &)
- static void getYieldAndBidPrice (const stdair::Inventory &, const std::string &, stdair::TravelSolutionStruct &)
- static bool sell (stdair::Inventory &, const std::string &iSegmentDateKey, const stdair::ClassCode_T &, const stdair::PartySize_T &)
- static bool sell (const stdair::BookingClassID_T &, const stdair::PartySize_T &)
- static bool cancel (stdair::Inventory &, const std::string &iSegmentDateKey, const stdair::ClassCode_T &, const stdair::PartySize_T &)
- static bool cancel (const stdair::BookingClassID_T &, const stdair::PartySize_T &)
- static void takeSnapshots (const stdair::Inventory &, const stdair::DateTime_T &)

22.70.1 Detailed Description

Class representing the actual business functions for an airline inventory.

Definition at line 23 of file InventoryHelper.hpp.

22.70.2 Member Function Documentation

22.70.2.1 void AIRINV::InventoryHelper::fillFromRouting (const stdair::Inventory & *ilInventory*) [static]

Fill the attributes derived from the routing legs (e.g., board and off dates).

Definition at line 28 of file InventoryHelper.cpp.

22.70.2.2 void AIRINV::InventoryHelper::calculateAvailability (const stdair::Inventory & *ilInventory,* const std::string & *iFullSegmentDateKey,* stdair::TravelSolutionStruct & *ioTravelSolution*) [static]

Compute the availability for the given travel solution.

Definition at line 44 of file InventoryHelper.cpp.

22.70.2.3 void AIRINV::InventoryHelper::getYieldAndBidPrice (const stdair::Inventory & *ilInventory,* const std::string & *iFullSegmentDateKey,* stdair::TravelSolutionStruct & *ioTravelSolution*) [static]

Get yield and bid price information for the given travel solution.

Definition at line 107 of file InventoryHelper.cpp.

22.70.2.4 bool AIRINV::InventoryHelper::sell (stdair::Inventory & *ioInventory,* const std::string & *iSegmentDateKey,* const stdair::ClassCode_T & *iClassCode,* const stdair::PartySize_T & *iPartySize*) [static]

Make a sale.

Definition at line 248 of file InventoryHelper.cpp.

Referenced by sell().

22.70.2.5 bool AIRINV::InventoryHelper::sell (const stdair::BookingClassID_T & *iClassID,* const stdair::PartySize_T & *iPartySize*) [static]

Make a sale.

Definition at line 282 of file InventoryHelper.cpp.

References sell().

22.70.2.6 bool AIRINV::InventoryHelper::cancel (stdair::Inventory & *ioInventory,* const std::string & *iSegmentDateKey,* const stdair::ClassCode_T & *iClassCode,* const stdair::PartySize_T & *iPartySize*) [static]

Make a cancellation.

Definition at line 328 of file InventoryHelper.cpp.

Referenced by cancel().

22.70.2.7 bool AIRINV::InventoryHelper::cancel (const stdair::BookingClassID_T & *iClassID,* const stdair::PartySize_T & *iPartySize*) [static]

Make a cancellation.

Definition at line 362 of file InventoryHelper.cpp.

References cancel().

22.70.2.8 void AIRINV::InventoryHelper::takeSnapshots (const stdair::Inventory & *iInventory,* const stdair::DateTime_T & *iSnapshotTime*) [static]

Take inventory snapshots.

Definition at line 407 of file InventoryHelper.cpp.

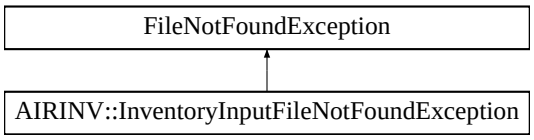
The documentation for this class was generated from the following files:

- airinv/bom/InventoryHelper.hpp
- airinv/bom/InventoryHelper.cpp

22.71 AIRINV::InventoryInputFileNotFoundException Class Reference

`#include <airinv/AIRINV_Types.hpp>`

Inheritance diagram for AIRINV::InventoryInputFileNotFoundException:



Public Member Functions

- InventoryInputFileNotFoundException (const std::string &iWhat)

22.71.1 Detailed Description

The inventory input file can not be found or opened.

Definition at line 106 of file AIRINV_Types.hpp.

22.71.2 Constructor & Destructor Documentation

22.71.2.1 AIRINV::InventoryInputFileNotFoundException::InventoryInputFileNotFoundException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 111 of file AIRINV_Types.hpp.

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

- airinv/AIRINV_Types.hpp

22.72 AIRINV::InventoryManager Class Reference

`#include <airinv/command/InventoryManager.hpp>`

Static Public Member Functions

- static void createDirectAccesses (const stdair::BomRoot &)
- static void createDirectAccesses (const stdair::BomRoot &, stdair::Inventory &)
- static void createDirectAccesses (const stdair::BomRoot &, stdair::Inventory &, stdair::FlightDate &)
- static void createDirectAccesses (stdair::SegmentDate &)
- static void createPartnerAccesses (const stdair::BomRoot &, stdair::Inventory &)
- static void createPartnerAccesses (stdair::FlightDate &)
- static void createPartnerAccesses (const stdair::BomRoot &, stdair::Inventory &, stdair::FlightDate &)
- static void buildSimilarSegmentCabinSets (const stdair::BomRoot &)
- static void buildSimilarSegmentCabinSets (stdair::Inventory &)
- static void buildSegmentSnapshotTable (stdair::Inventory &, const stdair::TableID_T &, const DepartureDate-SegmentCabinMap_T &)
- static void setDefaultBidPriceVector (stdair::BomRoot &)
- static void setDefaultBidPriceVector (stdair::Inventory &)
- static void initialiseYieldBasedNestingStructures (const stdair::BomRoot &)
- static void initialiseListsOfUsablePolicies (const stdair::BomRoot &)

Friends

- class AIRINV_Master_Service
- class AIRINV_Service

22.72.1 Detailed Description

Command wrapping the travel request process.

Definition at line 36 of file InventoryManager.hpp.

22.72.2 Member Function Documentation

22.72.2.1 void AIRINV::InventoryManager::createDirectAccesses (const stdair::BomRoot & *iBomRoot*) [static]

Create the direct accesses within the inventories suck as links between leg-date and segment-date, ect.

Definition at line 746 of file InventoryManager.cpp.

References createPartnerAccesses(), and AIRINV::BomRootHelper::fillFromRouting().

Referenced by AIRINV::AIRINV_Service::buildComplementaryLinks(), and createDirectAccesses().

22.72.2.2 void AIRINV::InventoryManager::createDirectAccesses (const stdair::BomRoot & *iBomRoot,* stdair::Inventory & *ioInventory*) [static]

Definition at line 776 of file InventoryManager.cpp.

References createDirectAccesses().

22.72.2.3 void AIRINV::InventoryManager::createDirectAccesses (const stdair::BomRoot & *ioBomRoot,* stdair::Inventory & *ioInventory,* stdair::FlightDate & *ioFlightDate*) [static]

Definition at line 811 of file InventoryManager.cpp.

References createDirectAccesses().

22.72.2.4 void AIRINV::InventoryManager::createDirectAccesses (stdair::SegmentDate & *ioSegmentDate*) [static]

Definition at line 860 of file InventoryManager.cpp.

22.72.2.5 void AIRINV::InventoryManager::createPartnerAccesses (const stdair::BomRoot & *iBomRoot,* stdair::Inventory & *ioInventory*) [static]

Create the direct accesses within the inventories suck as the link between a marketing segment date and its operating one.

Definition at line 926 of file InventoryManager.cpp.

Referenced by createDirectAccesses().

22.72.2.6 static void AIRINV::InventoryManager::createPartnerAccesses (stdair::FlightDate &) [static]

22.72.2.7 void AIRINV::InventoryManager::createPartnerAccesses (const stdair::BomRoot & *ioBomRoot,* stdair::Inventory & *ioInventory,* stdair::FlightDate & *ioFlightDate*) [static]

Definition at line 945 of file InventoryManager.cpp.

22.72.2.8 void AIRINV::InventoryManager::buildSimilarSegmentCabinSets (const stdair::BomRoot & *iBomRoot*) [static]

Build the similar segment-cabin sets and the corresponding snapshot tables and other data.

Definition at line 1027 of file InventoryManager.cpp.

Referenced by AIRINV::AIRINV_Service::buildComplementaryLinks().

22.72.2.9 void AIRINV::InventoryManager::buildSimilarSegmentCabinSets (stdair::Inventory & *ioInventory*) [static]

Definition at line 1043 of file InventoryManager.cpp.

References buildSegmentSnapshotTable().

22.72.2.10 void AIRINV::InventoryManager::buildSegmentSnapshotTable (stdair::Inventory & *ioInventory,* const stdair::TableID_T & *iTableID,* const DepartureDateSegmentCabinMap_T & *iDDSCMap*) [static]

Definition at line 1118 of file InventoryManager.cpp.

Referenced by buildSimilarSegmentCabinSets().

22.72.2.11 void AIRINV::InventoryManager::setDefaultBidPriceVector (stdair::BomRoot & *ioBomRoot*) [static]

Bid price vectors initialisation

Definition at line 601 of file InventoryManager.cpp.

Referenced by AIRINV::AIRINV_Service::buildComplementaryLinks().

22.72.2.12 void AIRINV::InventoryManager::setDefaultBidPriceVector (stdair::Inventory & *ioInventory*) [static]

Definition at line 633 of file InventoryManager.cpp.

22.72.2.13 void AIRINV::InventoryManager::initialiseYieldBasedNestingStructures (const stdair::BomRoot & *iBomRoot*) [static]

Yield-based nesting structure initialisation

Definition at line 1277 of file InventoryManager.cpp.

References AIRINV::SegmentCabinHelper::initYieldBasedNestingStructure().

Referenced by AIRINV::AIRINV_Service::buildComplementaryLinks().

22.72.2.14 void AIRINV::InventoryManager::initialiseListsOfUsablePolicies (const stdair::BomRoot & *iBomRoot*) [static]

Lists of usable policies initialisation.

Definition at line 1327 of file InventoryManager.cpp.

References AIRINV::SegmentCabinHelper::initListOfUsablePolicies().

Referenced by AIRINV::AIRINV_Service::buildComplementaryLinks().

22.72.3 Friends And Related Function Documentation

22.72.3.1 friend class AIRINV_Master_Service [friend]

Definition at line 37 of file InventoryManager.hpp.

22.72.3.2 friend class AIRINV_Service [friend]

Definition at line 38 of file InventoryManager.hpp.

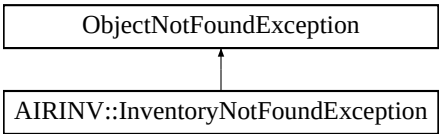
The documentation for this class was generated from the following files:

- airinv/command/InventoryManager.hpp
- airinv/command/InventoryManager.cpp

22.73 AIRINV::InventoryNotFoundException Class Reference

`#include <airinv/AIRINV_Types.hpp>`

Inheritance diagram for AIRINV::InventoryNotFoundException:



Public Member Functions

- InventoryNotFoundException (const std::string &iWhat)

22.73.1 Detailed Description

Inventory not found.

Definition at line 172 of file AIRINV_Types.hpp.

22.73.2 Constructor & Destructor Documentation

22.73.2.1 AIRINV::InventoryNotFoundException::InventoryNotFoundException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 177 of file AIRINV_Types.hpp.

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

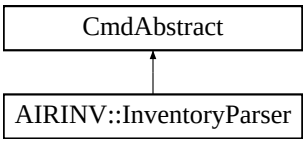
- airinv/AIRINV_Types.hpp

22.74 AIRINV::InventoryParser Class Reference

Class wrapping the parser entry point.

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

Inheritance diagram for AIRINV::InventoryParser:



Static Public Member Functions

- static void buildInventory (const InventoryFilePath &, stdair::BomRoot &)

22.74.1 Detailed Description

Class wrapping the parser entry point.

Definition at line 23 of file InventoryParser.hpp.

22.74.2 Member Function Documentation

22.74.2.1 void AIRINV::InventoryParser::buildInventory (const InventoryFilePath & *iInventoryFilename,* stdair::BomRoot & *ioBomRoot*) [static]

Parses the CSV file describing an airline inventory, and generates the corresponding data model in memory. It can then be used, for instance, in a simulator.

Parameters

const	InventoryFilePath& The file-name of the CSV-formatted inventory input file.
stdair::BomRoot&	Root of the BOM tree.

Definition at line 20 of file InventoryParser.cpp.

References AIRINV::InventoryFileParser::buildInventory().

Referenced by AIRINV::AIRINV_Service::parseAndLoad().

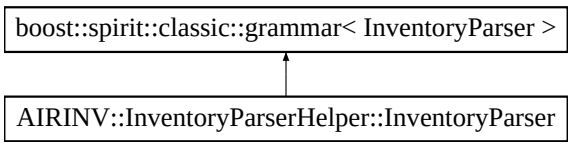
The documentation for this class was generated from the following files:

- airinv/command/InventoryParser.hpp
- airinv/command/InventoryParser.cpp

22.75 AIRINV::InventoryParserHelper::InventoryParser Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::InventoryParser:



Classes

- struct definition

Public Member Functions

- InventoryParser (stdair::BomRoot &, FlightDateStruct &, unsigned int &)

Public Attributes

- stdair::BomRoot & _bomRoot
- FlightDateStruct & _flightDate
- unsigned int & _nbOfFlights

22.75.1 Detailed Description

FlightDepDate; 2010-02-08; SIN; BKK; L; 10.0; 1.0;

Grammar: FlightDate ::= FlightDepDate ';' Origin ';' Destination EndOfFlightDate FlightDepDate ::= date EndOfFlightDate ::= ';' Grammar for the inventory parser.

Definition at line 470 of file InventoryParserHelper.hpp.

22.75.2 Constructor & Destructor Documentation

22.75.2.1 AIRINV::InventoryParserHelper::InventoryParser::InventoryParser (stdair::BomRoot & *ioBomRoot,* FlightDateStruct & *ioFlightDate,* unsigned int & *ioNbOfFlights*)

Definition at line 894 of file InventoryParserHelper.cpp.

22.75.3 Member Data Documentation

22.75.3.1 stdair::BomRoot& AIRINV::InventoryParserHelper::InventoryParser::_bomRoot

Definition at line 498 of file InventoryParserHelper.hpp.

22.75.3.2 FlightDateStruct& AIRINV::InventoryParserHelper::InventoryParser::_flightDate

Definition at line 499 of file InventoryParserHelper.hpp.

22.75.3.3 unsigned int& AIRINV::InventoryParserHelper::InventoryParser::_nbOfFlights

Definition at line 500 of file InventoryParserHelper.hpp.

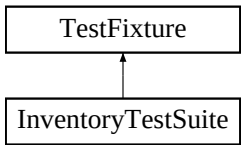
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.76 InventoryTestSuite Class Reference

`#include <test/airinv/InventoryTestSuite.hpp>`

Inheritance diagram for InventoryTestSuite:



Public Member Functions

- void simpleInventory ()
- InventoryTestSuite ()

Protected Attributes

- std::stringstream _describeKey

22.76.1 Detailed Description

Utility class for CPPUnit-based testing.

Definition at line 7 of file InventoryTestSuite.hpp.

22.76.2 Constructor & Destructor Documentation

22.76.2.1 InventoryTestSuite::InventoryTestSuite ( )

Test some error detection functionalities. Constructor.

22.76.3 Member Function Documentation

22.76.3.1 void InventoryTestSuite::simpleInventory ( )

Test a simple inventory functionality.

22.76.4 Member Data Documentation

22.76.4.1 std::stringstream InventoryTestSuite::_describeKey [protected]

Definition at line 28 of file InventoryTestSuite.hpp.

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

- test/airinv/InventoryTestSuite.hpp

22.77 AIRINV::LegCabinHelper Class Reference

`#include <airinv/bom/LegCabinHelper.hpp>`

22.77.1 Detailed Description

Class representing the actual business functions for an airline leg-cabin.

Definition at line 16 of file LegCabinHelper.hpp.

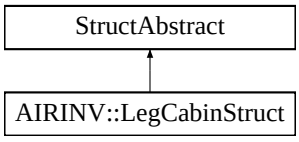
The documentation for this class was generated from the following file:

- airinv/bom/LegCabinHelper.hpp

22.78 AIRINV::LegCabinStruct Struct Reference

`#include <airinv/bom/LegCabinStruct.hpp>`

Inheritance diagram for AIRINV::LegCabinStruct:



Public Member Functions

- void fill (stdair::LegCabin &) const
- const std::string describe () const

Public Attributes

- stdair::CabinCode_T _cabinCode
- stdair::CabinCapacity_T _saleableCapacity
- stdair::CapacityAdjustment_T _adjustment
- stdair::CapacityAdjustment_T _dcsRegrade
- stdair::AuthorizationLevel_T _au
- stdair::Availability_T _avPool
- stdair::UPR_T _upr
- stdair::NbOfBookings_T _nbOfBookings
- stdair::Availability_T _nav
- stdair::Availability_T _gav
- stdair::OverbookingRate_T _acp
- stdair::NbOfBookings_T _etb
- stdair::NbOfBookings_T _staffNbOfBookings
- stdair::NbOfBookings_T _wlNbOfBookings
- stdair::NbOfBookings_T _groupNbOfBookings
- BucketStructList_T _bucketList

22.78.1 Detailed Description

Utility Structure for the parsing of LegCabin details.

Definition at line 24 of file LegCabinStruct.hpp.

22.78.2 Member Function Documentation

22.78.2.1 void AIRINV::LegCabinStruct::fill (stdair::LegCabin & *ioLegCabin*) const

Fill the LegCabin objects with the attributes of the LegCabinStruct.

Definition at line 38 of file LegCabinStruct.cpp.

References _saleableCapacity.

22.78.2.2 const std::string AIRINV::LegCabinStruct::describe ( ) const

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

Definition at line 15 of file LegCabinStruct.cpp.

References _acp, _adjustment, _au, _avPool, _bucketList, _cabinCode, _dcsRegrade, _etb, _gav, _groupNbOfBookings, _nav, _nbOfBookings, _saleableCapacity, _staffNbOfBookings, _upr, _wlNbOfBookings, and AIRINV::BucketStruct::describe().

Referenced by AIRINV::LegStruct::describe().

22.78.3 Member Data Documentation

22.78.3.1 stdair::CabinCode_T AIRINV::LegCabinStruct::_cabinCode

Definition at line 26 of file LegCabinStruct.hpp.

Referenced by describe(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), and AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()().

22.78.3.2 stdair::CabinCapacity_T AIRINV::LegCabinStruct::_saleableCapacity

Definition at line 27 of file LegCabinStruct.hpp.

Referenced by describe(), fill(), AIRINV::ScheduleParserHelper::storeCapacity::operator()(), and AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()().

22.78.3.3 stdair::CapacityAdjustment_T AIRINV::LegCabinStruct::_adjustment

Definition at line 28 of file LegCabinStruct.hpp.

Referenced by describe().

22.78.3.4 stdair::CapacityAdjustment_T AIRINV::LegCabinStruct::_dcsRegrade

Definition at line 29 of file LegCabinStruct.hpp.

Referenced by describe().

22.78.3.5 stdair::AuthorizationLevel_T AIRINV::LegCabinStruct::_au

Definition at line 30 of file LegCabinStruct.hpp.

Referenced by describe(), and AIRINV::InventoryParserHelper::storeAU::operator()().

22.78.3.6 stdair::Availability_T AIRINV::LegCabinStruct::_avPool

Definition at line 31 of file LegCabinStruct.hpp.

Referenced by describe().

22.78.3.7 stdair::UPR_T AIRINV::LegCabinStruct::_upr

Definition at line 32 of file LegCabinStruct.hpp.

Referenced by describe(), and AIRINV::InventoryParserHelper::storeUPR::operator()().

22.78.3.8 stdair::NbOfBookings_T AIRINV::LegCabinStruct::_nbOfBookings

Definition at line 33 of file LegCabinStruct.hpp.

Referenced by describe(), and AIRINV::InventoryParserHelper::storeBookingCounter::operator()().

22.78.3.9 stdair::Availability_T AIRINV::LegCabinStruct::_nav

Definition at line 34 of file LegCabinStruct.hpp.

Referenced by describe(), and AIRINV::InventoryParserHelper::storeNAV::operator()().

22.78.3.10 stdair::Availability_T AIRINV::LegCabinStruct::_gav

Definition at line 35 of file LegCabinStruct.hpp.

Referenced by describe(), and AIRINV::InventoryParserHelper::storeGAV::operator()().

22.78.3.11 stdair::OverbookingRate_T AIRINV::LegCabinStruct::_acp

Definition at line 36 of file LegCabinStruct.hpp.

Referenced by describe(), and AIRINV::InventoryParserHelper::storeACP::operator()().

22.78.3.12 stdair::NbOfBookings_T AIRINV::LegCabinStruct::_etb

Definition at line 37 of file LegCabinStruct.hpp.

Referenced by describe(), and AIRINV::InventoryParserHelper::storeETB::operator()().

22.78.3.13 stdair::NbOfBookings_T AIRINV::LegCabinStruct::_staffNbOfBookings

Definition at line 38 of file LegCabinStruct.hpp.

Referenced by describe().

22.78.3.14 stdair::NbOfBookings_T AIRINV::LegCabinStruct::_wlNbOfBookings

Definition at line 39 of file LegCabinStruct.hpp.

Referenced by describe().

22.78.3.15 stdair::NbOfBookings_T AIRINV::LegCabinStruct::_groupNbOfBookings

Definition at line 40 of file LegCabinStruct.hpp.

Referenced by describe().

22.78.3.16 BucketStructList_T AIRINV::LegCabinStruct::_bucketList

Definition at line 41 of file LegCabinStruct.hpp.

Referenced by describe(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), and AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()().

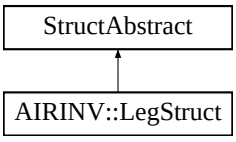
The documentation for this struct was generated from the following files:

- airinv/bom/LegCabinStruct.hpp
- airinv/bom/LegCabinStruct.cpp

22.79 AIRINV::LegStruct Struct Reference

`#include <airinv/bom/LegStruct.hpp>`

Inheritance diagram for AIRINV::LegStruct:



Public Member Functions

- void fill (const stdair::Date_T &iRefDate, stdair::LegDate &) const
- void fill (stdair::LegDate &) const
- const std::string describe () const
- LegStruct ()

Public Attributes

- stdair::AirlineCode_T _airlineCode
- stdair::FlightNumber_T _flightNumber
- stdair::AirportCode_T _boardingPoint
- stdair::DateOffset_T _boardingDateOffset
- stdair::Date_T _boardingDate
- stdair::Duration_T _boardingTime
- stdair::AirportCode_T _offPoint
- stdair::DateOffset_T _offDateOffset
- stdair::Date_T _offDate
- stdair::Duration_T _offTime
- stdair::Duration_T _elapsed
- LegCabinStructList_T _cabinList

22.79.1 Detailed Description

Utility Structure for the parsing of Leg structures.

Definition at line 24 of file LegStruct.hpp.

22.79.2 Constructor & Destructor Documentation

22.79.2.1 AIRINV::LegStruct::LegStruct ( )

Default Constructor.

Definition at line 16 of file LegStruct.cpp.

22.79.3 Member Function Documentation

22.79.3.1 void AIRINV::LegStruct::fill (const stdair::Date_T & *iRefDate,* stdair::LegDate & *ioLegDate*) const

Fill the LegDate objects with the attributes of the LegStruct.

The given reference date corresponds to the date of the FlightDate. Indeed, each Leg gets date off-sets, when compared to that (reference) flight-date, both for the boarding date and for the off date.

Definition at line 41 of file LegStruct.cpp.

References _airlineCode, _boardingDateOffset, _boardingTime, _elapsed, _flightNumber, _offDateOffset, _offPoint, and _offTime.

22.79.3.2 void AIRINV::LegStruct::fill (stdair::LegDate & *ioLegDate*) const

Fill the LegDate objects with the attributes of the LegStruct.

Definition at line 62 of file LegStruct.cpp.

References _airlineCode, _boardingTime, _elapsed, _flightNumber, _offDate, _offPoint, and _offTime.

22.79.3.3 const std::string AIRINV::LegStruct::describe ( ) const

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

Definition at line 21 of file LegStruct.cpp.

References _boardingDate, _boardingPoint, _boardingTime, _cabinList, _elapsed, _offDate, _offPoint, _offTime, and AIRINV::LegCabinStruct::describe().

Referenced by AIRINV::FlightPeriodStruct::describe(), and AIRINV::FlightDateStruct::describe().

22.79.4 Member Data Documentation

22.79.4.1 stdair::AirlineCode_T AIRINV::LegStruct::_airlineCode

Definition at line 26 of file LegStruct.hpp.

Referenced by fill(), AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeOperatingAirlineCode::operator()(), and AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()().

22.79.4.2 stdair::FlightNumber_T AIRINV::LegStruct::_flightNumber

Definition at line 27 of file LegStruct.hpp.

Referenced by fill(), AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeOperatingFlightNumber::operator()(), and AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()().

22.79.4.3 stdair::AirportCode_T AIRINV::LegStruct::_boardingPoint

Definition at line 28 of file LegStruct.hpp.

Referenced by describe(), AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator()(), and AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()().

22.79.4.4 stdair::DateOffset_T AIRINV::LegStruct::_boardingDateOffset

Definition at line 29 of file LegStruct.hpp.

Referenced by fill(), and AIRINV::ScheduleParserHelper::storeOffTime::operator()().

22.79.4.5 stdair::Date_T AIRINV::LegStruct::_boardingDate

Definition at line 30 of file LegStruct.hpp.

Referenced by describe(), and AIRINV::InventoryParserHelper::storeBoardingDate::operator()().

22.79.4.6 stdair::Duration_T AIRINV::LegStruct::_boardingTime

Definition at line 31 of file LegStruct.hpp.

Referenced by describe(), fill(), AIRINV::ScheduleParserHelper::storeBoardingTime::operator()(), and AIRINV::InventoryParserHelper::storeBoardingTime::operator()().

22.79.4.7 stdair::AirportCode_T AIRINV::LegStruct::_offPoint

Definition at line 32 of file LegStruct.hpp.

Referenced by describe(), fill(), AIRINV::ScheduleParserHelper::storeLegOffPoint::operator()(), and AIRINV::InventoryParserHelper::storeLegOffPoint::operator()().

22.79.4.8 stdair::DateOffset_T AIRINV::LegStruct::_offDateOffset

Definition at line 33 of file LegStruct.hpp.

Referenced by fill(), and AIRINV::ScheduleParserHelper::storeElapsedTime::operator()().

22.79.4.9 stdair::Date_T AIRINV::LegStruct::_offDate

Definition at line 34 of file LegStruct.hpp.

Referenced by describe(), fill(), and AIRINV::InventoryParserHelper::storeOffDate::operator()().

22.79.4.10 stdair::Duration_T AIRINV::LegStruct::_offTime

Definition at line 35 of file LegStruct.hpp.

Referenced by describe(), fill(), AIRINV::ScheduleParserHelper::storeOffTime::operator()(), and AIRINV::InventoryParserHelper::storeOffTime::operator()().

22.79.4.11 stdair::Duration_T AIRINV::LegStruct::_elapsed

Definition at line 36 of file LegStruct.hpp.

Referenced by describe(), fill(), and AIRINV::ScheduleParserHelper::storeElapsedTime::operator()().

22.79.4.12 LegCabinStructList_T AIRINV::LegStruct::_cabinList

Definition at line 37 of file LegStruct.hpp.

Referenced by describe(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeCapacity::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::ScheduleParserHelper::doEndFlight::operator()(), and AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()().

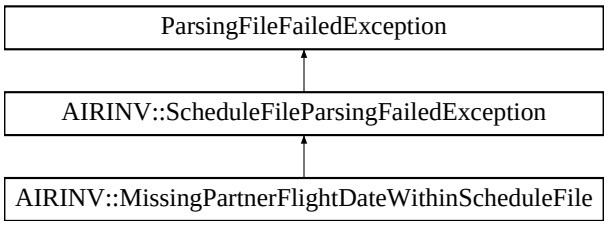
The documentation for this struct was generated from the following files:

- airinv/bom/LegStruct.hpp
- airinv/bom/LegStruct.cpp

22.80 AIRINV::MissingPartnerFlightDateWithinScheduleFile Class Reference

`#include <airinv/AIRINV_Types.hpp>`

Inheritance diagram for AIRINV::MissingPartnerFlightDateWithinScheduleFile:



Public Member Functions

- MissingPartnerFlightDateWithinScheduleFile (const std::string &iWhat)

22.80.1 Detailed Description

Missing partner flight date within the schedule file.

Definition at line 54 of file AIRINV_Types.hpp.

22.80.2 Constructor & Destructor Documentation

22.80.2.1 AIRINV::MissingPartnerFlightDateWithinScheduleFile::MissingPartnerFlightDateWithinScheduleFile (const std::string & *iWhat*) [inline]

Constructor.

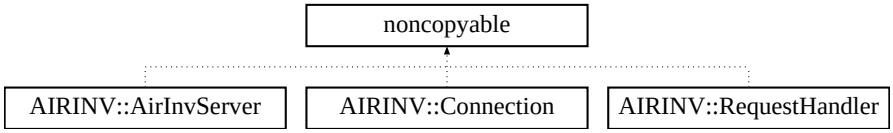
Definition at line 60 of file AIRINV_Types.hpp.

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

- airinv/AIRINV_Types.hpp

22.81 noncopyable Class Reference

Inheritance diagram for noncopyable:

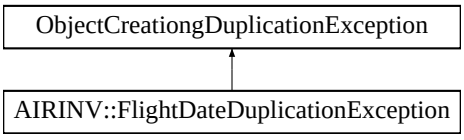


The documentation for this class was generated from the following file:

- airinv/server/Connection.hpp

22.82 ObjectCreationgDuplicationException Class Reference

Inheritance diagram for ObjectCreationgDuplicationException:

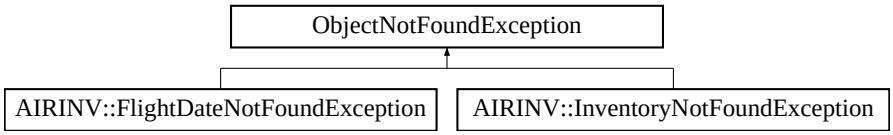


The documentation for this class was generated from the following file:

- airinv/AIRINV_Types.hpp

22.83 ObjectNotFoundException Class Reference

Inheritance diagram for ObjectNotFoundException:

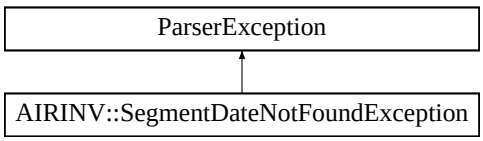


The documentation for this class was generated from the following file:

- airinv/AIRINV_Types.hpp

22.84 ParserException Class Reference

Inheritance diagram for ParserException:



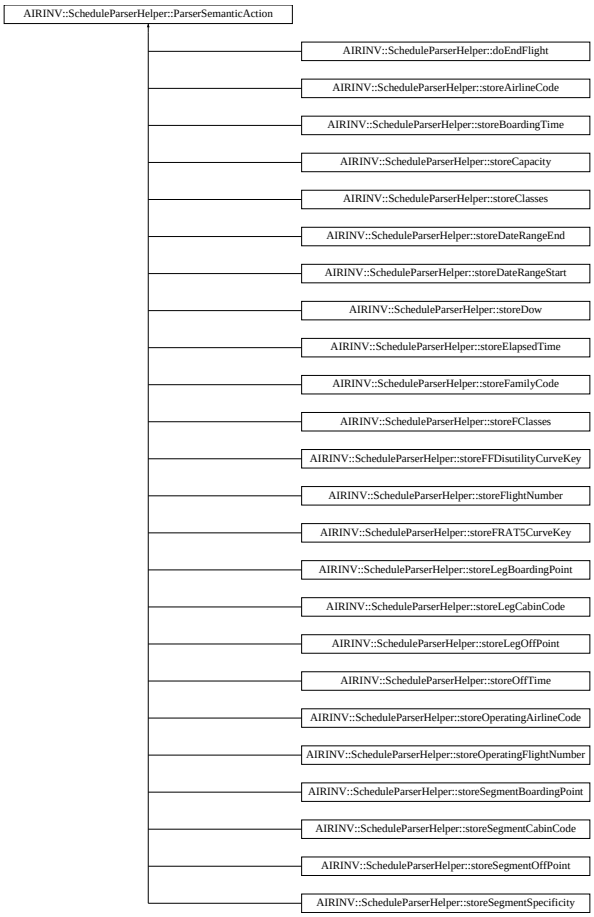
The documentation for this class was generated from the following file:

- airinv/AIRINV_Types.hpp

22.85 AIRINV::ScheduleParserHelper::ParserSemanticAction Struct Reference

`#include <airinv/command/ScheduleParserHelper.hpp>`

Inheritance diagram for AIRINV::ScheduleParserHelper::ParserSemanticAction:



Public Member Functions

- ParserSemanticAction (FlightPeriodStruct &)

Public Attributes

- FlightPeriodStruct & _flightPeriod

22.85.1 Detailed Description

Generic Semantic Action (Actor / Functor) for the Schedule Parser.

Definition at line 29 of file ScheduleParserHelper.hpp.

22.85.2 Constructor & Destructor Documentation

22.85.2.1 AIRINV::ScheduleParserHelper::ParserSemanticAction::ParserSemanticAction (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 28 of file ScheduleParserHelper.cpp.

22.85.3 Member Data Documentation

22.85.3.1 FlightPeriodStruct& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRINV::ScheduleParserHelper::storeAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRINV::ScheduleParserHelper::storeDow::operator()(), AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeBoardingTime::operator()(), AIRINV::ScheduleParserHelper::storeOffTime::operator()(), AIRINV::ScheduleParserHelper::storeElapsedTime::operator()(), AIRINV::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeCapacity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeClasses::operator()(), AIRINV::ScheduleParserHelper::storeFamilyCode::operator()(), AIRINV::ScheduleParserHelper::storeFRAT5CurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFFDisutilityCurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFClasses::operator()(), and AIRINV::ScheduleParserHelper::doEndFlight::operator()().

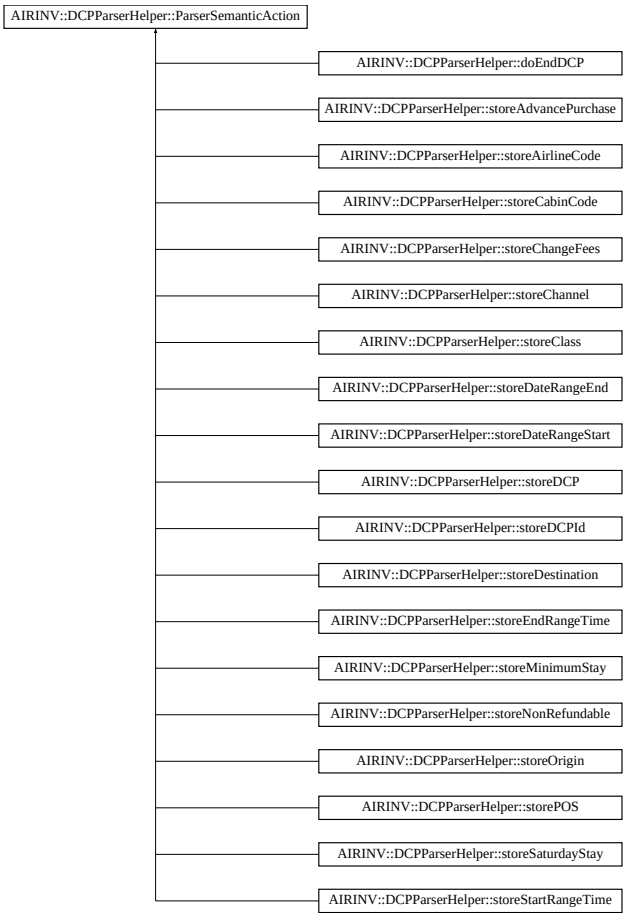
The documentation for this struct was generated from the following files:

- airinv/command/ScheduleParserHelper.hpp
- airinv/command/ScheduleParserHelper.cpp

22.86 AIRINV::DCPParserHelper::ParserSemanticAction Struct Reference

`#include <airinv/command/vault/DCPParserHelper.hpp>`

Inheritance diagram for AIRINV::DCPParserHelper::ParserSemanticAction:



Public Member Functions

- ParserSemanticAction (DCPRuleStruct &)

Public Attributes

- DCPRuleStruct & _DCPRule

22.86.1 Detailed Description

Generic Semantic Action (Actor / Functor) for the DCP Parser.

Definition at line 30 of file DCPParserHelper.hpp.

22.86.2 Constructor & Destructor Documentation

22.86.2.1 AIRINV::DCPParserHelper::ParserSemanticAction::ParserSemanticAction (DCPRuleStruct & *ioDCPRule*)

Actor Constructor.

Definition at line 25 of file DCPParserHelper.cpp.

22.86.3 Member Data Documentation

22.86.3.1 DCPRuleStruct& AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule

Actor Context.

Definition at line 34 of file DCPParserHelper.hpp.

Referenced by AIRINV::DCPParserHelper::storeDCPId::operator()(), AIRINV::DCPParserHelper::storeOrigin::operator()(), AIRINV::DCPParserHelper::storeDestination::operator()(), AIRINV::DCPParserHelper::storeDateRangeStart::operator()(), AIRINV::DCPParserHelper::storeDateRangeEnd::operator()(), AIRINV::DCPParserHelper::storeStartRangeTime::operator()(), AIRINV::DCPParserHelper::storeEndRangeTime::operator()(), AIRINV::DCPParserHelper::storePOS::operator()(), AIRINV::DCPParserHelper::storeCabinCode::operator()(), AIRINV::DCPParserHelper::storeChannel::operator()(), AIRINV::DCPParserHelper::storeAdvancePurchase::operator()(), AIRINV::DCPParserHelper::storeSaturdayStay::operator()(), AIRINV::DCPParserHelper::storeChangeFees::operator()(), AIRINV::DCPParserHelper::storeNonRefundable::operator()(), AIRINV::DCPParserHelper::storeMinimumStay::operator()(), AIRINV::DCPParserHelper::storeDCP::operator()(), AIRINV::DCPParserHelper::storeAirlineCode::operator()(), AIRINV::DCPParserHelper::storeClass::operator()(), and AIRINV::DCPParserHelper::doEndDCP::operator()().

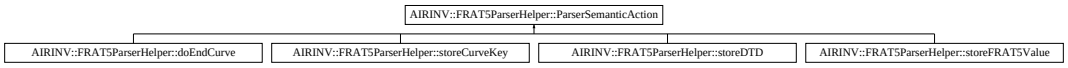
The documentation for this struct was generated from the following files:

- airinv/command/vault/DCPParserHelper.hpp
- airinv/command/vault/DCPParserHelper.cpp

22.87 AIRINV::FRAT5ParserHelper::ParserSemanticAction Struct Reference

`#include <airinv/command/FRAT5ParserHelper.hpp>`

Inheritance diagram for AIRINV::FRAT5ParserHelper::ParserSemanticAction:



Public Member Functions

- ParserSemanticAction (FRAT5Struct &)

Public Attributes

- FRAT5Struct & _frat5

22.87.1 Detailed Description

Generic Semantic Action (Actor / Functor) for the FRAT5 Parser.

Definition at line 29 of file FRAT5ParserHelper.hpp.

22.87.2 Constructor & Destructor Documentation

22.87.2.1 AIRINV::FRAT5ParserHelper::ParserSemanticAction::ParserSemanticAction (FRAT5Struct & *ioFRAT5*)

Actor Constructor.

Definition at line 27 of file FRAT5ParserHelper.cpp.

22.87.3 Member Data Documentation

22.87.3.1 FRAT5Struct& AIRINV::FRAT5ParserHelper::ParserSemanticAction::_frat5

Actor Context.

Definition at line 33 of file FRAT5ParserHelper.hpp.

Referenced by AIRINV::FRAT5ParserHelper::storeCurveKey::operator()(), AIRINV::FRAT5ParserHelper::storeDTD::operator()(), AIRINV::FRAT5ParserHelper::storeFRAT5Value::operator()(), and AIRINV::FRAT5ParserHelper::doEndCurve::operator()().

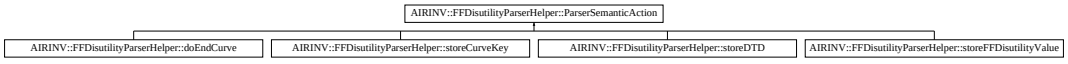
The documentation for this struct was generated from the following files:

- airinv/command/FRAT5ParserHelper.hpp
- airinv/command/FRAT5ParserHelper.cpp

22.88 AIRINV::FFDisutilityParserHelper::ParserSemanticAction Struct Reference

`#include <airinv/command/FFDisutilityParserHelper.hpp>`

Inheritance diagram for AIRINV::FFDisutilityParserHelper::ParserSemanticAction:



Public Member Functions

- ParserSemanticAction (FFDisutilityStruct &)

Public Attributes

- FFDisutilityStruct & _ffDisutility

22.88.1 Detailed Description

Generic Semantic Action (Actor / Functor) for the FFDisutility Parser.

Definition at line 29 of file FFDisutilityParserHelper.hpp.

22.88.2 Constructor & Destructor Documentation

22.88.2.1 AIRINV::FFDisutilityParserHelper::ParserSemanticAction::ParserSemanticAction (FFDisutilityStruct & *ioFFDisutility*)

Actor Constructor.

Definition at line 27 of file FFDisutilityParserHelper.cpp.

22.88.3 Member Data Documentation

22.88.3.1 FFDisutilityStruct& AIRINV::FFDisutilityParserHelper::ParserSemanticAction::_ffDisutility

Actor Context.

Definition at line 33 of file FFDisutilityParserHelper.hpp.

Referenced by AIRINV::FFDisutilityParserHelper::storeCurveKey::operator()(), AIRINV::FFDisutilityParserHelper::storeDTD::operator()(), AIRINV::FFDisutilityParserHelper::storeFFDisutilityValue::operator()(), and AIRINV::FFDisutilityParserHelper::doEndCurve::operator()().

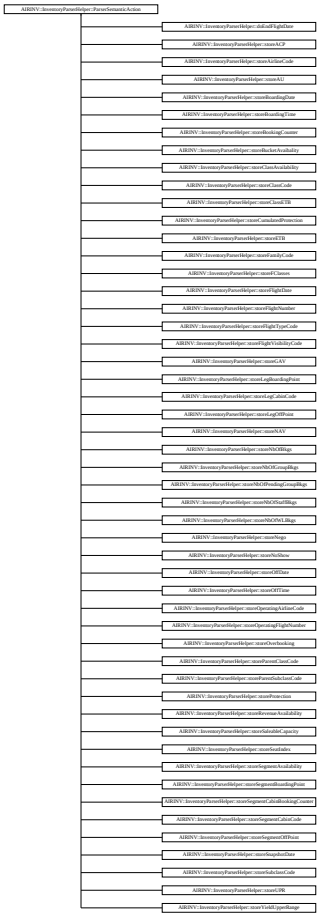
The documentation for this struct was generated from the following files:

- airinv/command/FFDisutilityParserHelper.hpp
- airinv/command/FFDisutilityParserHelper.cpp

22.89 AIRINV::InventoryParserHelper::ParserSemanticAction Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::ParserSemanticAction:



Public Member Functions

- ParserSemanticAction (FlightDateStruct &)

Public Attributes

- FlightDateStruct & _flightDate

22.89.1 Detailed Description

Generic Semantic Action (Actor / Functor) for the Inventory Parser.

Definition at line 29 of file InventoryParserHelper.hpp.

22.89.2 Constructor & Destructor Documentation

22.89.2.1 AIRINV::InventoryParserHelper::ParserSemanticAction::ParserSemanticAction (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 26 of file InventoryParserHelper.cpp.

22.89.3 Member Data Documentation

22.89.3.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

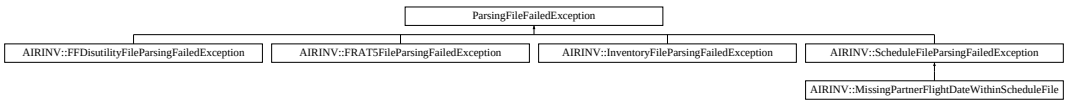
Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

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

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.90 ParsingFileFailedException Class Reference

Inheritance diagram for ParsingFileFailedException:



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

- airinv/AIRINV_Types.hpp

22.91 AIRINV::Reply Struct Reference

`#include <airinv/server/Reply.hpp>`

Public Member Functions

- std::vector
  < boost::asio::const_buffer > to_buffers ()

Public Attributes

- FlightRequestStatus::EN_FlightRequestStatus _status
- std::string content

22.91.1 Detailed Description

A reply to be sent to a client.

Definition at line 18 of file Reply.hpp.

22.91.2 Member Function Documentation

22.91.2.1 std::vector< boost::asio::const_buffer > AIRINV::Reply::to_buffers ( )

Convert the reply into a vector of buffers. The buffers do not own the underlying memory blocks, therefore the reply object must remain valid and not be changed until the write operation has completed.

Definition at line 15 of file Reply.cpp.

References content.

22.91.3 Member Data Documentation

22.91.3.1 FlightRequestStatus::EN_FlightRequestStatus AIRINV::Reply::_status

Status.

Definition at line 20 of file Reply.hpp.

Referenced by AIRINV::RequestHandler::handleRequest().

22.91.3.2 std::string AIRINV::Reply::content

The content to be sent in the reply.

Definition at line 23 of file Reply.hpp.

Referenced by AIRINV::RequestHandler::handleRequest(), and to_buffers().

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

- airinv/server/Reply.hpp
- airinv/server/Reply.cpp

22.92 AIRINV::Request Struct Reference

`#include <airinv/server/Request.hpp>`

Public Member Functions

- bool parseFlightDate ()

Public Attributes

- std::string _flightDetails
- stdair::AirlineCode_T _airlineCode
- stdair::FlightNumber_T _flightNumber
- stdair::Date_T _departureDate

22.92.1 Detailed Description

A request received from a client.

Definition at line 18 of file Request.hpp.

22.92.2 Member Function Documentation

22.92.2.1 bool AIRINV::Request::parseFlightDate ( )

Parse the incoming request.

Expected requested is of the form: <airline_code>,<flight_number>,<flight_date>, where date format is YYYY--MM-DD. For instance: BA,341,2010-09-20.

Definition at line 12 of file Request.cpp.

References _airlineCode, _departureDate, and _flightNumber.

Referenced by AIRINV::RequestHandler::handleRequest().

22.92.3 Member Data Documentation

22.92.3.1 std::string AIRINV::Request::_flightDetails

String as it comes from the connected client.

Definition at line 29 of file Request.hpp.

Referenced by AIRINV::RequestHandler::handleRequest().

22.92.3.2 stdair::AirlineCode_T AIRINV::Request::_airlineCode

Parsed airline code.

Definition at line 31 of file Request.hpp.

Referenced by parseFlightDate().

22.92.3.3 stdair::FlightNumber_T AIRINV::Request::_flightNumber

Parsed flight number.

Definition at line 33 of file Request.hpp.

Referenced by parseFlightDate().

22.92.3.4 stdair::Date_T AIRINV::Request::_departureDate

Parsed departure date.

Definition at line 35 of file Request.hpp.

Referenced by parseFlightDate().

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

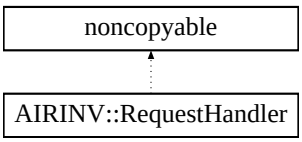
- airinv/server/Request.hpp
- airinv/server/Request.cpp

22.93 AIRINV::RequestHandler Class Reference

The common handler for all incoming requests.

`#include <airinv/server/RequestHandler.hpp>`

Inheritance diagram for AIRINV::RequestHandler:



Public Member Functions

- RequestHandler (const stdair::AirlineCode_T &)
- bool handleRequest (Request &, Reply &) const

22.93.1 Detailed Description

The common handler for all incoming requests.

Definition at line 28 of file RequestHandler.hpp.

22.93.2 Constructor & Destructor Documentation

22.93.2.1 AIRINV::RequestHandler::RequestHandler (const stdair::AirlineCode_T & *iAirlineCode*)

Constructor.

Parameters

const	stdair::AirlineCode_T& Airline code of the inventory owner.

Definition at line 20 of file RequestHandler.cpp.

22.93.3 Member Function Documentation

22.93.3.1 bool AIRINV::RequestHandler::handleRequest (Request & *ioRequest,* Reply & *ioReply*) const

Handle a request and produce a reply.

Definition at line 26 of file RequestHandler.cpp.

References AIRINV::Request::_flightDetails, AIRINV::Reply::_status, AIRINV::Reply::content, AIRINV::FlightRequestStatus::INTERNAL_ERROR, AIRINV::FlightRequestStatus::OK, and AIRINV::Request::parseFlightDate().

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

- airinv/server/RequestHandler.hpp
- airinv/server/RequestHandler.cpp

22.94 AIRINV::RequestParser Class Reference

Parser for incoming requests.

`#include <airinv/server/RequestParser.hpp>`

Public Member Functions

- RequestParser ()
  Construct ready to parse the request method.
- void reset ()
  Reset to initial parser state.
- template<typename InputIterator >
  boost::tuple< boost::tribool,
  InputIterator > parse (Request &req, InputIterator begin, InputIterator end)

22.94.1 Detailed Description

Parser for incoming requests.

Definition at line 17 of file RequestParser.hpp.

22.94.2 Constructor & Destructor Documentation

22.94.2.1 AIRINV::RequestParser::RequestParser ( )

Construct ready to parse the request method.

Definition at line 13 of file RequestParser.cpp.

22.94.3 Member Function Documentation

22.94.3.1 void AIRINV::RequestParser::reset ( )

Reset to initial parser state.

Definition at line 18 of file RequestParser.cpp.

22.94.3.2 template<typename InputIterator > boost::tuple<boost::tribool, InputIterator> AIRINV::RequestParser::parse (Request & *req*, InputIterator *begin*, InputIterator *end*) [inline]

Parse some data. The tribool return value is true when a complete request has been parsed, false if the data is invalid, indeterminate when more data is required. The InputIterator return value indicates how much of the input has been consumed.

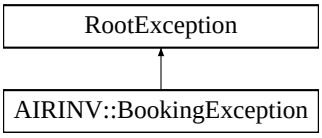
Definition at line 30 of file RequestParser.hpp.

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

- airinv/server/RequestParser.hpp
- airinv/server/RequestParser.cpp

22.95 RootException Class Reference

Inheritance diagram for RootException:



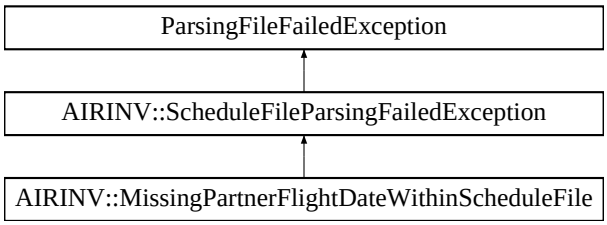
The documentation for this class was generated from the following file:

- airinv/AIRINV_Types.hpp

22.96 AIRINV::ScheduleFileParsingFailedException Class Reference

`#include <airinv/AIRINV_Types.hpp>`

Inheritance diagram for AIRINV::ScheduleFileParsingFailedException:



Public Member Functions

- ScheduleFileParsingFailedException (const std::string &iWhat)

22.96.1 Detailed Description

The schedule input file can not be parsed.

Definition at line 41 of file AIRINV_Types.hpp.

22.96.2 Constructor & Destructor Documentation

22.96.2.1 AIRINV::ScheduleFileParsingFailedException::ScheduleFileParsingFailedException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 47 of file AIRINV_Types.hpp.

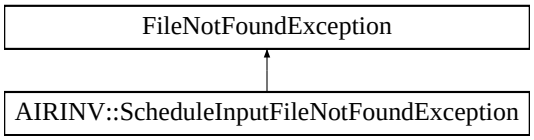
The documentation for this class was generated from the following file:

- airinv/AIRINV_Types.hpp

22.97 AIRINV::ScheduleInputFileNotFoundException Class Reference

`#include <airinv/AIRINV_Types.hpp>`

Inheritance diagram for AIRINV::ScheduleInputFileNotFoundException:



Public Member Functions

- ScheduleInputFileNotFoundException (const std::string &iWhat)

22.97.1 Detailed Description

The schedule input file can not be found or opened.

Definition at line 118 of file AIRINV_Types.hpp.

22.97.2 Constructor & Destructor Documentation

22.97.2.1 AIRINV::ScheduleInputFileNotFoundException::ScheduleInputFileNotFoundException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 123 of file AIRINV_Types.hpp.

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

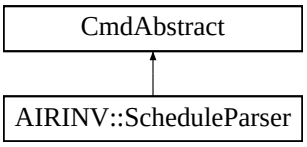
- airinv/AIRINV_Types.hpp

22.98 AIRINV::ScheduleParser Class Reference

Class wrapping the parser entry point.

`#include <airinv/command/ScheduleParser.hpp>`

Inheritance diagram for AIRINV::ScheduleParser:



Static Public Member Functions

- static void generateInventories (const stdair::ScheduleFilePath &iScheduleFilename, stdair::BomRoot &)

22.98.1 Detailed Description

Class wrapping the parser entry point.

Definition at line 22 of file ScheduleParser.hpp.

22.98.2 Member Function Documentation

22.98.2.1 void AIRINV::ScheduleParser::generateInventories (const stdair::ScheduleFilePath & *iScheduleFilename,* stdair::BomRoot & *ioBomRoot*) [static]

Parse the CSV file describing the airline schedules for the simulator, and generates the inventories accordingly.

Parameters

const	stdair::Filename_T& The file-name of the CSV-formatted schedule input file.
stdair::BomRoot&	Root of the BOM tree.

Definition at line 20 of file ScheduleParser.cpp.

References AIRINV::FlightPeriodFileParser::generateInventories().

Referenced by AIRINV::AIRINV_Service::parseAndLoad().

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

- airinv/command/ScheduleParser.hpp
- airinv/command/ScheduleParser.cpp

22.99 AIRINV::SegmentCabinHelper Class Reference

Class representing the actual business functions for an airline segment-cabin.

`#include <airinv/bom/SegmentCabinHelper.hpp>`

Static Public Member Functions

- static void updateFromReservation (const stdair::FlightDate &, stdair::SegmentCabin &, const stdair::PartySize_T &)
- static void buildPseudoBidPriceVector (stdair::SegmentCabin &)
- static void updateBookingControlsUsingPseudoBidPriceVector (const stdair::SegmentCabin &)
- static void updateAUs (const stdair::SegmentCabin &)
- static void updateAvailabilities (const stdair::SegmentCabin &)
- static void initialiseAU (stdair::SegmentCabin &)
- static void initYieldBasedNestingStructure (stdair::SegmentCabin &)
- static void initListOfUsablePolicies (stdair::SegmentCabin &)

22.99.1 Detailed Description

Class representing the actual business functions for an airline segment-cabin.

Definition at line 25 of file SegmentCabinHelper.hpp.

22.99.2 Member Function Documentation

22.99.2.1 void AIRINV::SegmentCabinHelper::updateFromReservation (const stdair::FlightDate & *iFlightDate,* stdair::SegmentCabin & *ioSegmentCabin,* const stdair::PartySize_T & *iNbOfBookings*) [static]

Update the segment-cabin with the reservation.

Definition at line 64 of file SegmentCabinHelper.cpp.

References AIRINV::FlightDateHelper::updateAvailability().

22.99.2.2 void AIRINV::SegmentCabinHelper::buildPseudoBidPriceVector (stdair::SegmentCabin & *ioSegmentCabin*) [static]

Build the pseudo bid price vector from the vectors of the leg-cabins.

Definition at line 77 of file SegmentCabinHelper.cpp.

Referenced by AIRINV::FlightDateHelper::updateBookingControls().

22.99.2.3 void AIRINV::SegmentCabinHelper::updateBookingControlsUsingPseudoBidPriceVector (const stdair::SegmentCabin & *iSegmentCabin*) [static]

Update the booking controls using the pseudo bid price vector.

Definition at line 128 of file SegmentCabinHelper.cpp.

References updateAUs().

Referenced by AIRINV::FlightDateHelper::updateBookingControls().

22.99.2.4 void AIRINV::SegmentCabinHelper::updateAUs (const stdair::SegmentCabin & *iSegmentCabin*) [static]

Update the authorisation levels using the booking limits.

Definition at line 186 of file SegmentCabinHelper.cpp.

Referenced by updateBookingControlsUsingPseudoBidPriceVector().

22.99.2.5 void AIRINV::SegmentCabinHelper::updateAvailabilities (const stdair::SegmentCabin & *iSegmentCabin*) [static]

Update the availability of the booking classes.

Definition at line 240 of file SegmentCabinHelper.cpp.

Referenced by AIRINV::FlightDateHelper::recalculateAvailability(), and AIRINV::SegmentSnapshotTableHelper-::takeSnapshots().

22.99.2.6 void AIRINV::SegmentCabinHelper::initialiseAU (stdair::SegmentCabin & *iSegmentCabin*) [static]

Initialise the AU for the booking classes.

Definition at line 28 of file SegmentCabinHelper.cpp.

Referenced by AIRINV::SegmentDateHelper::fillFromRouting().

22.99.2.7 void AIRINV::SegmentCabinHelper::initYieldBasedNestingStructure (stdair::SegmentCabin & *ioSegmentCabin*) [static]

Yield-based nesting structure initialisation.

Definition at line 333 of file SegmentCabinHelper.cpp.

Referenced by AIRINV::InventoryManager::initialiseYieldBasedNestingStructures().

22.99.2.8 void AIRINV::SegmentCabinHelper::initListOfUsablePolicies (stdair::SegmentCabin & *ioSegmentCabin*) [static]

List of usable policies initialisation.

Definition at line 385 of file SegmentCabinHelper.cpp.

Referenced by AIRINV::InventoryManager::initialiseListsOfUsablePolicies().

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

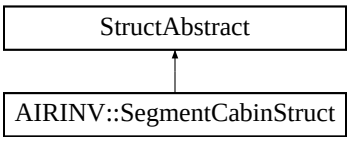
- airinv/bom/SegmentCabinHelper.hpp
- airinv/bom/SegmentCabinHelper.cpp

22.100 AIRINV::SegmentCabinStruct Struct Reference

Utility Structure for the parsing of SegmentCabin details.

`#include <airinv/bom/SegmentCabinStruct.hpp>`

Inheritance diagram for AIRINV::SegmentCabinStruct:



Public Member Functions

- void fill (stdair::SegmentCabin &) const
- const std::string describe () const

Public Attributes

- stdair::CabinCode_T _cabinCode
- stdair::NbOfBookings_T _nbOfBookings
- FareFamilyStruct _itFareFamily
- FareFamilyStructList_T _fareFamilies

22.100.1 Detailed Description

Utility Structure for the parsing of SegmentCabin details.

Definition at line 26 of file SegmentCabinStruct.hpp.

22.100.2 Member Function Documentation

22.100.2.1 void AIRINV::SegmentCabinStruct::fill (stdair::SegmentCabin & *ioSegmentCabin*) const

Fill the SegmentCabin objects with the attributes of the SegmentCabinStruct.

Definition at line 33 of file SegmentCabinStruct.cpp.

22.100.2.2 const std::string AIRINV::SegmentCabinStruct::describe ( ) const

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

Definition at line 15 of file SegmentCabinStruct.cpp.

References _cabinCode, _fareFamilies, _nbOfBookings, and AIRINV::FareFamilyStruct::describe().

Referenced by AIRINV::SegmentStruct::describe().

22.100.3 Member Data Documentation

22.100.3.1 stdair::CabinCode_T AIRINV::SegmentCabinStruct::_cabinCode

Definition at line 28 of file SegmentCabinStruct.hpp.

Referenced by AIRINV::FlightPeriodStruct::addFareFamily(), AIRINV::FlightDateStruct::addFareFamily(), describe(), AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator()(), and AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()().

22.100.3.2 stdair::NbOfBookings_T AIRINV::SegmentCabinStruct::_nbOfBookings

Definition at line 29 of file SegmentCabinStruct.hpp.

Referenced by describe(), and AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()().

22.100.3.3 FareFamilyStruct AIRINV::SegmentCabinStruct::_itFareFamily

Definition at line 30 of file SegmentCabinStruct.hpp.

Referenced by AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeClasses::operator()(), AIRINV::ScheduleParserHelper::storeFamilyCode::operator()(), AIRINV::ScheduleParserHelper::storeFRAT5CurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFFDisutilityCurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFClasses::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), and AIRINV::InventoryParserHelper::storeFClasses::operator()().

22.100.3.4 FareFamilyStructList_T AIRINV::SegmentCabinStruct::_fareFamilies

Definition at line 31 of file SegmentCabinStruct.hpp.

Referenced by AIRINV::FlightPeriodStruct::addFareFamily(), AIRINV::FlightDateStruct::addFareFamily(), describe(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), and AIRINV::InventoryParserHelper::storeFClasses::operator()().

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

- airinv/bom/SegmentCabinStruct.hpp
- airinv/bom/SegmentCabinStruct.cpp

22.101 AIRINV::SegmentDateHelper Class Reference

`#include <airinv/bom/SegmentDateHelper.hpp>`

Static Public Member Functions

- static void fillFromRouting (stdair::SegmentDate &)
- static void updateElapsedTimeFromRouting (stdair::SegmentDate &)
- static void updateDistanceFromElapsedTime (stdair::SegmentDate &)

22.101.1 Detailed Description

Class representing the actual business functions for an airline segment-date.

Definition at line 16 of file SegmentDateHelper.hpp.

22.101.2 Member Function Documentation

22.101.2.1 void AIRINV::SegmentDateHelper::fillFromRouting (stdair::SegmentDate & *ioSegmentDate*) [static]

Fill the attributes derived from the routing legs (e.g., board and off dates).

Definition at line 18 of file SegmentDateHelper.cpp.

References AIRINV::SegmentCabinHelper::initialiseAU(), and updateElapsedTimeFromRouting().

22.101.2.2 void AIRINV::SegmentDateHelper::updateElapsedTimeFromRouting (stdair::SegmentDate & *ioSegmentDate*) [static]

Calculate and set the elapsed time according to the leg routing.

Actually, the elapsed time of the segment is the sum of the elapsed times of the routing legs, plus the stop-over times. The stop-over time is the difference between the board time of a routing leg, and the off time of the previous leg. That is, it is the time spent at the corresponding airport.

Of course, in case of mono-leg segments, there is no stop-over, and the elapsed time of the segment is equal to the elapsed time of the single routing leg.

Definition at line 72 of file SegmentDateHelper.cpp.

References updateDistanceFromElapsedTime().

Referenced by fillFromRouting().

22.101.2.3 void AIRINV::SegmentDateHelper::updateDistanceFromElapsedTime (stdair::SegmentDate & *ioSegmentDate*) [static]

Method computing the distance of the segment (in kilometers).

Definition at line 115 of file SegmentDateHelper.cpp.

Referenced by updateElapsedTimeFromRouting().

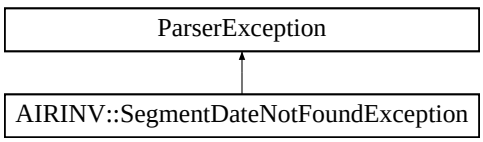
The documentation for this class was generated from the following files:

- airinv/bom/SegmentDateHelper.hpp
- airinv/bom/SegmentDateHelper.cpp

22.102 AIRINV::SegmentDateNotFoundException Class Reference

`#include <airinv/AIRINV_Types.hpp>`

Inheritance diagram for AIRINV::SegmentDateNotFoundException:



Public Member Functions

- SegmentDateNotFoundException (const std::string &iWhat)

22.102.1 Detailed Description

Specific exception when some BOM objects can not be found within the inventory.

Definition at line 94 of file AIRINV_Types.hpp.

22.102.2 Constructor & Destructor Documentation

22.102.2.1 AIRINV::SegmentDateNotFoundException::SegmentDateNotFoundException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 99 of file AIRINV_Types.hpp.

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

- airinv/AIRINV_Types.hpp

22.103 AIRINV::SegmentSnapshotTableHelper Class Reference

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

Static Public Member Functions

- static void takeSnapshots (stdair::SegmentSnapshotTable &, const stdair::DateTime_T &)

22.103.1 Detailed Description

Class representing the actual business functions for an airline inventory.

Definition at line 22 of file SegmentSnapshotTableHelper.hpp.

22.103.2 Member Function Documentation

22.103.2.1 void AIRINV::SegmentSnapshotTableHelper::takeSnapshots (stdair::SegmentSnapshotTable & *ioSegmentSnapshotTable,* const stdair::DateTime_T & *iSnapshotTime*) [static]

Take inventory snapshots.

Definition at line 27 of file SegmentSnapshotTableHelper.cpp.

References AIRINV::SegmentCabinHelper::updateAvailabilities().

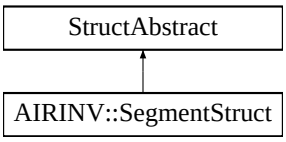
The documentation for this class was generated from the following files:

- airinv/bom/SegmentSnapshotTableHelper.hpp
- airinv/bom/SegmentSnapshotTableHelper.cpp

22.104 AIRINV::SegmentStruct Struct Reference

`#include <airinv/bom/SegmentStruct.hpp>`

Inheritance diagram for AIRINV::SegmentStruct:



Public Member Functions

- void fill (stdair::SegmentDate &) const
- const std::string describe () const

Public Attributes

- stdair::AirportCode_T _boardingPoint
- stdair::AirportCode_T _offPoint
- stdair::Date_T _boardingDate
- stdair::Duration_T _boardingTime
- stdair::Date_T _offDate
- stdair::Duration_T _offTime
- stdair::Duration_T _elapsed
- SegmentCabinStructList_T _cabinList

22.104.1 Detailed Description

Utility Structure for the parsing of Segment structures.

Definition at line 23 of file SegmentStruct.hpp.

22.104.2 Member Function Documentation

22.104.2.1 void AIRINV::SegmentStruct::fill (stdair::SegmentDate & *ioSegmentDate*) const

Fill the SegmentDate objects with the attributes of the SegmentStruct.

Definition at line 36 of file SegmentStruct.cpp.

References _boardingTime, _elapsed, _offDate, and _offTime.

22.104.2.2 const std::string AIRINV::SegmentStruct::describe ( ) const

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

Definition at line 14 of file SegmentStruct.cpp.

References _boardingPoint, _boardingTime, _cabinList, _elapsed, _offPoint, _offTime, and AIRINV::SegmentCabinStruct::describe().

Referenced by AIRINV::FlightPeriodStruct::describe(), and AIRINV::FlightDateStruct::describe().

22.104.3 Member Data Documentation

22.104.3.1 stdair::AirportCode_T AIRINV::SegmentStruct::_boardingPoint

Definition at line 25 of file SegmentStruct.hpp.

Referenced by AIRINV::FlightPeriodStruct::addFareFamily(), AIRINV::FlightDateStruct::addFareFamily(), AIRINV::FlightPeriodStruct::addSegmentCabin(), AIRINV::FlightDateStruct::addSegmentCabin(), AIRINV::FlightPeriodStruct::buildSegments(), AIRINV::FlightDateStruct::buildSegments(), describe(), AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), and AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()().

22.104.3.2 stdair::AirportCode_T AIRINV::SegmentStruct::_offPoint

Definition at line 26 of file SegmentStruct.hpp.

Referenced by AIRINV::FlightPeriodStruct::addFareFamily(), AIRINV::FlightDateStruct::addFareFamily(), AIRINV::FlightPeriodStruct::addSegmentCabin(), AIRINV::FlightDateStruct::addSegmentCabin(), AIRINV::FlightPeriodStruct::buildSegments(), AIRINV::FlightDateStruct::buildSegments(), describe(), AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator()(), and AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()().

22.104.3.3 stdair::Date_T AIRINV::SegmentStruct::_boardingDate

Definition at line 27 of file SegmentStruct.hpp.

22.104.3.4 stdair::Duration_T AIRINV::SegmentStruct::_boardingTime

Definition at line 28 of file SegmentStruct.hpp.

Referenced by describe(), and fill().

22.104.3.5 stdair::Date_T AIRINV::SegmentStruct::_offDate

Definition at line 29 of file SegmentStruct.hpp.

Referenced by fill().

22.104.3.6 stdair::Duration_T AIRINV::SegmentStruct::_offTime

Definition at line 30 of file SegmentStruct.hpp.

Referenced by describe(), and fill().

22.104.3.7 stdair::Duration_T AIRINV::SegmentStruct::_elapsed

Definition at line 31 of file SegmentStruct.hpp.

Referenced by describe(), and fill().

22.104.3.8 SegmentCabinStructList_T AIRINV::SegmentStruct::_cabinList

Definition at line 32 of file SegmentStruct.hpp.

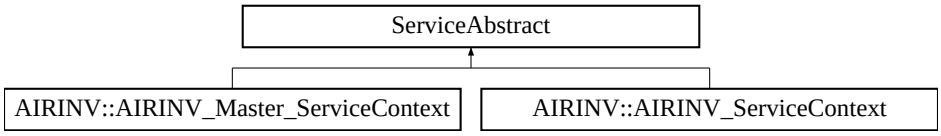
Referenced by AIRINV::FlightPeriodStruct::addFareFamily(), AIRINV::FlightDateStruct::addFareFamily(), AIRINV::FlightPeriodStruct::addSegmentCabin(), AIRINV::FlightDateStruct::addSegmentCabin(), describe(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

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

- airinv/bom/SegmentStruct.hpp
- airinv/bom/SegmentStruct.cpp

22.105 ServiceAbstract Class Reference

Inheritance diagram for ServiceAbstract:



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

- airinv/service/AIRINV_Master_ServiceContext.hpp

22.106 AIRINV::ServiceAbstract Class Reference

`#include <airinv/service/ServiceAbstract.hpp>`

Public Member Functions

- virtual ~ServiceAbstract ()
- virtual void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

Protected Member Functions

- ServiceAbstract ()

22.106.1 Detailed Description

Base class for the Service layer.

Definition at line 14 of file ServiceAbstract.hpp.

22.106.2 Constructor & Destructor Documentation

22.106.2.1 virtual AIRINV::ServiceAbstract::~ServiceAbstract ( ) [inline], [virtual]

Destructor.

Definition at line 18 of file ServiceAbstract.hpp.

22.106.2.2 AIRINV::ServiceAbstract::ServiceAbstract ( ) [inline], [protected]

Protected Default Constructor to ensure this class is abtract.

Definition at line 30 of file ServiceAbstract.hpp.

22.106.3 Member Function Documentation

22.106.3.1 virtual void AIRINV::ServiceAbstract::toStream (std::ostream & *ioOut*) const [inline], [virtual]

Dump a Business Object into an output stream.

Parameters

ostream&	the output stream.

Definition at line 22 of file ServiceAbstract.hpp.

22.106.3.2 virtual void AIRINV::ServiceAbstract::fromStream (std::istream & *ioIn*) [inline], [virtual]

Read a Business Object from an input stream.

Parameters

istream&	the input stream.

Definition at line 26 of file ServiceAbstract.hpp.

Referenced by operator>>().

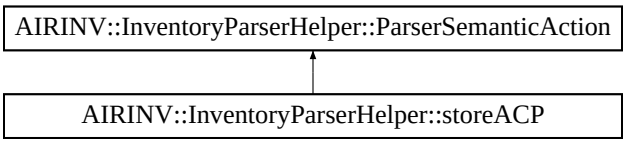
The documentation for this class was generated from the following file:

- airinv/service/ServiceAbstract.hpp

22.107 AIRINV::InventoryParserHelper::storeACP Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeACP:



Public Member Functions

- storeACP (FlightDateStruct &)
- void operator() (double iReal) const

Public Attributes

- FlightDateStruct & _flightDate

22.107.1 Detailed Description

Store the parsed Average Cancellation Percentage (ACP).

Definition at line 205 of file InventoryParserHelper.hpp.

22.107.2 Constructor & Destructor Documentation

22.107.2.1 AIRINV::InventoryParserHelper::storeACP::storeACP (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 350 of file InventoryParserHelper.cpp.

22.107.3 Member Function Documentation

22.107.3.1 void AIRINV::InventoryParserHelper::storeACP::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 355 of file InventoryParserHelper.cpp.

References AIRINV::LegCabinStruct::_acp, AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, and AIRINV::FlightDateStruct::_itLegCabin.

22.107.4 Member Data Documentation

22.107.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

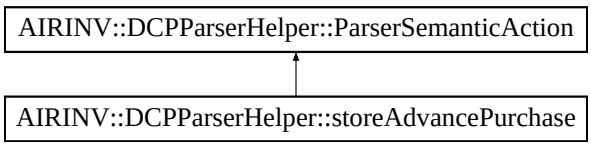
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.108 AIRINV::DCPParserHelper::storeAdvancePurchase Struct Reference

`#include <airinv/command/vault/DCPParserHelper.hpp>`

Inheritance diagram for AIRINV::DCPParserHelper::storeAdvancePurchase:



Public Member Functions

- storeAdvancePurchase (DCPRuleStruct &)
- void operator() (unsigned int, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- DCPRuleStruct & _DCPRule

22.108.1 Detailed Description

Store the parsed advance purchase days.

Definition at line 138 of file DCPParserHelper.hpp.

22.108.2 Constructor & Destructor Documentation

22.108.2.1 AIRINV::DCPParserHelper::storeAdvancePurchase::storeAdvancePurchase (DCPRuleStruct & *ioDCPRule*)

Actor Constructor.

Definition at line 208 of file DCPParserHelper.cpp.

22.108.3 Member Function Documentation

22.108.3.1 void AIRINV::DCPParserHelper::storeAdvancePurchase::operator() (unsigned int *iAdancePurchase,* boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 213 of file DCPParserHelper.cpp.

References AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule.

22.108.4 Member Data Documentation

22.108.4.1 DCPRuleStruct& AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule [inherited]

Actor Context.

Definition at line 34 of file DCPParserHelper.hpp.

Referenced by AIRINV::DCPParserHelper::storeDCPId::operator()(), AIRINV::DCPParserHelper::storeOrigin::operator()(), AIRINV::DCPParserHelper::storeDestination::operator()(), AIRINV::DCPParserHelper::storeDateRangeStart::operator()(), AIRINV::DCPParserHelper::storeDateRangeEnd::operator()(), AIRINV::DCPParserHelper::storeStartRangeTime::operator()(), AIRINV::DCPParserHelper::storeEndRangeTime::operator()(), AIRINV::DCPParserHelper::storePOS::operator()(), AIRINV::DCPParserHelper::storeCabinCode::operator()(), AIRINV::DCPParserHelper::storeChannel::operator()(), operator()(), AIRINV::DCPParserHelper::storeSaturdayStay::operator()(), AIRINV::DCPParserHelper::storeChangeFees::operator()(), AIRINV::DCPParserHelper::storeNonRefundable::operator()(), AIRINV::DCPParserHelper::storeMinimumStay::operator()(), AIRINV::DCPParserHelper-

::storeDCP::operator()(), AIRINV::DCPParserHelper::storeAirlineCode::operator()(), AIRINV::DCPParserHelper::storeClass::operator()(), and AIRINV::DCPParserHelper::doEndDCP::operator()().

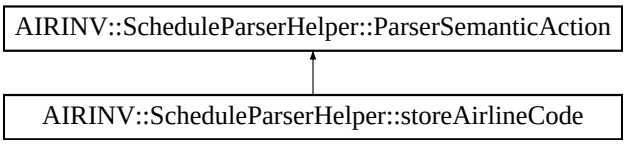
The documentation for this struct was generated from the following files:

- airinv/command/vault/DCPParserHelper.hpp
- airinv/command/vault/DCPParserHelper.cpp

22.109 AIRINV::ScheduleParserHelper::storeAirlineCode Struct Reference

`#include <airinv/command/ScheduleParserHelper.hpp>`

Inheritance diagram for AIRINV::ScheduleParserHelper::storeAirlineCode:



Public Member Functions

- storeAirlineCode (FlightPeriodStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

22.109.1 Detailed Description

Store the parsed airline code.

Definition at line 37 of file ScheduleParserHelper.hpp.

22.109.2 Constructor & Destructor Documentation

22.109.2.1 AIRINV::ScheduleParserHelper::storeAirlineCode::storeAirlineCode (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 34 of file ScheduleParserHelper.cpp.

22.109.3 Member Function Documentation

22.109.3.1 void AIRINV::ScheduleParserHelper::storeAirlineCode::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 39 of file ScheduleParserHelper.cpp.

References AIRINV::FlightPeriodStruct::_airlineCode, AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod, and AIRINV::FlightPeriodStruct::_legList.

22.109.4 Member Data Documentation

22.109.4.1 FlightPeriodStruct& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by operator()(), AIRINV::ScheduleParserHelper::storeFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRINV::ScheduleParserHelper::storeDow::operator()(), AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeBoardingTime::operator()(), AIRINV::ScheduleParserHelper::storeOffTime::operator()(), AIRINV::ScheduleParserHelper::storeElapsedTime::operator()(), AIRINV::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeCapacity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeClasses::operator()(), AIRINV::ScheduleParserHelper::storeFamilyCode::operator()(), AIRINV::ScheduleParserHelper::storeFRAT5CurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFFDisutilityCurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFClasses::operator()(), and AIRINV::ScheduleParserHelper::doEndFlight::operator()().

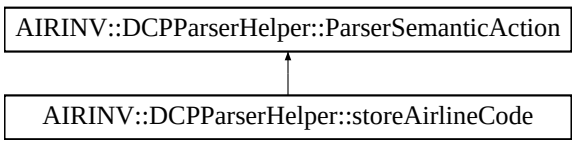
The documentation for this struct was generated from the following files:

- airinv/command/ScheduleParserHelper.hpp
- airinv/command/ScheduleParserHelper.cpp

22.110 AIRINV::DCPParserHelper::storeAirlineCode Struct Reference

`#include <airinv/command/vault/DCPParserHelper.hpp>`

Inheritance diagram for AIRINV::DCPParserHelper::storeAirlineCode:



Public Member Functions

- storeAirlineCode (DCPRuleStruct &)
- void operator() (std::vector< char >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- DCPRuleStruct & _DCPRule

22.110.1 Detailed Description

Store the parsed airline code.

Definition at line 198 of file DCPParserHelper.hpp.

22.110.2 Constructor & Destructor Documentation

22.110.2.1 AIRINV::DCPParserHelper::storeAirlineCode::storeAirlineCode (DCPRuleStruct & *ioDCPRule*)

Actor Constructor.

Definition at line 329 of file DCPParserHelper.cpp.

22.110.3 Member Function Documentation

22.110.3.1 void AIRINV::DCPParserHelper::storeAirlineCode::operator() (std::vector< char > *iChar,* boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 334 of file DCPParserHelper.cpp.

References AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule.

22.110.4 Member Data Documentation

22.110.4.1 DCPRuleStruct& AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule [inherited]

Actor Context.

Definition at line 34 of file DCPParserHelper.hpp.

Referenced by AIRINV::DCPParserHelper::storeDCPId::operator()(), AIRINV::DCPParserHelper::storeOrigin::operator()(), AIRINV::DCPParserHelper::storeDestination::operator()(), AIRINV::DCPParserHelper::storeDateRangeStart::operator()(), AIRINV::DCPParserHelper::storeDateRangeEnd::operator()(), AIRINV::DCPParserHelper::storeStartRangeTime::operator()(), AIRINV::DCPParserHelper::storeEndRangeTime::operator()(), AIRINV::DCPParserHelper::storePOS::operator()(), AIRINV::DCPParserHelper::storeCabinCode::operator()(), AIRINV::DCPParserHelper::storeChannel::operator()(), AIRINV::DCPParserHelper::storeAdvancePurchase::operator()(), AIRINV::DCPParserHelper::storeSaturdayStay::operator()(), AIRINV::DCPParserHelper::storeChangeFees::operator()(), AIRINV::DCPParserHelper::storeNonRefundable::operator()(), AIRINV::DCPParserHelper::storeMinimumStay::operator()(), AIRINV::DCPParserHelper::storeDCP::operator()(), operator()(), AIRINV::DCPParserHelper::storeClass::operator()(), and AIRINV::DCPParserHelper::doEndDCP::operator()().

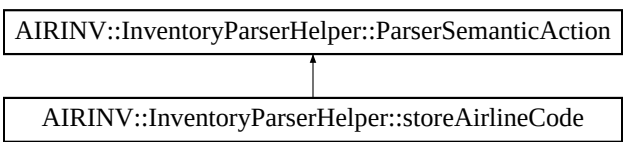
The documentation for this struct was generated from the following files:

- airinv/command/vault/DCPParserHelper.hpp
- airinv/command/vault/DCPParserHelper.cpp

22.111 AIRINV::InventoryParserHelper::storeAirlineCode Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeAirlineCode:



Public Member Functions

- storeAirlineCode (FlightDateStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightDateStruct & _flightDate

22.111.1 Detailed Description

Store the parsed airline code.

Definition at line 45 of file InventoryParserHelper.hpp.

22.111.2 Constructor & Destructor Documentation

22.111.2.1 AIRINV::InventoryParserHelper::storeAirlineCode::storeAirlineCode (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 44 of file InventoryParserHelper.cpp.

22.111.3 Member Function Documentation

22.111.3.1 void AIRINV::InventoryParserHelper::storeAirlineCode::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 49 of file InventoryParserHelper.cpp.

References AIRINV::FlightDateStruct::_airlineCode, AIRINV::LegCabinStruct::_bucketList, AIRINV::SegmentStruct::_cabinList, AIRINV::LegStruct::_cabinList, AIRINV::BookingClassStruct::_classCode, AIRINV::FareFamilyStruct::_classList, AIRINV::SegmentCabinStruct::_fareFamilies, AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::FlightDateStruct::_itBookingClass, AIRINV::FlightDateStruct::_itBucket, AIRINV::SegmentCabinStruct::_itFareFamily, AIRINV::FlightDateStruct::_itLeg, AIRINV::FlightDateStruct::_itLegCabin, AIRINV::FlightDateStruct::_itSegment, AIRINV::FlightDateStruct::_itSegmentCabin, AIRINV::FlightDateStruct::_legList, AIRINV::FlightDateStruct::_segmentList, and AIRINV::BucketStruct::_yieldRangeUpperValue.

22.111.4 Member Data Documentation

22.111.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegment-

CabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

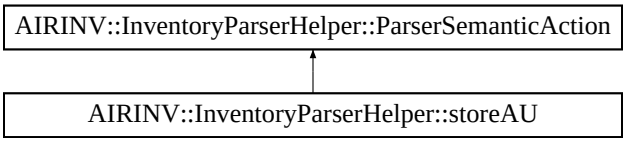
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.112 AIRINV::InventoryParserHelper::storeAU Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeAU:



Public Member Functions

- storeAU (FlightDateStruct &)
- void operator() (double iReal) const

Public Attributes

- FlightDateStruct & _flightDate

22.112.1 Detailed Description

Store the parsed Authorisation Level (AU).

Definition at line 165 of file InventoryParserHelper.hpp.

22.112.2 Constructor & Destructor Documentation

22.112.2.1 AIRINV::InventoryParserHelper::storeAU::storeAU (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 295 of file InventoryParserHelper.cpp.

22.112.3 Member Function Documentation

22.112.3.1 void AIRINV::InventoryParserHelper::storeAU::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 300 of file InventoryParserHelper.cpp.

References AIRINV::LegCabinStruct::_au, AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, and AIRINV::FlightDateStruct::_itLegCabin.

22.112.4 Member Data Documentation

22.112.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

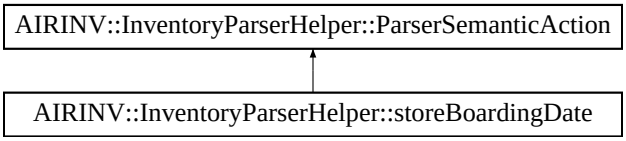
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.113 AIRINV::InventoryParserHelper::storeBoardingDate Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeBoardingDate:



Public Member Functions

- storeBoardingDate (FlightDateStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightDateStruct & _flightDate

22.113.1 Detailed Description

Store the boarding date.

Definition at line 117 of file InventoryParserHelper.hpp.

22.113.2 Constructor & Destructor Documentation

22.113.2.1 AIRINV::InventoryParserHelper::storeBoardingDate::storeBoardingDate (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 204 of file InventoryParserHelper.cpp.

22.113.3 Member Function Documentation

22.113.3.1 void AIRINV::InventoryParserHelper::storeBoardingDate::operator() (iterator_t *iStr*, iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 209 of file InventoryParserHelper.cpp.

References AIRINV::LegStruct::_boardingDate, AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::FlightDateStruct::_itLeg, and AIRINV::FlightDateStruct::getDate().

22.113.4 Member Data Documentation

22.113.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(),

AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

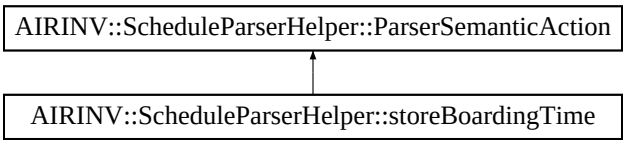
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.114 AIRINV::ScheduleParserHelper::storeBoardingTime Struct Reference

`#include <airinv/command/ScheduleParserHelper.hpp>`

Inheritance diagram for AIRINV::ScheduleParserHelper::storeBoardingTime:



Public Member Functions

- storeBoardingTime (FlightPeriodStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

22.114.1 Detailed Description

Store the boarding time.

Definition at line 109 of file ScheduleParserHelper.hpp.

22.114.2 Constructor & Destructor Documentation

22.114.2.1 AIRINV::ScheduleParserHelper::storeBoardingTime::storeBoardingTime (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 190 of file ScheduleParserHelper.cpp.

22.114.3 Member Function Documentation

22.114.3.1 void AIRINV::ScheduleParserHelper::storeBoardingTime::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 195 of file ScheduleParserHelper.cpp.

References AIRINV::LegStruct::_boardingTime, AIRINV::FlightPeriodStruct::_dateOffset, AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod, AIRINV::FlightPeriodStruct::_itLeg, AIRINV::FlightPeriodStruct::_itSeconds, and AIRINV::FlightPeriodStruct::getTime().

22.114.4 Member Data Documentation

22.114.4.1 FlightPeriodStruct& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRINV::ScheduleParserHelper::storeAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRINV::ScheduleParserHelper::storeDow::operator()(), AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeOperatingFlightNumber::operator()(), operator()(), AIRINV::ScheduleParserHelper::storeOffTime::operator()(), AIRINV::ScheduleParserHelper::storeElapsedTime::operator()(), AIRINV::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeCapacity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeClasses::operator()(), AIRINV::ScheduleParserHelper::storeFamilyCode::operator()(), AIRINV::ScheduleParserHelper::storeFRAT5CurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFFDisutilityCurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFClasses::operator()(), and AIRINV::ScheduleParserHelper::doEndFlight::operator()().

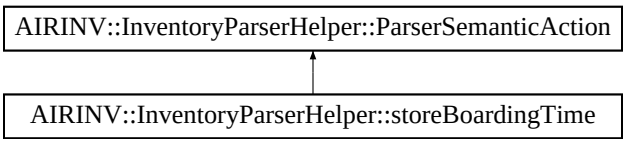
The documentation for this struct was generated from the following files:

- airinv/command/ScheduleParserHelper.hpp
- airinv/command/ScheduleParserHelper.cpp

22.115 AIRINV::InventoryParserHelper::storeBoardingTime Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeBoardingTime:



Public Member Functions

- storeBoardingTime (FlightDateStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightDateStruct & _flightDate

22.115.1 Detailed Description

Store the boarding time.

Definition at line 125 of file InventoryParserHelper.hpp.

22.115.2 Constructor & Destructor Documentation

22.115.2.1 AIRINV::InventoryParserHelper::storeBoardingTime::storeBoardingTime (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 215 of file InventoryParserHelper.cpp.

22.115.3 Member Function Documentation

22.115.3.1 void AIRINV::InventoryParserHelper::storeBoardingTime::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 220 of file InventoryParserHelper.cpp.

References AIRINV::LegStruct::_boardingTime, AIRINV::FlightDateStruct::_dateOffSet, AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::FlightDateStruct::_itLeg, AIRINV::FlightDateStruct::_itSeconds, and AIRINV::FlightDateStruct::getTime().

22.115.4 Member Data Documentation

22.115.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime-

::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

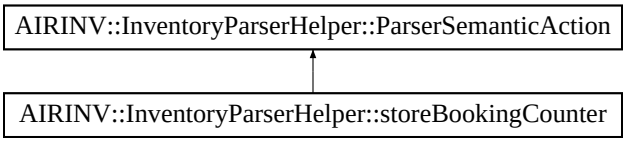
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.116 AIRINV::InventoryParserHelper::storeBookingCounter Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeBookingCounter:



Public Member Functions

- storeBookingCounter (FlightDateStruct &)
- void operator() (double iReal) const

Public Attributes

- FlightDateStruct & _flightDate

22.116.1 Detailed Description

Store the parsed booking counter.

Definition at line 181 of file InventoryParserHelper.hpp.

22.116.2 Constructor & Destructor Documentation

22.116.2.1 AIRINV::InventoryParserHelper::storeBookingCounter::storeBookingCounter (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 317 of file InventoryParserHelper.cpp.

22.116.3 Member Function Documentation

22.116.3.1 void AIRINV::InventoryParserHelper::storeBookingCounter::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 322 of file InventoryParserHelper.cpp.

References AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::FlightDateStruct::_itLegCabin, and AIRINV::LegCabinStruct::_nbOfBookings.

22.116.4 Member Data Documentation

22.116.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

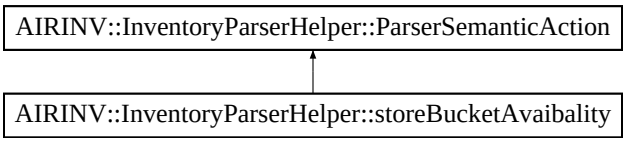
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.117 AIRINV::InventoryParserHelper::storeBucketAvaibality Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeBucketAvaibality:



Public Member Functions

- storeBucketAvaibality (FlightDateStruct &)
- void operator() (double iReal) const

Public Attributes

- FlightDateStruct & _flightDate

22.117.1 Detailed Description

Store the parsed bucket availability.

Definition at line 229 of file InventoryParserHelper.hpp.

22.117.2 Constructor & Destructor Documentation

22.117.2.1 AIRINV::InventoryParserHelper::storeBucketAvaibality::storeBucketAvaibality (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 392 of file InventoryParserHelper.cpp.

22.117.3 Member Function Documentation

22.117.3.1 void AIRINV::InventoryParserHelper::storeBucketAvaibality::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 397 of file InventoryParserHelper.cpp.

References AIRINV::BucketStruct::_availability, AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, and AIRINV::FlightDateStruct::_itBucket.

22.117.4 Member Data Documentation

22.117.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

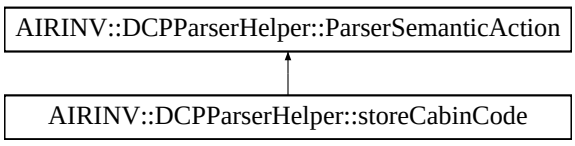
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.118 AIRINV::DCPParserHelper::storeCabinCode Struct Reference

`#include <airinv/command/vault/DCPParserHelper.hpp>`

Inheritance diagram for AIRINV::DCPParserHelper::storeCabinCode:



Public Member Functions

- storeCabinCode (DCPRuleStruct &)
- void operator() (char, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- DCPRuleStruct & _DCPRule

22.118.1 Detailed Description

Store the cabin code.

Definition at line 118 of file DCPParserHelper.hpp.

22.118.2 Constructor & Destructor Documentation

22.118.2.1 AIRINV::DCPParserHelper::storeCabinCode::storeCabinCode (DCPRuleStruct & *ioDCPRule*)

Actor Constructor.

Definition at line 166 of file DCPParserHelper.cpp.

22.118.3 Member Function Documentation

22.118.3.1 void AIRINV::DCPParserHelper::storeCabinCode::operator() (char *iChar,* boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 171 of file DCPParserHelper.cpp.

References AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule.

22.118.4 Member Data Documentation

22.118.4.1 DCPRuleStruct& AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule [inherited]

Actor Context.

Definition at line 34 of file DCPParserHelper.hpp.

Referenced by AIRINV::DCPParserHelper::storeDCPId::operator()(), AIRINV::DCPParserHelper::storeOrigin::operator()(), AIRINV::DCPParserHelper::storeDestination::operator()(), AIRINV::DCPParserHelper::storeDateRangeStart::operator()(), AIRINV::DCPParserHelper::storeDateRangeEnd::operator()(), AIRINV::DCPParserHelper::storeStartRangeTime::operator()(), AIRINV::DCPParserHelper::storeEndRangeTime::operator()(), AIRINV::DCPParserHelper::storePOS::operator()(), operator()(), AIRINV::DCPParserHelper::storeChannel::operator()(), AIRINV::DCPParserHelper::storeAdvancePurchase::operator()(), AIRINV::DCPParserHelper::storeSaturdayStay::operator()(), AIRINV::DCPParserHelper::storeChangeFees::operator()(), AIRINV::DCPParserHelper::storeNonRefundable::operator()(), AIRINV::DCPParserHelper::storeMinimumStay::operator()(), AIRINV::DCPParserHelper::storeDCP::operator()(), AIRINV::DCPParserHelper::storeAirlineCode::operator()(), AIRINV::DCPParserHelper::storeClass::operator()(), and AIRINV::DCPParserHelper::doEndDCP::operator()().

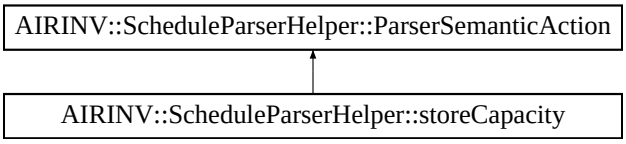
The documentation for this struct was generated from the following files:

- airinv/command/vault/DCPParserHelper.hpp
- airinv/command/vault/DCPParserHelper.cpp

22.119 AIRINV::ScheduleParserHelper::storeCapacity Struct Reference

`#include <airinv/command/ScheduleParserHelper.hpp>`

Inheritance diagram for AIRINV::ScheduleParserHelper::storeCapacity:



Public Member Functions

- storeCapacity (FlightPeriodStruct &)
- void operator() (double iReal) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

22.119.1 Detailed Description

Store the parsed capacity.

Definition at line 141 of file ScheduleParserHelper.hpp.

22.119.2 Constructor & Destructor Documentation

22.119.2.1 AIRINV::ScheduleParserHelper::storeCapacity::storeCapacity (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 262 of file ScheduleParserHelper.cpp.

22.119.3 Member Function Documentation

22.119.3.1 void AIRINV::ScheduleParserHelper::storeCapacity::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 267 of file ScheduleParserHelper.cpp.

References AIRINV::LegStruct::_cabinList, AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod, AIRINV::FlightPeriodStruct::_itLeg, AIRINV::FlightPeriodStruct::_itLegCabin, and AIRINV::LegCabinStruct::_saleableCapacity.

22.119.4 Member Data Documentation

22.119.4.1 FlightPeriodStruct& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRINV::ScheduleParserHelper::storeAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRINV::ScheduleParserHelper::storeDow::operator()(), AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeBoardingTime::operator()(), AIRINV::ScheduleParserHelper::storeOffTime::operator()(), AIRINV::ScheduleParserHelper::storeElapsedTime::operator()(), AIRINV::ScheduleParserHelper::storeLegCabinCode::operator()(), operator()(),

AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeClasses::operator()(), AIRINV::ScheduleParserHelper::storeFamilyCode::operator()(), AIRINV::ScheduleParserHelper::storeFRAT5CurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFFDisutilityCurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFClasses::operator()(), and AIRINV::ScheduleParserHelper::doEndFlight::operator()().

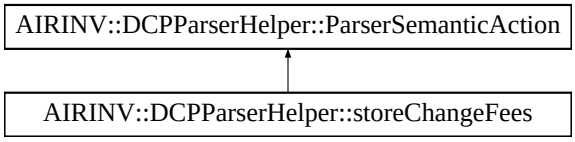
The documentation for this struct was generated from the following files:

- airinv/command/ScheduleParserHelper.hpp
- airinv/command/ScheduleParserHelper.cpp

22.120 AIRINV::DCPParserHelper::storeChangeFees Struct Reference

`#include <airinv/command/vault/DCPParserHelper.hpp>`

Inheritance diagram for AIRINV::DCPParserHelper::storeChangeFees:



Public Member Functions

- storeChangeFees (DCPRuleStruct &)
- void operator() (char, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- DCPRuleStruct & _DCPRule

22.120.1 Detailed Description

Store the parsed change fees.

Definition at line 158 of file DCPParserHelper.hpp.

22.120.2 Constructor & Destructor Documentation

22.120.2.1 AIRINV::DCPParserHelper::storeChangeFees::storeChangeFees (DCPRuleStruct & *ioDCPRule*)

Actor Constructor.

Definition at line 248 of file DCPParserHelper.cpp.

22.120.3 Member Function Documentation

22.120.3.1 void AIRINV::DCPParserHelper::storeChangeFees::operator() (char *iChangefees,* boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 253 of file DCPParserHelper.cpp.

References AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule.

22.120.4 Member Data Documentation

22.120.4.1 DCPRuleStruct& AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule [inherited]

Actor Context.

Definition at line 34 of file DCPParserHelper.hpp.

Referenced by AIRINV::DCPParserHelper::storeDCPId::operator()(), AIRINV::DCPParserHelper::storeOrigin::operator()(), AIRINV::DCPParserHelper::storeDestination::operator()(), AIRINV::DCPParserHelper::storeDateRangeStart::operator()(), AIRINV::DCPParserHelper::storeDateRangeEnd::operator()(), AIRINV::DCPParserHelper::storeStartRangeTime::operator()(), AIRINV::DCPParserHelper::storeEndRangeTime::operator()(), AIRINV::DCPParserHelper::storePOS::operator()(), AIRINV::DCPParserHelper::storeCabinCode::operator()(), AIRINV::DCPParserHelper::storeChannel::operator()(), AIRINV::DCPParserHelper::storeAdvancePurchase::operator()(), AIRINV::DCPParserHelper::storeSaturdayStay::operator()(), operator()(), AIRINV::DCPParserHelper::storeNonRefundable::operator()(), AIRINV::DCPParserHelper::storeMinimumStay::operator()(), AIRINV::DCPParserHelper::storeDCP::operator()(), AIRINV::DCPParserHelper::storeAirlineCode::operator()(), AIRINV::DCPParserHelper::storeClass::operator()(), and AIRINV::DCPParserHelper::doEndDCP::operator()().

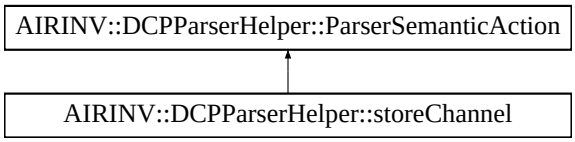
The documentation for this struct was generated from the following files:

- airinv/command/vault/DCPParserHelper.hpp
- airinv/command/vault/DCPParserHelper.cpp

22.121 AIRINV::DCPParserHelper::storeChannel Struct Reference

`#include <airinv/command/vault/DCPParserHelper.hpp>`

Inheritance diagram for AIRINV::DCPParserHelper::storeChannel:



Public Member Functions

- storeChannel (DCPRuleStruct &)
- void operator() (std::vector< char >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- DCPRuleStruct & _DCPRule

22.121.1 Detailed Description

Store the channel distribution.

Definition at line 128 of file DCPParserHelper.hpp.

22.121.2 Constructor & Destructor Documentation

22.121.2.1 AIRINV::DCPParserHelper::storeChannel::storeChannel (DCPRuleStruct & *ioDCPRule*)

Actor Constructor.

Definition at line 187 of file DCPParserHelper.cpp.

22.121.3 Member Function Documentation

22.121.3.1 void AIRINV::DCPParserHelper::storeChannel::operator() (std::vector< char > *iChar,* boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 192 of file DCPParserHelper.cpp.

References AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule.

22.121.4 Member Data Documentation

22.121.4.1 DCPRuleStruct& AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule [inherited]

Actor Context.

Definition at line 34 of file DCPParserHelper.hpp.

Referenced by AIRINV::DCPParserHelper::storeDCPId::operator()(), AIRINV::DCPParserHelper::storeOrigin::operator()(), AIRINV::DCPParserHelper::storeDestination::operator()(), AIRINV::DCPParserHelper::storeDateRangeStart::operator()(), AIRINV::DCPParserHelper::storeDateRangeEnd::operator()(), AIRINV::DCPParserHelper::storeStartRangeTime::operator()(), AIRINV::DCPParserHelper::storeEndRangeTime::operator()(), AIRINV::DCPParserHelper::storePOS::operator()(), AIRINV::DCPParserHelper::storeCabinCode::operator()(), operator()(), AIRINV::DCPParserHelper::storeAdvancePurchase::operator()(), AIRINV::DCPParserHelper::storeSaturdayStay::operator()(), AIRINV::DCPParserHelper::storeChangeFees::operator()(), AIRINV::DCPParserHelper::storeNonRefundable::operator()(), AIRINV::DCPParserHelper::storeMinimumStay::operator()(), AIRINV::DCPParserHelper::storeDCP::operator()(), AIRINV::DCPParserHelper::storeAirlineCode::operator()(), AIRINV::DCPParserHelper::storeClass::operator()(), and AIRINV::DCPParserHelper::doEndDCP::operator()().

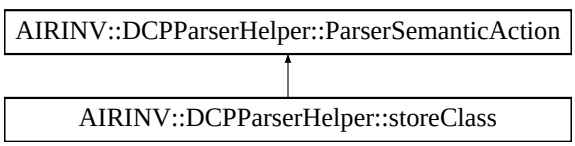
The documentation for this struct was generated from the following files:

- airinv/command/vault/DCPParserHelper.hpp
- airinv/command/vault/DCPParserHelper.cpp

22.122 AIRINV::DCPParserHelper::storeClass Struct Reference

`#include <airinv/command/vault/DCPParserHelper.hpp>`

Inheritance diagram for AIRINV::DCPParserHelper::storeClass:



Public Member Functions

- storeClass (DCPRuleStruct &)
- void operator() (std::vector< char >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- DCPRuleStruct & _DCPRule

22.122.1 Detailed Description

Store the parsed class.

Definition at line 208 of file DCPParserHelper.hpp.

22.122.2 Constructor & Destructor Documentation

22.122.2.1 AIRINV::DCPParserHelper::storeClass::storeClass (DCPRuleStruct & *ioDCPRule*)

Actor Constructor.

Definition at line 376 of file DCPParserHelper.cpp.

22.122.3 Member Function Documentation

22.122.3.1 void AIRINV::DCPParserHelper::storeClass::operator() (std::vector< char > *iChar,* boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 381 of file DCPParserHelper.cpp.

References AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule.

22.122.4 Member Data Documentation

22.122.4.1 DCPRuleStruct& AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule [inherited]

Actor Context.

Definition at line 34 of file DCPParserHelper.hpp.

Referenced by AIRINV::DCPParserHelper::storeDCPId::operator()(), AIRINV::DCPParserHelper::storeOrigin::operator()(), AIRINV::DCPParserHelper::storeDestination::operator()(), AIRINV::DCPParserHelper::storeDateRangeStart::operator()(), AIRINV::DCPParserHelper::storeDateRangeEnd::operator()(), AIRINV::DCPParserHelper::storeStartRangeTime::operator()(), AIRINV::DCPParserHelper::storeEndRangeTime::operator()(), AIRINV::DCPParserHelper::storePOS::operator()(), AIRINV::DCPParserHelper::storeCabinCode::operator()(), AIRINV::DCPParserHelper::storeChannel::operator()(), AIRINV::DCPParserHelper::storeAdvancePurchase::operator()(), AIRINV::DCPParserHelper::storeSaturdayStay::operator()(), AIRINV::DCPParserHelper::storeChangeFees::operator()(), AIRINV::DCPParserHelper::storeNonRefundable::operator()(), AIRINV::DCPParserHelper::storeMinimumStay::operator()(), AIRINV::DCPParserHelper::storeDCP::operator()(), AIRINV::DCPParserHelper::storeAirlineCode::operator()(), operator()(), and AIRINV::DCPParserHelper::doEndDCP::operator()().

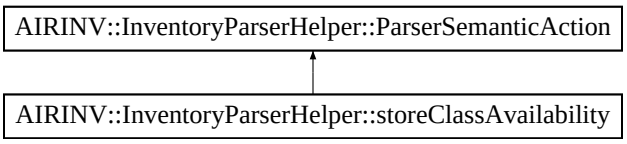
The documentation for this struct was generated from the following files:

- airinv/command/vault/DCPParserHelper.hpp
- airinv/command/vault/DCPParserHelper.cpp

22.123 AIRINV::InventoryParserHelper::storeClassAvailability Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeClassAvailability:



Public Member Functions

- storeClassAvailability (FlightDateStruct &)
- void operator() (double iReal) const

Public Attributes

- FlightDateStruct & _flightDate

22.123.1 Detailed Description

Store the parsed number of net class availability (at booking class level).

Definition at line 399 of file InventoryParserHelper.hpp.

22.123.2 Constructor & Destructor Documentation

22.123.2.1 AIRINV::InventoryParserHelper::storeClassAvailability::storeClassAvailability (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 702 of file InventoryParserHelper.cpp.

22.123.3 Member Function Documentation

22.123.3.1 void AIRINV::InventoryParserHelper::storeClassAvailability::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 707 of file InventoryParserHelper.cpp.

References AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::FlightDateStruct::_itBookingClass, and AIRINV::BookingClassStruct::_netClassAvailability.

22.123.4 Member Data Documentation

22.123.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::store-

LegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::-InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::-InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange-::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper-::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRI-NV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegment-CabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), A-IRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclass-Code::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::Inventory-ParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection-::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::store-Nego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper-::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::Inventory-ParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs-::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper-::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), operator()(), AIRI-NV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenue-Availability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParser-Helper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

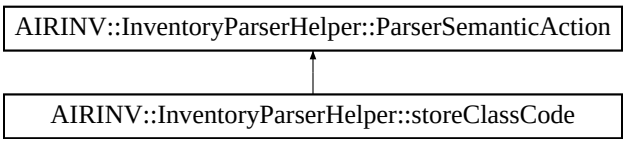
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.124 AIRINV::InventoryParserHelper::storeClassCode Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeClassCode:



Public Member Functions

- storeClassCode (FlightDateStruct &)
- void operator() (char iChar) const

Public Attributes

- FlightDateStruct & _flightDate

22.124.1 Detailed Description

Store the parsed booking class code.

Definition at line 277 of file InventoryParserHelper.hpp.

22.124.2 Constructor & Destructor Documentation

22.124.2.1 AIRINV::InventoryParserHelper::storeClassCode::storeClassCode (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 524 of file InventoryParserHelper.cpp.

22.124.3 Member Function Documentation

22.124.3.1 void AIRINV::InventoryParserHelper::storeClassCode::operator() (char *iChar*) const

Actor Function (functor).

Definition at line 529 of file InventoryParserHelper.cpp.

References AIRINV::BookingClassStruct::_classCode, AIRINV::FareFamilyStruct::_classList, AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::FlightDateStruct::_itBookingClass, AIRINV::SegmentCabinStruct::_itFareFamily, and AIRINV::FlightDateStruct::_itSegmentCabin.

22.124.4 Member Data Documentation

22.124.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

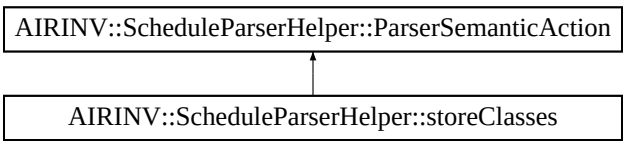
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.125 AIRINV::ScheduleParserHelper::storeClasses Struct Reference

`#include <airinv/command/ScheduleParserHelper.hpp>`

Inheritance diagram for AIRINV::ScheduleParserHelper::storeClasses:



Public Member Functions

- storeClasses (FlightPeriodStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

22.125.1 Detailed Description

Store the parsed list of class codes.

Definition at line 184 of file ScheduleParserHelper.hpp.

22.125.2 Constructor & Destructor Documentation

22.125.2.1 AIRINV::ScheduleParserHelper::storeClasses::storeClasses (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 344 of file ScheduleParserHelper.cpp.

22.125.3 Member Function Documentation

22.125.3.1 void AIRINV::ScheduleParserHelper::storeClasses::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 349 of file ScheduleParserHelper.cpp.

References AIRINV::FlightPeriodStruct::_areSegmentDefinitionsSpecific, AIRINV::FareFamilyStruct::_classes, AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod, AIRINV::SegmentCabinStruct::_itFareFamily, AIRINV::FlightPeriodStruct::_itSegment, AIRINV::FlightPeriodStruct::_itSegmentCabin, and AIRINV::FlightPeriodStruct::addSegmentCabin().

22.125.4 Member Data Documentation

22.125.4.1 FlightPeriodStruct& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRINV::ScheduleParserHelper::storeAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRINV::ScheduleParserHelper::storeDow::operator()(), AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeBoardingTime::operator()(), AIRINV::ScheduleParserHelper::storeOffTime::operator()(), AIRINV::ScheduleParserHelper::storeElapsedTime::operator()(), AIRINV::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeCapacity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator()(), operator()(), AIRINV::ScheduleParserHelper::storeFamilyCode::operator()(), AIRINV::ScheduleParserHelper::storeFRAT5CurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFFDisutilityCurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFClasses::operator()(), and AIRINV::ScheduleParserHelper::doEndFlight::operator()().

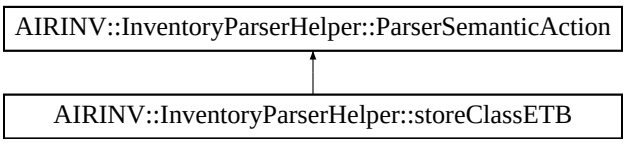
The documentation for this struct was generated from the following files:

- airinv/command/ScheduleParserHelper.hpp
- airinv/command/ScheduleParserHelper.cpp

22.126 AIRINV::InventoryParserHelper::storeClassETB Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeClassETB:



Public Member Functions

- storeClassETB (FlightDateStruct &)
- void operator() (double iReal) const

Public Attributes

- FlightDateStruct & _flightDate

22.126.1 Detailed Description

Store the parsed expected to board number (at booking class level).

Definition at line 390 of file InventoryParserHelper.hpp.

22.126.2 Constructor & Destructor Documentation

22.126.2.1 AIRINV::InventoryParserHelper::storeClassETB::storeClassETB (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 690 of file InventoryParserHelper.cpp.

22.126.3 Member Function Documentation

22.126.3.1 void AIRINV::InventoryParserHelper::storeClassETB::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 695 of file InventoryParserHelper.cpp.

References AIRINV::BookingClassStruct::_etb, AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, and AIRINV::FlightDateStruct::_itBookingClass.

22.126.4 Member Data Documentation

22.126.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

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

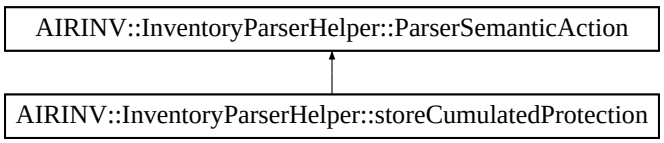
- airinv/command/InventoryParserHelper.hpp

- airinv/command/InventoryParserHelper.cpp

22.127 AIRINV::InventoryParserHelper::storeCumulatedProtection Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeCumulatedProtection:



Public Member Functions

- storeCumulatedProtection (FlightDateStruct &)
- void operator() (double iReal) const

Public Attributes

- FlightDateStruct & _flightDate

22.127.1 Detailed Description

Store the parsed cumulated protection (at booking class level).

Definition at line 309 of file InventoryParserHelper.hpp.

22.127.2 Constructor & Destructor Documentation

22.127.2.1 AIRINV::InventoryParserHelper::storeCumulatedProtection::storeCumulatedProtection (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 579 of file InventoryParserHelper.cpp.

22.127.3 Member Function Documentation

22.127.3.1 void AIRINV::InventoryParserHelper::storeCumulatedProtection::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 584 of file InventoryParserHelper.cpp.

References AIRINV::BookingClassStruct::_cumulatedProtection, AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, and AIRINV::FlightDateStruct::_itBookingClass.

22.127.4 Member Data Documentation

22.127.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

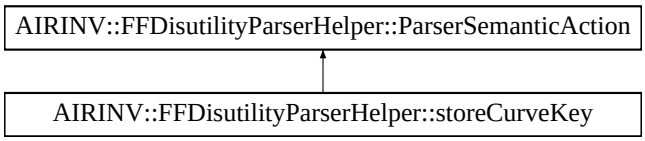
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.128 AIRINV::FFDisutilityParserHelper::storeCurveKey Struct Reference

`#include <airinv/command/FFDisutilityParserHelper.hpp>`

Inheritance diagram for AIRINV::FFDisutilityParserHelper::storeCurveKey:



Public Member Functions

- storeCurveKey (FFDisutilityStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FFDisutilityStruct & _ffDisutility

22.128.1 Detailed Description

Store the parsed curve key.

Definition at line 37 of file FFDisutilityParserHelper.hpp.

22.128.2 Constructor & Destructor Documentation

22.128.2.1 AIRINV::FFDisutilityParserHelper::storeCurveKey::storeCurveKey (FFDisutilityStruct & *ioFFDisutility*)

Actor Constructor.

Definition at line 33 of file FFDisutilityParserHelper.cpp.

22.128.3 Member Function Documentation

22.128.3.1 void AIRINV::FFDisutilityParserHelper::storeCurveKey::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 38 of file FFDisutilityParserHelper.cpp.

References AIRINV::FFDisutilityParserHelper::ParserSemanticAction::_ffDisutility, and AIRINV::FFDisutilityStruct::-_key.

22.128.4 Member Data Documentation

22.128.4.1 FFDisutilityStruct& AIRINV::FFDisutilityParserHelper::ParserSemanticAction::_ffDisutility [inherited]

Actor Context.

Definition at line 33 of file FFDisutilityParserHelper.hpp.

Referenced by operator()(), AIRINV::FFDisutilityParserHelper::storeDTD::operator()(), AIRINV::FFDisutilityParser-Helper::storeFFDisutilityValue::operator()(), and AIRINV::FFDisutilityParserHelper::doEndCurve::operator()().

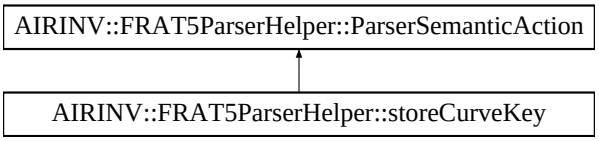
The documentation for this struct was generated from the following files:

- airinv/command/FFDisutilityParserHelper.hpp
- airinv/command/FFDisutilityParserHelper.cpp

22.129 AIRINV::FRAT5ParserHelper::storeCurveKey Struct Reference

`#include <airinv/command/FRAT5ParserHelper.hpp>`

Inheritance diagram for AIRINV::FRAT5ParserHelper::storeCurveKey:



Public Member Functions

- storeCurveKey (FRAT5Struct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FRAT5Struct & _frat5

22.129.1 Detailed Description

Store the parsed curve key.

Definition at line 37 of file FRAT5ParserHelper.hpp.

22.129.2 Constructor & Destructor Documentation

22.129.2.1 AIRINV::FRAT5ParserHelper::storeCurveKey::storeCurveKey (FRAT5Struct & *ioFRAT5*)

Actor Constructor.

Definition at line 33 of file FRAT5ParserHelper.cpp.

22.129.3 Member Function Documentation

22.129.3.1 void AIRINV::FRAT5ParserHelper::storeCurveKey::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 38 of file FRAT5ParserHelper.cpp.

References AIRINV::FRAT5ParserHelper::ParserSemanticAction::_frat5, and AIRINV::FRAT5Struct::_key.

22.129.4 Member Data Documentation

22.129.4.1 FRAT5Struct& AIRINV::FRAT5ParserHelper::ParserSemanticAction::_frat5 [inherited]

Actor Context.

Definition at line 33 of file FRAT5ParserHelper.hpp.

Referenced by operator()(), AIRINV::FRAT5ParserHelper::storeDTD::operator()(), AIRINV::FRAT5ParserHelper::storeFRAT5Value::operator()(), and AIRINV::FRAT5ParserHelper::doEndCurve::operator()().

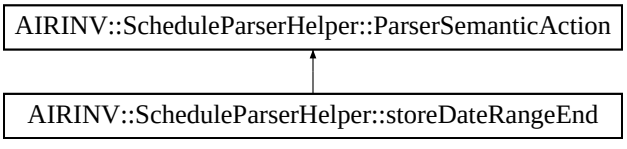
The documentation for this struct was generated from the following files:

- airinv/command/FRAT5ParserHelper.hpp
- airinv/command/FRAT5ParserHelper.cpp

22.130 AIRINV::ScheduleParserHelper::storeDateRangeEnd Struct Reference

`#include <airinv/command/ScheduleParserHelper.hpp>`

Inheritance diagram for AIRINV::ScheduleParserHelper::storeDateRangeEnd:



Public Member Functions

- storeDateRangeEnd (FlightPeriodStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

22.130.1 Detailed Description

Store the end of the date range.

Definition at line 61 of file ScheduleParserHelper.hpp.

22.130.2 Constructor & Destructor Documentation

22.130.2.1 AIRINV::ScheduleParserHelper::storeDateRangeEnd::storeDateRangeEnd (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 77 of file ScheduleParserHelper.cpp.

22.130.3 Member Function Documentation

22.130.3.1 void AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 82 of file ScheduleParserHelper.cpp.

References AIRINV::LegStruct::_airlineCode, AIRINV::FlightPeriodStruct::_airlineCode, AIRINV::FlightPeriodStruct::_dateRange, AIRINV::FlightPeriodStruct::_dateRangeEnd, AIRINV::FlightPeriodStruct::_dateRangeStart, AIRINV::LegStruct::_flightNumber, AIRINV::FlightPeriodStruct::_flightNumber, AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod, AIRINV::FlightPeriodStruct::_itLeg, AIRINV::FlightPeriodStruct::_itSeconds, and AIRINV::FlightPeriodStruct::getDate().

22.130.4 Member Data Documentation

22.130.4.1 FlightPeriodStruct& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRINV::ScheduleParserHelper::storeAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeStart::operator()(), operator()(), AIRINV::ScheduleParserHelper::storeDow::operator()(), AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeBoardingTime::operator()(), AIRINV::ScheduleParserHelper::storeOffTime::operator()(), AIRINV::ScheduleParserHelper::storeElapsedTime::operator()(), AIRINV::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeCapacity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeClasses::operator()(), AIRINV::ScheduleParserHelper::storeFamilyCode::operator()(), AIRINV::ScheduleParserHelper::storeFRAT5CurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFFDisutilityCurveKey::operator()(),

AIRINV::ScheduleParserHelper::storeFClasses::operator()(), and AIRINV::ScheduleParserHelper::doEndFlight-::operator()().

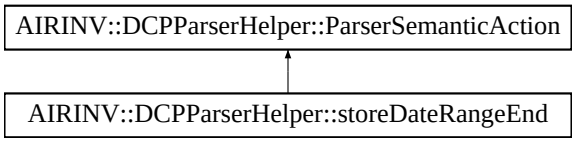
The documentation for this struct was generated from the following files:

- airinv/command/ScheduleParserHelper.hpp
- airinv/command/ScheduleParserHelper.cpp

22.131 AIRINV::DCPParserHelper::storeDateRangeEnd Struct Reference

`#include <airinv/command/vault/DCPParserHelper.hpp>`

Inheritance diagram for AIRINV::DCPParserHelper::storeDateRangeEnd:



Public Member Functions

- storeDateRangeEnd (DCPRuleStruct &)
- void operator() (boost::spirit::qi::unused_type, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- DCPRuleStruct & _DCPRule

22.131.1 Detailed Description

Store the parsed end of the date range.

Definition at line 78 of file DCPParserHelper.hpp.

22.131.2 Constructor & Destructor Documentation

22.131.2.1 AIRINV::DCPParserHelper::storeDateRangeEnd::storeDateRangeEnd (DCPRuleStruct & *ioDCPRule*)

Actor Constructor.

Definition at line 101 of file DCPParserHelper.cpp.

22.131.3 Member Function Documentation

22.131.3.1 void AIRINV::DCPParserHelper::storeDateRangeEnd::operator() (boost::spirit::qi::unused_type , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 106 of file DCPParserHelper.cpp.

References AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule.

22.131.4 Member Data Documentation

22.131.4.1 DCPRuleStruct& AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule [inherited]

Actor Context.

Definition at line 34 of file DCPParserHelper.hpp.

Referenced by AIRINV::DCPParserHelper::storeDCPId::operator()(), AIRINV::DCPParserHelper::storeOrigin::operator()(), AIRINV::DCPParserHelper::storeDestination::operator()(), AIRINV::DCPParserHelper::storeDateRangeStart::operator()(), operator()(), AIRINV::DCPParserHelper::storeStartRangeTime::operator()(), AIRINV::DCPParserHelper::storeEndRangeTime::operator()(), AIRINV::DCPParserHelper::storePOS::operator()(), AIRINV::DCPParserHelper::storeCabinCode::operator()(), AIRINV::DCPParserHelper::storeChannel::operator()(), AIRINV::DCPParserHelper::storeAdvancePurchase::operator()(), AIRINV::DCPParserHelper::storeSaturdayStay::operator()(), AIRINV::DCPParserHelper::storeChangeFees::operator()(), AIRINV::DCPParserHelper::storeNonRefundable::operator()(), AIRINV::DCPParserHelper::storeMinimumStay::operator()(), AIRINV::DCPParserHelper::storeDCP::operator()(), AIRINV::DCPParserHelper::storeAirlineCode::operator()(), AIRINV::DCPParserHelper::storeClass::operator()(), and AIRINV::DCPParserHelper::doEndDCP::operator()().

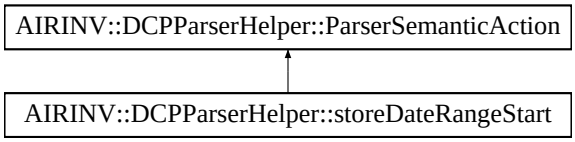
The documentation for this struct was generated from the following files:

- airinv/command/vault/DCPParserHelper.hpp
- airinv/command/vault/DCPParserHelper.cpp

22.132 AIRINV::DCPParserHelper::storeDateRangeStart Struct Reference

`#include <airinv/command/vault/DCPParserHelper.hpp>`

Inheritance diagram for AIRINV::DCPParserHelper::storeDateRangeStart:



Public Member Functions

- storeDateRangeStart (DCPRuleStruct &)
- void operator() (boost::spirit::qi::unused_type, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- DCPRuleStruct & _DCPRule

22.132.1 Detailed Description

Store the parsed start of the date range.

Definition at line 68 of file DCPParserHelper.hpp.

22.132.2 Constructor & Destructor Documentation

22.132.2.1 AIRINV::DCPParserHelper::storeDateRangeStart::storeDateRangeStart (DCPRuleStruct & *ioDCPRule*)

Actor Constructor.

Definition at line 86 of file DCPParserHelper.cpp.

22.132.3 Member Function Documentation

22.132.3.1 void AIRINV::DCPParserHelper::storeDateRangeStart::operator() (boost::spirit::qi::unused_type , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 91 of file DCPParserHelper.cpp.

References AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule.

22.132.4 Member Data Documentation

22.132.4.1 DCPRuleStruct& AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule [inherited]

Actor Context.

Definition at line 34 of file DCPParserHelper.hpp.

Referenced by AIRINV::DCPParserHelper::storeDCPId::operator()(), AIRINV::DCPParserHelper::storeOrigin::operator()(), AIRINV::DCPParserHelper::storeDestination::operator()(), operator()(), AIRINV::DCPParserHelper::storeDateRangeEnd::operator()(), AIRINV::DCPParserHelper::storeStartRangeTime::operator()(), AIRINV::DCPParserHelper::storeEndRangeTime::operator()(), AIRINV::DCPParserHelper::storePOS::operator()(), AIRINV::DCPParserHelper::storeCabinCode::operator()(), AIRINV::DCPParserHelper::storeChannel::operator()(), AIRINV::DCPParserHelper::storeAdvancePurchase::operator()(), AIRINV::DCPParserHelper::storeSaturdayStay::operator()(), AIRINV::DCPParserHelper::storeChangeFees::operator()(), AIRINV::DCPParserHelper::storeNonRefundable::operator()(), AIRINV::DCPParserHelper::storeMinimumStay::operator()(), AIRINV::DCPParserHelper::storeDCP::operator()(), AIRINV::DCPParserHelper::storeAirlineCode::operator()(), AIRINV::DCPParserHelper::storeClass::operator()(), and AIRINV::DCPParserHelper::doEndDCP::operator()().

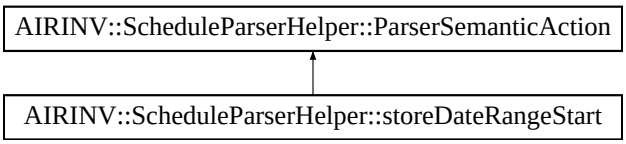
The documentation for this struct was generated from the following files:

- airinv/command/vault/DCPParserHelper.hpp
- airinv/command/vault/DCPParserHelper.cpp

22.133 AIRINV::ScheduleParserHelper::storeDateRangeStart Struct Reference

`#include <airinv/command/ScheduleParserHelper.hpp>`

Inheritance diagram for AIRINV::ScheduleParserHelper::storeDateRangeStart:



Public Member Functions

- storeDateRangeStart (FlightPeriodStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

22.133.1 Detailed Description

Store the start of the date range.

Definition at line 53 of file ScheduleParserHelper.hpp.

22.133.2 Constructor & Destructor Documentation

22.133.2.1 AIRINV::ScheduleParserHelper::storeDateRangeStart::storeDateRangeStart (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 62 of file ScheduleParserHelper.cpp.

22.133.3 Member Function Documentation

22.133.3.1 void AIRINV::ScheduleParserHelper::storeDateRangeStart::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 67 of file ScheduleParserHelper.cpp.

References AIRINV::FlightPeriodStruct::_dateRangeStart, AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod, AIRINV::FlightPeriodStruct::_itSeconds, and AIRINV::FlightPeriodStruct::getDate().

22.133.4 Member Data Documentation

22.133.4.1 FlightPeriodStruct& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRINV::ScheduleParserHelper::storeAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeFlightNumber::operator()(), operator()(), AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRINV::ScheduleParserHelper::storeDow::operator()(), AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeBoardingTime::operator()(), AIRINV::ScheduleParserHelper::storeOffTime::operator()(), AIRINV::ScheduleParserHelper::storeElapsedTime::operator()(), AIRINV::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeCapacity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeClasses::operator()(), AIRINV::ScheduleParserHelper::storeFamilyCode::operator()(), AIRINV::ScheduleParserHelper::storeFRAT5CurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFFDisutilityCurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFClasses::operator()(), and AIRINV::ScheduleParserHelper::doEndFlight::operator()().

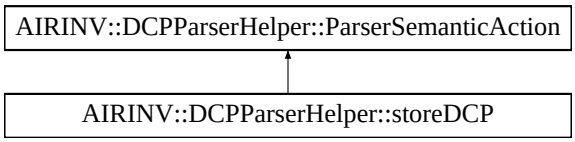
The documentation for this struct was generated from the following files:

- airinv/command/ScheduleParserHelper.hpp
- airinv/command/ScheduleParserHelper.cpp

22.134 AIRINV::DCPParserHelper::storeDCP Struct Reference

```
#include <airinv/command/vault/DCPParserHelper.hpp>
```

Inheritance diagram for AIRINV::DCPParserHelper::storeDCP:



Public Member Functions

- storeDCP (DCPRuleStruct &)
- void operator() (double, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- DCPRuleStruct & _DCPRule

22.134.1 Detailed Description

Store the parsed DCP value.

Definition at line 188 of file DCPParserHelper.hpp.

22.134.2 Constructor & Destructor Documentation

22.134.2.1 AIRINV::DCPParserHelper::storeDCP::storeDCP (DCPRuleStruct & *ioDCPRule*)

Actor Constructor.

Definition at line 314 of file DCPParserHelper.cpp.

22.134.3 Member Function Documentation

22.134.3.1 void AIRINV::DCPParserHelper::storeDCP::operator() (double *iDCP*, boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 319 of file DCPParserHelper.cpp.

References AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule.

22.134.4 Member Data Documentation

22.134.4.1 DCPRuleStruct& AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule [inherited]

Actor Context.

Definition at line 34 of file DCPParserHelper.hpp.

Referenced by AIRINV::DCPParserHelper::storeDCPId::operator()(), AIRINV::DCPParserHelper::storeOrigin::operator()(), AIRINV::DCPParserHelper::storeDestination::operator()(), AIRINV::DCPParserHelper::storeDateRangeStart::operator()(), AIRINV::DCPParserHelper::storeDateRangeEnd::operator()(), AIRINV::DCPParserHelper::storeStartRangeTime::operator()(), AIRINV::DCPParserHelper::storeEndRangeTime::operator()(), AIRINV::DCPParserHelper::storePOS::operator()(), AIRINV::DCPParserHelper::storeCabinCode::operator()(), AIRINV::DCPParserHelper::storeChannel::operator()(), AIRINV::DCPParserHelper::storeAdvancePurchase::operator()(),

AIRINV::DCPParserHelper::storeSaturdayStay::operator()(), AIRINV::DCPParserHelper::storeChangeFees::operator()(), AIRINV::DCPParserHelper::storeNonRefundable::operator()(), AIRINV::DCPParserHelper::storeMinimumStay::operator()(), operator()(), AIRINV::DCPParserHelper::storeAirlineCode::operator()(), AIRINV::DCPParserHelper::storeClass::operator()(), and AIRINV::DCPParserHelper::doEndDCP::operator()().

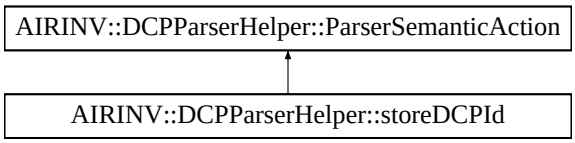
The documentation for this struct was generated from the following files:

- airinv/command/vault/DCPParserHelper.hpp
- airinv/command/vault/DCPParserHelper.cpp

22.135 AIRINV::DCPParserHelper::storeDCPId Struct Reference

`#include <airinv/command/vault/DCPParserHelper.hpp>`

Inheritance diagram for AIRINV::DCPParserHelper::storeDCPId:



Public Member Functions

- storeDCPId (DCPRuleStruct &)
- void operator() (unsigned int, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- DCPRuleStruct & _DCPRule

22.135.1 Detailed Description

Store the parsed DCP Id.

Definition at line 38 of file DCPParserHelper.hpp.

22.135.2 Constructor & Destructor Documentation

22.135.2.1 AIRINV::DCPParserHelper::storeDCPId::storeDCPId (DCPRuleStruct & *ioDCPRule*)

Actor Constructor.

Definition at line 30 of file DCPParserHelper.cpp.

22.135.3 Member Function Documentation

22.135.3.1 void AIRINV::DCPParserHelper::storeDCPId::operator() (unsigned int *iDCPId,* boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 35 of file DCPParserHelper.cpp.

References AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule.

22.135.4 Member Data Documentation

22.135.4.1 DCPRuleStruct& AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule [inherited]

Actor Context.

Definition at line 34 of file DCPParserHelper.hpp.

Referenced by operator()(), AIRINV::DCPParserHelper::storeOrigin::operator()(), AIRINV::DCPParserHelper::storeDestination::operator()(), AIRINV::DCPParserHelper::storeDateRangeStart::operator()(), AIRINV::DCPParserHelper::storeDateRangeEnd::operator()(), AIRINV::DCPParserHelper::storeStartRangeTime::operator()(), AIRINV::DCPParserHelper::storeEndRangeTime::operator()(), AIRINV::DCPParserHelper::storePOS::operator()(), AIRINV::DCPParserHelper::storeCabinCode::operator()(), AIRINV::DCPParserHelper::storeChannel::operator()(), AIRINV::DCPParserHelper::storeAdvancePurchase::operator()(), AIRINV::DCPParserHelper::storeSaturdayStay::operator()(), AIRINV::DCPParserHelper::storeChangeFees::operator()(), AIRINV::DCPParserHelper::storeNonRefundable::operator()(), AIRINV::DCPParserHelper::storeMinimumStay::operator()(), AIRINV::DCPParserHelper::storeDCP::operator()(), AIRINV::DCPParserHelper::storeAirlineCode::operator()(), AIRINV::DCPParserHelper::storeClass::operator()(), and AIRINV::DCPParserHelper::doEndDCP::operator()().

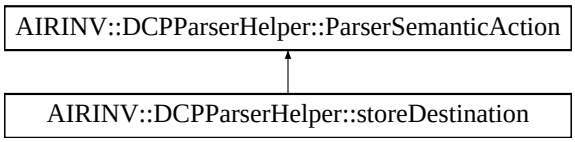
The documentation for this struct was generated from the following files:

- airinv/command/vault/DCPParserHelper.hpp
- airinv/command/vault/DCPParserHelper.cpp

22.136 AIRINV::DCPParserHelper::storeDestination Struct Reference

`#include <airinv/command/vault/DCPParserHelper.hpp>`

Inheritance diagram for AIRINV::DCPParserHelper::storeDestination:



Public Member Functions

- storeDestination (DCPRuleStruct &)
- void operator() (std::vector< char >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- DCPRuleStruct & _DCPRule

22.136.1 Detailed Description

Store the parsed destination.

Definition at line 58 of file DCPParserHelper.hpp.

22.136.2 Constructor & Destructor Documentation

22.136.2.1 AIRINV::DCPParserHelper::storeDestination::storeDestination (DCPRuleStruct & *ioDCPRule*)

Actor Constructor.

Definition at line 70 of file DCPParserHelper.cpp.

22.136.3 Member Function Documentation

22.136.3.1 void AIRINV::DCPParserHelper::storeDestination::operator() (std::vector< char > *iChar,* boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 75 of file DCPParserHelper.cpp.

References AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule.

22.136.4 Member Data Documentation

22.136.4.1 DCPRuleStruct& AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule [inherited]

Actor Context.

Definition at line 34 of file DCPParserHelper.hpp.

Referenced by AIRINV::DCPParserHelper::storeDCPId::operator()(), AIRINV::DCPParserHelper::storeOrigin::operator()(), operator()(), AIRINV::DCPParserHelper::storeDateRangeStart::operator()(), AIRINV::DCPParserHelper::storeDateRangeEnd::operator()(), AIRINV::DCPParserHelper::storeStartRangeTime::operator()(), AIRINV::DCPParserHelper::storeEndRangeTime::operator()(), AIRINV::DCPParserHelper::storePOS::operator()(), AIRINV::DCPParserHelper::storeCabinCode::operator()(), AIRINV::DCPParserHelper::storeChannel::operator()(), AIRINV::DCPParserHelper::storeAdvancePurchase::operator()(), AIRINV::DCPParserHelper::storeSaturdayStay::operator()(), AIRINV::DCPParserHelper::storeChangeFees::operator()(), AIRINV::DCPParserHelper::storeNonRefundable::operator()(), AIRINV::DCPParserHelper::storeMinimumStay::operator()(), AIRINV::DCPParserHelper::storeDCP::operator()(), AIRINV::DCPParserHelper::storeAirlineCode::operator()(), AIRINV::DCPParserHelper::storeClass::operator()(), and AIRINV::DCPParserHelper::doEndDCP::operator()().

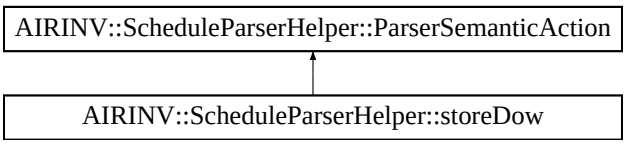
The documentation for this struct was generated from the following files:

- airinv/command/vault/DCPParserHelper.hpp
- airinv/command/vault/DCPParserHelper.cpp

22.137 AIRINV::ScheduleParserHelper::storeDow Struct Reference

`#include <airinv/command/ScheduleParserHelper.hpp>`

Inheritance diagram for AIRINV::ScheduleParserHelper::storeDow:



Public Member Functions

- storeDow (FlightPeriodStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

22.137.1 Detailed Description

Store the DOW (day of the Week).

Definition at line 69 of file ScheduleParserHelper.hpp.

22.137.2 Constructor & Destructor Documentation

22.137.2.1 AIRINV::ScheduleParserHelper::storeDow::storeDow (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 105 of file ScheduleParserHelper.cpp.

22.137.3 Member Function Documentation

22.137.3.1 void AIRINV::ScheduleParserHelper::storeDow::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 110 of file ScheduleParserHelper.cpp.

References AIRINV::FlightPeriodStruct::_dow, and AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod.

22.137.4 Member Data Documentation

22.137.4.1 FlightPeriodStruct& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRINV::ScheduleParserHelper::storeAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator()(), operator()(), AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeBoardingTime::operator()(), AIRINV::ScheduleParserHelper::storeOffTime::operator()(), AIRINV::ScheduleParserHelper::storeElapsedTime::operator()(), AIRINV::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeCapacity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeClasses::operator()(), AIRINV::ScheduleParserHelper::storeFamilyCode::operator()(), AIRINV::ScheduleParserHelper::storeFRAT5CurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFFDisutilityCurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFClasses::operator()(), and AIRINV::ScheduleParserHelper::doEndFlight::operator()().

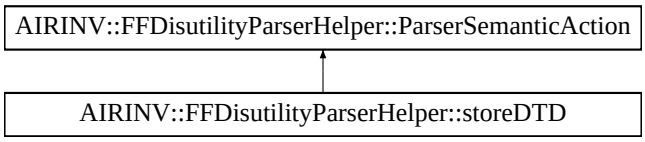
The documentation for this struct was generated from the following files:

- airinv/command/ScheduleParserHelper.hpp
- airinv/command/ScheduleParserHelper.cpp

22.138 AIRINV::FFDisutilityParserHelper::storeDTD Struct Reference

`#include <airinv/command/FFDisutilityParserHelper.hpp>`

Inheritance diagram for AIRINV::FFDisutilityParserHelper::storeDTD:



Public Member Functions

- storeDTD (FFDisutilityStruct &)
- void operator() (int iDTD) const

Public Attributes

- FFDisutilityStruct & _ffDisutility

22.138.1 Detailed Description

Store the DTD.

Definition at line 45 of file FFDisutilityParserHelper.hpp.

22.138.2 Constructor & Destructor Documentation

22.138.2.1 AIRINV::FFDisutilityParserHelper::storeDTD::storeDTD (FFDisutilityStruct & *ioFFDisutility*)

Actor Constructor.

Definition at line 45 of file FFDisutilityParserHelper.cpp.

22.138.3 Member Function Documentation

22.138.3.1 void AIRINV::FFDisutilityParserHelper::storeDTD::operator() (int *iDTD*) const

Actor Function (functor).

Definition at line 50 of file FFDisutilityParserHelper.cpp.

References AIRINV::FFDisutilityStruct::_dtd, and AIRINV::FFDisutilityParserHelper::ParserSemanticAction::_ffDisutility.

22.138.4 Member Data Documentation

22.138.4.1 FFDisutilityStruct& AIRINV::FFDisutilityParserHelper::ParserSemanticAction::_ffDisutility [inherited]

Actor Context.

Definition at line 33 of file FFDisutilityParserHelper.hpp.

Referenced by AIRINV::FFDisutilityParserHelper::storeCurveKey::operator()(), operator()(), AIRINV::FFDisutilityParserHelper::storeFFDisutilityValue::operator()(), and AIRINV::FFDisutilityParserHelper::doEndCurve::operator()().

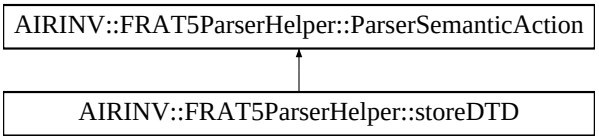
The documentation for this struct was generated from the following files:

- airinv/command/FFDisutilityParserHelper.hpp
- airinv/command/FFDisutilityParserHelper.cpp

22.139 AIRINV::FRAT5ParserHelper::storeDTD Struct Reference

`#include <airinv/command/FRAT5ParserHelper.hpp>`

Inheritance diagram for AIRINV::FRAT5ParserHelper::storeDTD:



Public Member Functions

- storeDTD (FRAT5Struct &)
- void operator() (int iDTD) const

Public Attributes

- FRAT5Struct & _frat5

22.139.1 Detailed Description

Store the DTD.

Definition at line 45 of file FRAT5ParserHelper.hpp.

22.139.2 Constructor & Destructor Documentation

22.139.2.1 AIRINV::FRAT5ParserHelper::storeDTD::storeDTD (FRAT5Struct & *ioFRAT5*)

Actor Constructor.

Definition at line 46 of file FRAT5ParserHelper.cpp.

22.139.3 Member Function Documentation

22.139.3.1 void AIRINV::FRAT5ParserHelper::storeDTD::operator() (int *iDTD*) const

Actor Function (functor).

Definition at line 51 of file FRAT5ParserHelper.cpp.

References AIRINV::FRAT5Struct::_dtd, and AIRINV::FRAT5ParserHelper::ParserSemanticAction::_frat5.

22.139.4 Member Data Documentation

22.139.4.1 FRAT5Struct& AIRINV::FRAT5ParserHelper::ParserSemanticAction::_frat5 [inherited]

Actor Context.

Definition at line 33 of file FRAT5ParserHelper.hpp.

Referenced by AIRINV::FRAT5ParserHelper::storeCurveKey::operator()(), operator()(), AIRINV::FRAT5ParserHelper::storeFRAT5Value::operator()(), and AIRINV::FRAT5ParserHelper::doEndCurve::operator()().

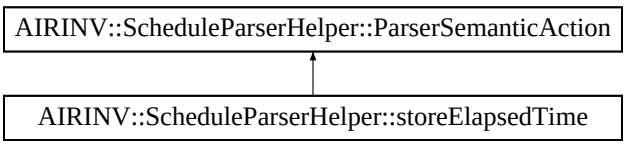
The documentation for this struct was generated from the following files:

- airinv/command/FRAT5ParserHelper.hpp
- airinv/command/FRAT5ParserHelper.cpp

22.140 AIRINV::ScheduleParserHelper::storeElapsedTime Struct Reference

`#include <airinv/command/ScheduleParserHelper.hpp>`

Inheritance diagram for AIRINV::ScheduleParserHelper::storeElapsedTime:



Public Member Functions

- storeElapsedTime (FlightPeriodStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

22.140.1 Detailed Description

Store the elapsed time.

Definition at line 125 of file ScheduleParserHelper.hpp.

22.140.2 Constructor & Destructor Documentation

22.140.2.1 AIRINV::ScheduleParserHelper::storeElapsedTime::storeElapsedTime (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 229 of file ScheduleParserHelper.cpp.

22.140.3 Member Function Documentation

22.140.3.1 void AIRINV::ScheduleParserHelper::storeElapsedTime::operator() (iterator_t *iStr*, iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 234 of file ScheduleParserHelper.cpp.

References AIRINV::FlightPeriodStruct::_dateOffset, AIRINV::LegStruct::_elapsed, AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod, AIRINV::FlightPeriodStruct::_itLeg, AIRINV::FlightPeriodStruct::_itSeconds, AIRINV::LegStruct::_offDateOffset, and AIRINV::FlightPeriodStruct::getTime().

22.140.4 Member Data Documentation

22.140.4.1 FlightPeriodStruct& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRINV::ScheduleParserHelper::storeAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRINV::ScheduleParserHelper::storeDow::operator()(), AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeBoardingTime::operator()(), AIRINV::ScheduleParserHelper::storeOffTime::operator()(), operator()(), AIRINV::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeCapacity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeClasses::operator()(), AIRINV::ScheduleParserHelper::storeFamilyCode::operator()(), AIRINV::ScheduleParserHelper::storeFRAT5CurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFFDisutilityCurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFClasses::operator()(), and AIRINV::ScheduleParserHelper::doEndFlight::operator()().

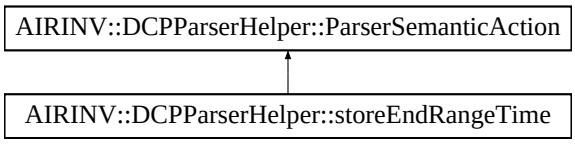
The documentation for this struct was generated from the following files:

- airinv/command/ScheduleParserHelper.hpp
- airinv/command/ScheduleParserHelper.cpp

22.141 AIRINV::DCPParserHelper::storeEndRangeTime Struct Reference

`#include <airinv/command/vault/DCPParserHelper.hpp>`

Inheritance diagram for AIRINV::DCPParserHelper::storeEndRangeTime:



Public Member Functions

- storeEndRangeTime (DCPRuleStruct &)
- void operator() (boost::spirit::qi::unused_type, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- DCPRuleStruct & _DCPRule

22.141.1 Detailed Description

Store the parsed end start range time.

Definition at line 98 of file DCPParserHelper.hpp.

22.141.2 Constructor & Destructor Documentation

22.141.2.1 AIRINV::DCPParserHelper::storeEndRangeTime::storeEndRangeTime (DCPRuleStruct & *ioDCPRule*)

Actor Constructor.

Definition at line 133 of file DCPParserHelper.cpp.

22.141.3 Member Function Documentation

22.141.3.1 void AIRINV::DCPParserHelper::storeEndRangeTime::operator() (boost::spirit::qi::unused_type , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 138 of file DCPParserHelper.cpp.

References AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule.

22.141.4 Member Data Documentation

22.141.4.1 DCPRuleStruct& AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule [inherited]

Actor Context.

Definition at line 34 of file DCPParserHelper.hpp.

Referenced by AIRINV::DCPParserHelper::storeDCPId::operator()(), AIRINV::DCPParserHelper::storeOrigin::operator()(), AIRINV::DCPParserHelper::storeDestination::operator()(), AIRINV::DCPParserHelper::storeDateRangeStart::operator()(), AIRINV::DCPParserHelper::storeDateRangeEnd::operator()(), AIRINV::DCPParserHelper::storeStartRangeTime::operator()(), operator()(), AIRINV::DCPParserHelper::storePOS::operator()(), AIRINV::DCPParserHelper::storeCabinCode::operator()(), AIRINV::DCPParserHelper::storeChannel::operator()(), AIRINV::DCPParserHelper::storeAdvancePurchase::operator()(), AIRINV::DCPParserHelper::storeSaturdayStay::operator()(), AIRINV::DCPParserHelper::storeChangeFees::operator()(), AIRINV::DCPParserHelper::storeNonRefundable::operator()(), AIRINV::DCPParserHelper::storeMinimumStay::operator()(), AIRINV::DCPParserHelper::storeDCP::operator()(), AIRINV::DCPParserHelper::storeAirlineCode::operator()(), AIRINV::DCPParserHelper::storeClass::operator()(), and AIRINV::DCPParserHelper::doEndDCP::operator()().

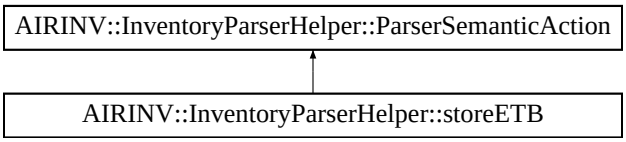
The documentation for this struct was generated from the following files:

- airinv/command/vault/DCPParserHelper.hpp
- airinv/command/vault/DCPParserHelper.cpp

22.142 AIRINV::InventoryParserHelper::storeETB Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeETB:



Public Member Functions

- storeETB (FlightDateStruct &)
- void operator() (double iReal) const

Public Attributes

- FlightDateStruct & _flightDate

22.142.1 Detailed Description

Store the parsed Expected To Board (ETB) number.

Definition at line 213 of file InventoryParserHelper.hpp.

22.142.2 Constructor & Destructor Documentation

22.142.2.1 AIRINV::InventoryParserHelper::storeETB::storeETB (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 361 of file InventoryParserHelper.cpp.

22.142.3 Member Function Documentation

22.142.3.1 void AIRINV::InventoryParserHelper::storeETB::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 366 of file InventoryParserHelper.cpp.

References AIRINV::LegCabinStruct::_etb, AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, and AIRINV::FlightDateStruct::_itLegCabin.

22.142.4 Member Data Documentation

22.142.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV-

::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

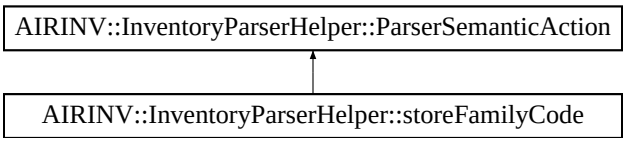
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.143 AIRINV::InventoryParserHelper::storeFamilyCode Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeFamilyCode:



Public Member Functions

- storeFamilyCode (FlightDateStruct &)
- void operator() (int iCode) const

Public Attributes

- FlightDateStruct & _flightDate

22.143.1 Detailed Description

Store the parsed family code.

Definition at line 425 of file InventoryParserHelper.hpp.

22.143.2 Constructor & Destructor Documentation

22.143.2.1 AIRINV::InventoryParserHelper::storeFamilyCode::storeFamilyCode (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 737 of file InventoryParserHelper.cpp.

22.143.3 Member Function Documentation

22.143.3.1 void AIRINV::InventoryParserHelper::storeFamilyCode::operator() (int *iCode*) const

Actor Function (functor).

Definition at line 742 of file InventoryParserHelper.cpp.

References AIRINV::FareFamilyStruct::_familyCode, AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::SegmentCabinStruct::_itFareFamily, and AIRINV::FlightDateStruct::_itSegmentCabin.

22.143.4 Member Data Documentation

22.143.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

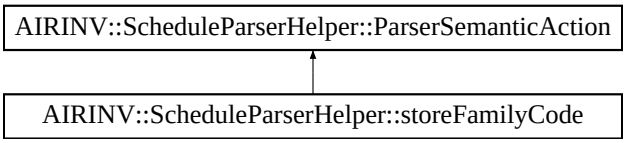
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.144 AIRINV::ScheduleParserHelper::storeFamilyCode Struct Reference

`#include <airinv/command/ScheduleParserHelper.hpp>`

Inheritance diagram for AIRINV::ScheduleParserHelper::storeFamilyCode:



Public Member Functions

- storeFamilyCode (FlightPeriodStruct &)
- void operator() (int iCode) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

22.144.1 Detailed Description

Store the parsed family code.

Definition at line 192 of file ScheduleParserHelper.hpp.

22.144.2 Constructor & Destructor Documentation

22.144.2.1 AIRINV::ScheduleParserHelper::storeFamilyCode::storeFamilyCode (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 369 of file ScheduleParserHelper.cpp.

22.144.3 Member Function Documentation

22.144.3.1 void AIRINV::ScheduleParserHelper::storeFamilyCode::operator() (int *iCode*) const

Actor Function (functor).

Definition at line 374 of file ScheduleParserHelper.cpp.

References AIRINV::FareFamilyStruct::_familyCode, AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod, AIRINV::SegmentCabinStruct::_itFareFamily, and AIRINV::FlightPeriodStruct::_itSegmentCabin.

22.144.4 Member Data Documentation

22.144.4.1 FlightPeriodStruct& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRINV::ScheduleParserHelper::storeAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRINV::ScheduleParserHelper::storeDow::operator()(), AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeBoardingTime::operator()(), AIRINV::ScheduleParserHelper::storeOffTime::operator()(), AIRINV::ScheduleParserHelper::storeElapsedTime::operator()(), AIRINV::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeCapacity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeClasses::operator()(), operator()(), AIRINV::ScheduleParserHelper::storeFRAT5CurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFFDisutilityCurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFClasses::operator()(), and AIRINV::ScheduleParserHelper::doEndFlight::operator()().

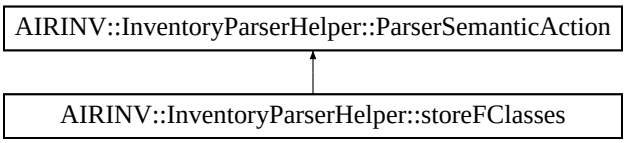
The documentation for this struct was generated from the following files:

- airinv/command/ScheduleParserHelper.hpp
- airinv/command/ScheduleParserHelper.cpp

22.145 AIRINV::InventoryParserHelper::storeFClasses Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeFClasses:



Public Member Functions

- storeFClasses (FlightDateStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightDateStruct & _flightDate

22.145.1 Detailed Description

Store the parsed list of class codes (for families).

Definition at line 433 of file InventoryParserHelper.hpp.

22.145.2 Constructor & Destructor Documentation

22.145.2.1 AIRINV::InventoryParserHelper::storeFClasses::storeFClasses (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 749 of file InventoryParserHelper.cpp.

22.145.3 Member Function Documentation

22.145.3.1 void AIRINV::InventoryParserHelper::storeFClasses::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 754 of file InventoryParserHelper.cpp.

References AIRINV::SegmentStruct::_cabinList, AIRINV::BookingClassStruct::_classCode, AIRINV::FareFamilyStruct::_classes, AIRINV::FareFamilyStruct::_classList, AIRINV::SegmentCabinStruct::_fareFamilies, AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::FlightDateStruct::_itBookingClass, AIRINV::SegmentCabinStruct::_itFareFamily, AIRINV::FlightDateStruct::_itSegment, and AIRINV::FlightDateStruct::_itSegmentCabin.

22.145.4 Member Data Documentation

22.145.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

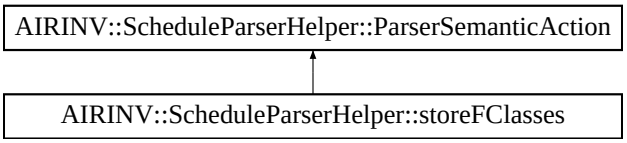
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.146 AIRINV::ScheduleParserHelper::storeFClasses Struct Reference

`#include <airinv/command/ScheduleParserHelper.hpp>`

Inheritance diagram for AIRINV::ScheduleParserHelper::storeFClasses:



Public Member Functions

- storeFClasses (FlightPeriodStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

22.146.1 Detailed Description

Store the parsed list of class codes (for families).

Definition at line 216 of file ScheduleParserHelper.hpp.

22.146.2 Constructor & Destructor Documentation

22.146.2.1 AIRINV::ScheduleParserHelper::storeFClasses::storeFClasses (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 409 of file ScheduleParserHelper.cpp.

22.146.3 Member Function Documentation

22.146.3.1 void AIRINV::ScheduleParserHelper::storeFClasses::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 414 of file ScheduleParserHelper.cpp.

References AIRINV::FlightPeriodStruct::_areSegmentDefinitionsSpecific, AIRINV::FareFamilyStruct::_familyCode, AIRINV::FareFamilyStruct::_ffDisutilityCurveKey, AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod, AIRINV::FareFamilyStruct::_frat5CurveKey, AIRINV::SegmentCabinStruct::_itFareFamily, AIRINV::FlightPeriodStruct::_itSegment, AIRINV::FlightPeriodStruct::_itSegmentCabin, and AIRINV::FlightPeriodStruct::addFareFamily().

22.146.4 Member Data Documentation

22.146.4.1 FlightPeriodStruct& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRINV::ScheduleParserHelper::storeAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRINV::ScheduleParserHelper::storeDow::operator()(), AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeBoardingTime::operator()(), AIRINV::ScheduleParserHelper::storeOffTime::operator()(), AIRINV::ScheduleParserHelper::storeElapsedTime::operator()(), AIRINV::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeCapacity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeClasses::operator()(), AIRINV::ScheduleParserHelper::storeFamilyCode::operator()(),

AIRINV::ScheduleParserHelper::storeFRAT5CurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFFDisutilityCurveKey::operator()(), operator()(), and AIRINV::ScheduleParserHelper::doEndFlight::operator()().

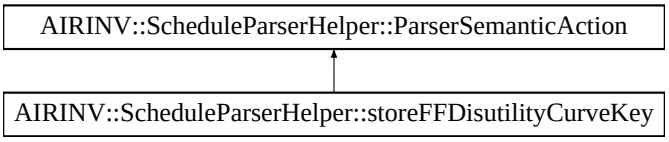
The documentation for this struct was generated from the following files:

- airinv/command/ScheduleParserHelper.hpp
- airinv/command/ScheduleParserHelper.cpp

22.147 AIRINV::ScheduleParserHelper::storeFFDisutilityCurveKey Struct Reference

`#include <airinv/command/ScheduleParserHelper.hpp>`

Inheritance diagram for AIRINV::ScheduleParserHelper::storeFFDisutilityCurveKey:



Public Member Functions

- storeFFDisutilityCurveKey (FlightPeriodStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

22.147.1 Detailed Description

Store the FFDisutility curve key.

Definition at line 208 of file ScheduleParserHelper.hpp.

22.147.2 Constructor & Destructor Documentation

22.147.2.1 AIRINV::ScheduleParserHelper::storeFFDisutilityCurveKey::storeFFDisutilityCurveKey (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 396 of file ScheduleParserHelper.cpp.

22.147.3 Member Function Documentation

22.147.3.1 void AIRINV::ScheduleParserHelper::storeFFDisutilityCurveKey::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 401 of file ScheduleParserHelper.cpp.

References AIRINV::FareFamilyStruct::_ffDisutilityCurveKey, AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod, AIRINV::SegmentCabinStruct::_itFareFamily, and AIRINV::FlightPeriodStruct::_itSegmentCabin.

22.147.4 Member Data Documentation

22.147.4.1 FlightPeriodStruct& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRINV::ScheduleParserHelper::storeAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRINV::ScheduleParserHelper::storeDow::operator()(), AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeBoardingTime::operator()(), AIRINV::ScheduleParserHelper::storeOffTime::operator()(), AIRINV::ScheduleParserHelper::storeElapsedTime::operator()(), AIRINV::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeCapacity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeClasses::operator()(), AIRINV::ScheduleParserHelper::storeFamilyCode::operator()(), AIRINV::ScheduleParserHelper::storeFRAT5CurveKey::operator()(), operator()(), AIRINV::ScheduleParserHelper::storeFClasses::operator()(), and AIRINV::ScheduleParserHelper::doEndFlight::operator()().

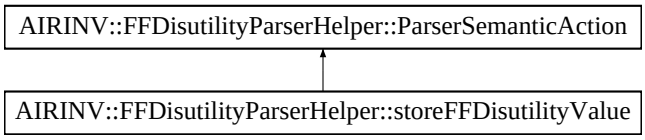
The documentation for this struct was generated from the following files:

- airinv/command/ScheduleParserHelper.hpp
- airinv/command/ScheduleParserHelper.cpp

22.148 AIRINV::FFDisutilityParserHelper::storeFFDisutilityValue Struct Reference

`#include <airinv/command/FFDisutilityParserHelper.hpp>`

Inheritance diagram for AIRINV::FFDisutilityParserHelper::storeFFDisutilityValue:



Public Member Functions

- storeFFDisutilityValue (FFDisutilityStruct &)
- void operator() (double iReal) const

Public Attributes

- FFDisutilityStruct & _ffDisutility

22.148.1 Detailed Description

Store the FFDisutility value.

Definition at line 53 of file FFDisutilityParserHelper.hpp.

22.148.2 Constructor & Destructor Documentation

22.148.2.1 AIRINV::FFDisutilityParserHelper::storeFFDisutilityValue::storeFFDisutilityValue (FFDisutilityStruct & *ioFFDisutility*)

Actor Constructor.

Definition at line 56 of file FFDisutilityParserHelper.cpp.

22.148.3 Member Function Documentation

22.148.3.1 void AIRINV::FFDisutilityParserHelper::storeFFDisutilityValue::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 61 of file FFDisutilityParserHelper.cpp.

References AIRINV::FFDisutilityStruct::_curve, AIRINV::FFDisutilityStruct::_dtd, and AIRINV::FFDisutilityParserHelper::ParserSemanticAction::_ffDisutility.

22.148.4 Member Data Documentation

22.148.4.1 FFDisutilityStruct& AIRINV::FFDisutilityParserHelper::ParserSemanticAction::_ffDisutility [inherited]

Actor Context.

Definition at line 33 of file FFDisutilityParserHelper.hpp.

Referenced by AIRINV::FFDisutilityParserHelper::storeCurveKey::operator()(), AIRINV::FFDisutilityParserHelper::storeDTD::operator()(), operator()(), and AIRINV::FFDisutilityParserHelper::doEndCurve::operator()().

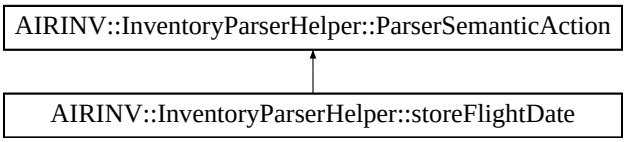
The documentation for this struct was generated from the following files:

- airinv/command/FFDisutilityParserHelper.hpp
- airinv/command/FFDisutilityParserHelper.cpp

22.149 AIRINV::InventoryParserHelper::storeFlightDate Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeFlightDate:



Public Member Functions

- storeFlightDate (FlightDateStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightDateStruct & _flightDate

22.149.1 Detailed Description

Store the flight date.

Definition at line 61 of file InventoryParserHelper.hpp.

22.149.2 Constructor & Destructor Documentation

22.149.2.1 AIRINV::InventoryParserHelper::storeFlightDate::storeFlightDate (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 80 of file InventoryParserHelper.cpp.

22.149.3 Member Function Documentation

22.149.3.1 void AIRINV::InventoryParserHelper::storeFlightDate::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 85 of file InventoryParserHelper.cpp.

References AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::FlightDateStruct::_flightDate, and AIRINV::FlightDateStruct::getDate().

22.149.4 Member Data Documentation

22.149.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV-

::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

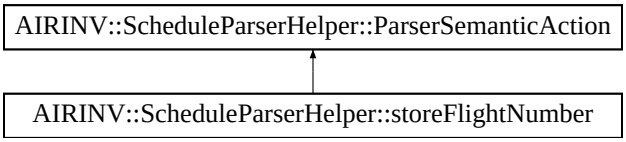
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.150 AIRINV::ScheduleParserHelper::storeFlightNumber Struct Reference

`#include <airinv/command/ScheduleParserHelper.hpp>`

Inheritance diagram for AIRINV::ScheduleParserHelper::storeFlightNumber:



Public Member Functions

- storeFlightNumber (FlightPeriodStruct &)
- void operator() (unsigned int iNumber) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

22.150.1 Detailed Description

Store the parsed flight number.

Definition at line 45 of file ScheduleParserHelper.hpp.

22.150.2 Constructor & Destructor Documentation

22.150.2.1 AIRINV::ScheduleParserHelper::storeFlightNumber::storeFlightNumber (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 51 of file ScheduleParserHelper.cpp.

22.150.3 Member Function Documentation

22.150.3.1 void AIRINV::ScheduleParserHelper::storeFlightNumber::operator() (unsigned int *iNumber*) const

Actor Function (functor).

Definition at line 56 of file ScheduleParserHelper.cpp.

References AIRINV::FlightPeriodStruct::_flightNumber, and AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod.

22.150.4 Member Data Documentation

22.150.4.1 FlightPeriodStruct& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRINV::ScheduleParserHelper::storeAirlineCode::operator()(), operator()(), AIRINV::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRINV::ScheduleParserHelper::storeDow::operator()(), AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeBoardingTime::operator()(), AIRINV::ScheduleParserHelper::storeOffTime::operator()(), AIRINV::ScheduleParserHelper::storeElapsedTime::operator()(), AIRINV::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeCapacity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeClasses::operator()(), AIRINV::ScheduleParserHelper::storeFamilyCode::operator()(), AIRINV::ScheduleParserHelper::storeFRAT5CurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFFDisutilityCurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFClasses::operator()(), and AIRINV::ScheduleParserHelper::doEndFlight::operator()().

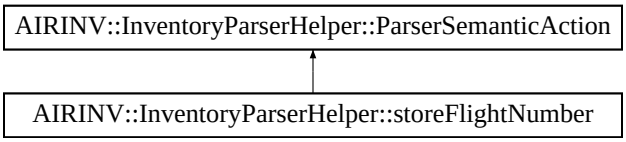
The documentation for this struct was generated from the following files:

- airinv/command/ScheduleParserHelper.hpp
- airinv/command/ScheduleParserHelper.cpp

22.151 AIRINV::InventoryParserHelper::storeFlightNumber Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeFlightNumber:



Public Member Functions

- storeFlightNumber (FlightDateStruct &)
- void operator() (unsigned int iNumber) const

Public Attributes

- FlightDateStruct & _flightDate

22.151.1 Detailed Description

Store the parsed flight number.

Definition at line 53 of file InventoryParserHelper.hpp.

22.151.2 Constructor & Destructor Documentation

22.151.2.1 AIRINV::InventoryParserHelper::storeFlightNumber::storeFlightNumber (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 70 of file InventoryParserHelper.cpp.

22.151.3 Member Function Documentation

22.151.3.1 void AIRINV::InventoryParserHelper::storeFlightNumber::operator() (unsigned int *iNumber*) const

Actor Function (functor).

Definition at line 75 of file InventoryParserHelper.cpp.

References AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, and AIRINV::FlightDateStruct::_flightNumber.

22.151.4 Member Data Documentation

22.151.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

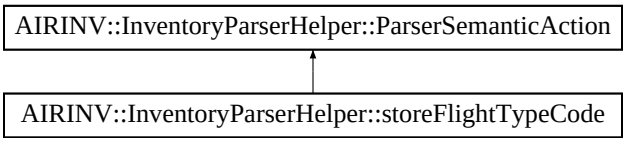
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.152 AIRINV::InventoryParserHelper::storeFlightTypeCode Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeFlightTypeCode:



Public Member Functions

- storeFlightTypeCode (FlightDateStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightDateStruct & _flightDate

22.152.1 Detailed Description

Store the flight type code.

Definition at line 69 of file InventoryParserHelper.hpp.

22.152.2 Constructor & Destructor Documentation

22.152.2.1 AIRINV::InventoryParserHelper::storeFlightTypeCode::storeFlightTypeCode (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 91 of file InventoryParserHelper.cpp.

22.152.3 Member Function Documentation

22.152.3.1 void AIRINV::InventoryParserHelper::storeFlightTypeCode::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 96 of file InventoryParserHelper.cpp.

References AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::FlightDateStruct::_flightTypeCode, and AIRINV::FlightTypeCode::getCode().

22.152.4 Member Data Documentation

22.152.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

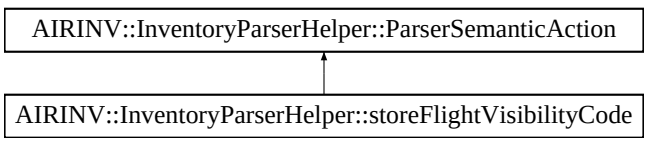
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.153 AIRINV::InventoryParserHelper::storeFlightVisibilityCode Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeFlightVisibilityCode:



Public Member Functions

- storeFlightVisibilityCode (FlightDateStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightDateStruct & _flightDate

22.153.1 Detailed Description

Store the flight visibility code.

Definition at line 77 of file InventoryParserHelper.hpp.

22.153.2 Constructor & Destructor Documentation

22.153.2.1 AIRINV::InventoryParserHelper::storeFlightVisibilityCode::storeFlightVisibilityCode (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 106 of file InventoryParserHelper.cpp.

22.153.3 Member Function Documentation

22.153.3.1 void AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 111 of file InventoryParserHelper.cpp.

References AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::FlightDateStruct::_flightVisibilityCode, and AIRINV::FlightVisibilityCode::getCode().

22.153.4 Member Data Documentation

22.153.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(),

AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

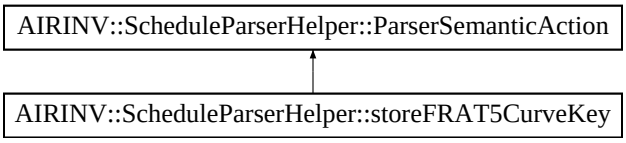
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.154 AIRINV::ScheduleParserHelper::storeFRAT5CurveKey Struct Reference

`#include <airinv/command/ScheduleParserHelper.hpp>`

Inheritance diagram for AIRINV::ScheduleParserHelper::storeFRAT5CurveKey:



Public Member Functions

- storeFRAT5CurveKey (FlightPeriodStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

22.154.1 Detailed Description

Store the FRAT5 curve key.

Definition at line 200 of file ScheduleParserHelper.hpp.

22.154.2 Constructor & Destructor Documentation

22.154.2.1 AIRINV::ScheduleParserHelper::storeFRAT5CurveKey::storeFRAT5CurveKey (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 382 of file ScheduleParserHelper.cpp.

22.154.3 Member Function Documentation

22.154.3.1 void AIRINV::ScheduleParserHelper::storeFRAT5CurveKey::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 387 of file ScheduleParserHelper.cpp.

References AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod, AIRINV::FareFamilyStruct::_frat5CurveKey, AIRINV::SegmentCabinStruct::_itFareFamily, and AIRINV::FlightPeriodStruct::_itSegmentCabin.

22.154.4 Member Data Documentation

22.154.4.1 FlightPeriodStruct& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRINV::ScheduleParserHelper::storeAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRINV::ScheduleParserHelper::storeDow::operator()(), AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeBoardingTime::operator()(), AIRINV::ScheduleParserHelper::storeOffTime::operator()(), AIRINV::ScheduleParserHelper::storeElapsedTime::operator()(), AIRINV::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeCapacity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeClasses::operator()(), AIRINV::ScheduleParserHelper::storeFamilyCode::operator()(), operator()(), AIRINV::ScheduleParserHelper::storeFFDisutilityCurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFClasses::operator()(), and AIRINV::ScheduleParserHelper::doEndFlight::operator()().

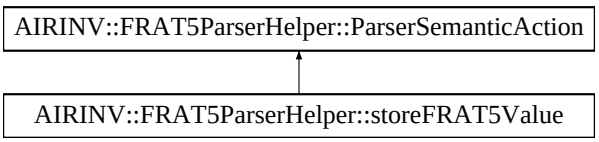
The documentation for this struct was generated from the following files:

- airinv/command/ScheduleParserHelper.hpp
- airinv/command/ScheduleParserHelper.cpp

22.155 AIRINV::FRAT5ParserHelper::storeFRAT5Value Struct Reference

`#include <airinv/command/FRAT5ParserHelper.hpp>`

Inheritance diagram for AIRINV::FRAT5ParserHelper::storeFRAT5Value:



Public Member Functions

- storeFRAT5Value (FRAT5Struct &)
- void operator() (double iReal) const

Public Attributes

- FRAT5Struct & _frat5

22.155.1 Detailed Description

Store the FRAT5 value.

Definition at line 53 of file FRAT5ParserHelper.hpp.

22.155.2 Constructor & Destructor Documentation

22.155.2.1 AIRINV::FRAT5ParserHelper::storeFRAT5Value::storeFRAT5Value (FRAT5Struct & *ioFRAT5*)

Actor Constructor.

Definition at line 57 of file FRAT5ParserHelper.cpp.

22.155.3 Member Function Documentation

22.155.3.1 void AIRINV::FRAT5ParserHelper::storeFRAT5Value::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 62 of file FRAT5ParserHelper.cpp.

References AIRINV::FRAT5Struct::_curve, AIRINV::FRAT5Struct::_dtd, and AIRINV::FRAT5ParserHelper::ParserSemanticAction::_frat5.

22.155.4 Member Data Documentation

22.155.4.1 FRAT5Struct& AIRINV::FRAT5ParserHelper::ParserSemanticAction::_frat5 [inherited]

Actor Context.

Definition at line 33 of file FRAT5ParserHelper.hpp.

Referenced by AIRINV::FRAT5ParserHelper::storeCurveKey::operator()(), AIRINV::FRAT5ParserHelper::storeDTD::operator()(), operator()(), and AIRINV::FRAT5ParserHelper::doEndCurve::operator()().

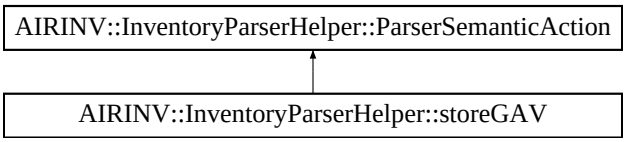
The documentation for this struct was generated from the following files:

- airinv/command/FRAT5ParserHelper.hpp
- airinv/command/FRAT5ParserHelper.cpp

22.156 AIRINV::InventoryParserHelper::storeGAV Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeGAV:



Public Member Functions

- storeGAV (FlightDateStruct &)
- void operator() (double iReal) const

Public Attributes

- FlightDateStruct & _flightDate

22.156.1 Detailed Description

Store the parsed Gross Availability (GAV).

Definition at line 197 of file InventoryParserHelper.hpp.

22.156.2 Constructor & Destructor Documentation

22.156.2.1 AIRINV::InventoryParserHelper::storeGAV::storeGAV (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 339 of file InventoryParserHelper.cpp.

22.156.3 Member Function Documentation

22.156.3.1 void AIRINV::InventoryParserHelper::storeGAV::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 344 of file InventoryParserHelper.cpp.

References AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::LegCabinStruct::_gav, and AIRINV::FlightDateStruct::_itLegCabin.

22.156.4 Member Data Documentation

22.156.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV-

::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

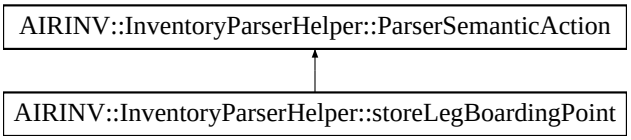
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.157 AIRINV::InventoryParserHelper::storeLegBoardingPoint Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeLegBoardingPoint:



Public Member Functions

- storeLegBoardingPoint (FlightDateStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightDateStruct & _flightDate

22.157.1 Detailed Description

Store the parsed leg boarding point.

Definition at line 85 of file InventoryParserHelper.hpp.

22.157.2 Constructor & Destructor Documentation

22.157.2.1 AIRINV::InventoryParserHelper::storeLegBoardingPoint::storeLegBoardingPoint (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 121 of file InventoryParserHelper.cpp.

22.157.3 Member Function Documentation

22.157.3.1 void AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 126 of file InventoryParserHelper.cpp.

References AIRINV::LegStruct::_airlineCode, AIRINV::FlightDateStruct::_airlineCode, AIRINV::LegStruct::_boardingPoint, AIRINV::LegCabinStruct::_bucketList, AIRINV::LegCabinStruct::_cabinCode, AIRINV::LegStruct::_cabinList, AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::LegStruct::_flightNumber, AIRINV::FlightDateStruct::_flightNumber, AIRINV::FlightDateStruct::_itBucket, AIRINV::FlightDateStruct::_itLeg, AIRINV::FlightDateStruct::_itLegCabin, AIRINV::FlightDateStruct::_legList, AIRINV::BucketStruct::_yieldRangeUpperValue, and AIRINV::FlightDateStruct::addAirport().

22.157.4 Member Data Documentation

22.157.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

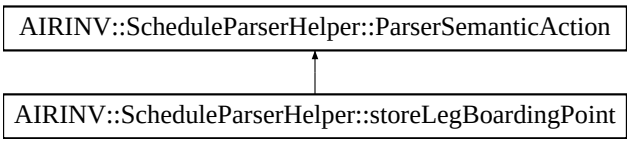
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.158 AIRINV::ScheduleParserHelper::storeLegBoardingPoint Struct Reference

`#include <airinv/command/ScheduleParserHelper.hpp>`

Inheritance diagram for AIRINV::ScheduleParserHelper::storeLegBoardingPoint:



Public Member Functions

- storeLegBoardingPoint (FlightPeriodStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

22.158.1 Detailed Description

Store the parsed leg boarding point.

Definition at line 77 of file ScheduleParserHelper.hpp.

22.158.2 Constructor & Destructor Documentation

22.158.2.1 AIRINV::ScheduleParserHelper::storeLegBoardingPoint::storeLegBoardingPoint (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 117 of file ScheduleParserHelper.cpp.

22.158.3 Member Function Documentation

22.158.3.1 void AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 122 of file ScheduleParserHelper.cpp.

References AIRINV::LegStruct::_boardingPoint, AIRINV::LegStruct::_cabinList, AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod, AIRINV::FlightPeriodStruct::_itLeg, AIRINV::FlightPeriodStruct::_legAlreadyDefined, AIRINV::FlightPeriodStruct::_legList, and AIRINV::FlightPeriodStruct::addAirport().

22.158.4 Member Data Documentation

22.158.4.1 FlightPeriodStruct& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRINV::ScheduleParserHelper::storeAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRINV::ScheduleParserHelper::storeDow::operator()(), operator()(), AIRINV::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeBoardingTime::operator()(), AIRINV::Schedule-

ParserHelper::storeOffTime::operator()(), AIRINV::ScheduleParserHelper::storeElapsedTime::operator()(), AIRINV::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeCapacity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeClasses::operator()(), AIRINV::ScheduleParserHelper::storeFamilyCode::operator()(), AIRINV::ScheduleParserHelper::storeFRAT5CurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFFDisutilityCurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFClasses::operator()(), and AIRINV::ScheduleParserHelper::doEndFlight::operator()().

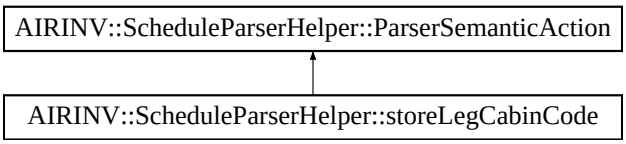
The documentation for this struct was generated from the following files:

- airinv/command/ScheduleParserHelper.hpp
- airinv/command/ScheduleParserHelper.cpp

22.159 AIRINV::ScheduleParserHelper::storeLegCabinCode Struct Reference

`#include <airinv/command/ScheduleParserHelper.hpp>`

Inheritance diagram for AIRINV::ScheduleParserHelper::storeLegCabinCode:



Public Member Functions

- storeLegCabinCode (FlightPeriodStruct &)
- void operator() (char iChar) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

22.159.1 Detailed Description

Store the parsed leg cabin code.

Definition at line 133 of file ScheduleParserHelper.hpp.

22.159.2 Constructor & Destructor Documentation

22.159.2.1 AIRINV::ScheduleParserHelper::storeLegCabinCode::storeLegCabinCode (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 250 of file ScheduleParserHelper.cpp.

22.159.3 Member Function Documentation

22.159.3.1 void AIRINV::ScheduleParserHelper::storeLegCabinCode::operator() (char *iChar*) const

Actor Function (functor).

Definition at line 255 of file ScheduleParserHelper.cpp.

References AIRINV::LegCabinStruct::_cabinCode, AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod, and AIRINV::FlightPeriodStruct::_itLegCabin.

22.159.4 Member Data Documentation

22.159.4.1 FlightPeriodStruct& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRINV::ScheduleParserHelper::storeAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRINV::ScheduleParserHelper::storeDow::operator()(), AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeBoardingTime::operator()(), AIRINV::ScheduleParserHelper::storeOffTime::operator()(), AIRINV::ScheduleParserHelper::storeElapsedTime::operator()(), operator()(), AIRINV::ScheduleParserHelper::storeCapacity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeClasses::operator()(), AIRINV::ScheduleParserHelper::storeFamilyCode::operator()(), AIRINV::ScheduleParserHelper::storeFRAT5CurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFFDisutilityCurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFClasses::operator()(), and AIRINV::ScheduleParserHelper::doEndFlight::operator()().

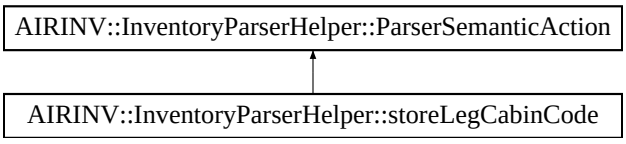
The documentation for this struct was generated from the following files:

- airinv/command/ScheduleParserHelper.hpp
- airinv/command/ScheduleParserHelper.cpp

22.160 AIRINV::InventoryParserHelper::storeLegCabinCode Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeLegCabinCode:



Public Member Functions

- storeLegCabinCode (FlightDateStruct &)
- void operator() (char iChar) const

Public Attributes

- FlightDateStruct & _flightDate

22.160.1 Detailed Description

Store the parsed leg cabin code.

Definition at line 149 of file InventoryParserHelper.hpp.

22.160.2 Constructor & Destructor Documentation

22.160.2.1 AIRINV::InventoryParserHelper::storeLegCabinCode::storeLegCabinCode (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 255 of file InventoryParserHelper.cpp.

22.160.3 Member Function Documentation

22.160.3.1 void AIRINV::InventoryParserHelper::storeLegCabinCode::operator() (char *iChar*) const

Actor Function (functor).

Definition at line 260 of file InventoryParserHelper.cpp.

References AIRINV::LegCabinStruct::_bucketList, AIRINV::LegCabinStruct::_cabinCode, AIRINV::LegStruct::_cabinList, AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::FlightDateStruct::_itBucket, AIRINV::FlightDateStruct::_itLeg, AIRINV::FlightDateStruct::_itLegCabin, and AIRINV::BucketStruct::_yieldRangeUpperValue.

22.160.4 Member Data Documentation

22.160.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(),

AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

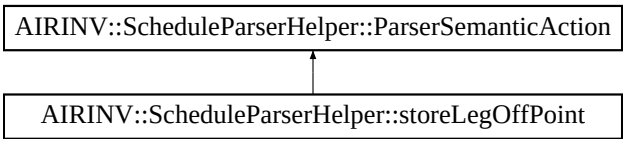
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.161 AIRINV::ScheduleParserHelper::storeLegOffPoint Struct Reference

`#include <airinv/command/ScheduleParserHelper.hpp>`

Inheritance diagram for AIRINV::ScheduleParserHelper::storeLegOffPoint:



Public Member Functions

- storeLegOffPoint (FlightPeriodStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

22.161.1 Detailed Description

Store the parsed leg off point.

Definition at line 85 of file ScheduleParserHelper.hpp.

22.161.2 Constructor & Destructor Documentation

22.161.2.1 AIRINV::ScheduleParserHelper::storeLegOffPoint::storeLegOffPoint (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 146 of file ScheduleParserHelper.cpp.

22.161.3 Member Function Documentation

22.161.3.1 void AIRINV::ScheduleParserHelper::storeLegOffPoint::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 151 of file ScheduleParserHelper.cpp.

References AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod, AIRINV::FlightPeriodStruct::_itLeg, AIRINV::LegStruct::_offPoint, and AIRINV::FlightPeriodStruct::addAirport().

22.161.4 Member Data Documentation

22.161.4.1 FlightPeriodStruct& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRINV::ScheduleParserHelper::storeAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRINV::ScheduleParserHelper::storeDow::operator()(), AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator()(), operator()(), AIRINV::ScheduleParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeBoardingTime::operator()(), AIRINV::ScheduleParserHelper::storeOffTime::operator()(), AIRINV::ScheduleParserHelper::storeElapsedTime::operator()(), AIRINV::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeCapacity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeClasses::operator()(), AIRINV::ScheduleParserHelper::storeFamilyCode::operator()(), AIRINV::ScheduleParserHelper::storeFRAT5CurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFFDisutilityCurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFClasses::operator()(), and AIRINV::ScheduleParserHelper::doEndFlight::operator()().

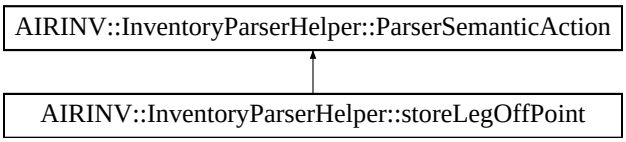
The documentation for this struct was generated from the following files:

- airinv/command/ScheduleParserHelper.hpp
- airinv/command/ScheduleParserHelper.cpp

22.162 AIRINV::InventoryParserHelper::storeLegOffPoint Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeLegOffPoint:



Public Member Functions

- storeLegOffPoint (FlightDateStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightDateStruct & _flightDate

22.162.1 Detailed Description

Store the parsed leg off point.

Definition at line 93 of file InventoryParserHelper.hpp.

22.162.2 Constructor & Destructor Documentation

22.162.2.1 AIRINV::InventoryParserHelper::storeLegOffPoint::storeLegOffPoint (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 160 of file InventoryParserHelper.cpp.

22.162.3 Member Function Documentation

22.162.3.1 void AIRINV::InventoryParserHelper::storeLegOffPoint::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 165 of file InventoryParserHelper.cpp.

References AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::FlightDateStruct::_itLeg, AIRINV::LegStruct::_offPoint, and AIRINV::FlightDateStruct::addAirport().

22.162.4 Member Data Documentation

22.162.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AI-

RINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate-::operator()().

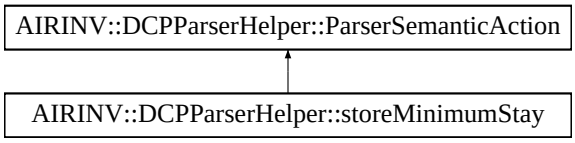
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.163 AIRINV::DCPParserHelper::storeMinimumStay Struct Reference

`#include <airinv/command/vault/DCPParserHelper.hpp>`

Inheritance diagram for AIRINV::DCPParserHelper::storeMinimumStay:



Public Member Functions

- storeMinimumStay (DCPRuleStruct &)
- void operator() (unsigned int, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- DCPRuleStruct & _DCPRule

22.163.1 Detailed Description

Store the parsed minimum stay.

Definition at line 178 of file DCPParserHelper.hpp.

22.163.2 Constructor & Destructor Documentation

22.163.2.1 AIRINV::DCPParserHelper::storeMinimumStay::storeMinimumStay (DCPRuleStruct & *ioDCPRule*)

Actor Constructor.

Definition at line 299 of file DCPParserHelper.cpp.

22.163.3 Member Function Documentation

22.163.3.1 void AIRINV::DCPParserHelper::storeMinimumStay::operator() (unsigned int *iMinStay,* boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 304 of file DCPParserHelper.cpp.

References AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule.

22.163.4 Member Data Documentation

22.163.4.1 DCPRuleStruct& AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule [inherited]

Actor Context.

Definition at line 34 of file DCPParserHelper.hpp.

Referenced by AIRINV::DCPParserHelper::storeDCPId::operator()(), AIRINV::DCPParserHelper::storeOrigin::operator()(), AIRINV::DCPParserHelper::storeDestination::operator()(), AIRINV::DCPParserHelper::storeDateRangeStart::operator()(), AIRINV::DCPParserHelper::storeDateRangeEnd::operator()(), AIRINV::DCPParserHelper::storeStartRangeTime::operator()(), AIRINV::DCPParserHelper::storeEndRangeTime::operator()(), AIRINV::DCPParserHelper::storePOS::operator()(), AIRINV::DCPParserHelper::storeCabinCode::operator()(), AIRINV::DCPParserHelper::storeChannel::operator()(), AIRINV::DCPParserHelper::storeAdvancePurchase::operator()(), AIRINV::DCPParserHelper::storeSaturdayStay::operator()(), AIRINV::DCPParserHelper::storeChangeFees::operator()(), AIRINV::DCPParserHelper::storeNonRefundable::operator()(), operator()(), AIRINV::DCPParserHelper::storeDCP::operator()(), AIRINV::DCPParserHelper::storeAirlineCode::operator()(), AIRINV::DCPParserHelper::storeClass::operator()(), and AIRINV::DCPParserHelper::doEndDCP::operator()().

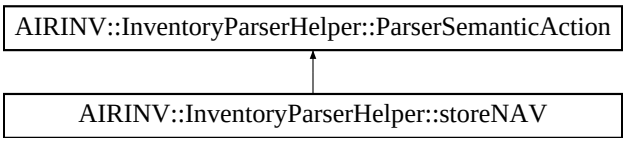
The documentation for this struct was generated from the following files:

- airinv/command/vault/DCPParserHelper.hpp
- airinv/command/vault/DCPParserHelper.cpp

22.164 AIRINV::InventoryParserHelper::storeNAV Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeNAV:



Public Member Functions

- storeNAV (FlightDateStruct &)
- void operator() (double iReal) const

Public Attributes

- FlightDateStruct & _flightDate

22.164.1 Detailed Description

Store the parsed Net Availability (NAV).

Definition at line 189 of file InventoryParserHelper.hpp.

22.164.2 Constructor & Destructor Documentation

22.164.2.1 AIRINV::InventoryParserHelper::storeNAV::storeNAV (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 328 of file InventoryParserHelper.cpp.

22.164.3 Member Function Documentation

22.164.3.1 void AIRINV::InventoryParserHelper::storeNAV::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 333 of file InventoryParserHelper.cpp.

References AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::FlightDateStruct::_itLegCabin, and AIRINV::LegCabinStruct::_nav.

22.164.4 Member Data Documentation

22.164.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

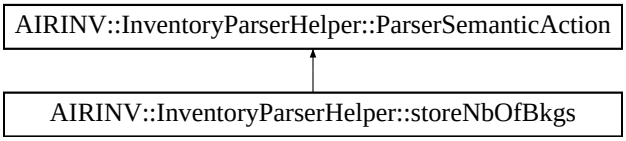
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.165 AIRINV::InventoryParserHelper::storeNbOfBkgs Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeNbOfBkgs:



Public Member Functions

- storeNbOfBkgs (FlightDateStruct &)
- void operator() (double iReal) const

Public Attributes

- FlightDateStruct & _flightDate

22.165.1 Detailed Description

Store the parsed number of bookings (at booking class level).

Definition at line 349 of file InventoryParserHelper.hpp.

22.165.2 Constructor & Destructor Documentation

22.165.2.1 AIRINV::InventoryParserHelper::storeNbOfBkgs::storeNbOfBkgs (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 634 of file InventoryParserHelper.cpp.

22.165.3 Member Function Documentation

22.165.3.1 void AIRINV::InventoryParserHelper::storeNbOfBkgs::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 639 of file InventoryParserHelper.cpp.

References AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::FlightDateStruct::_itBookingClass, and AIRINV::BookingClassStruct::_nbOfBookings.

22.165.4 Member Data Documentation

22.165.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIR-

INV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

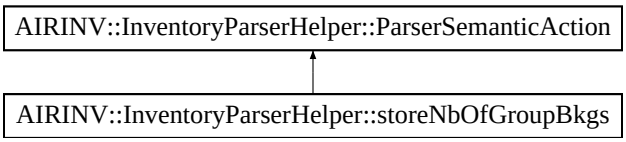
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.166 AIRINV::InventoryParserHelper::storeNbOfGroupBkgs Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeNbOfGroupBkgs:



Public Member Functions

- storeNbOfGroupBkgs (FlightDateStruct &)
- void operator() (double iReal) const

Public Attributes

- FlightDateStruct & _flightDate

22.166.1 Detailed Description

Store the parsed number of group bookings (at booking class level).

Definition at line 357 of file InventoryParserHelper.hpp.

22.166.2 Constructor & Destructor Documentation

22.166.2.1 AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::storeNbOfGroupBkgs (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 645 of file InventoryParserHelper.cpp.

22.166.3 Member Function Documentation

22.166.3.1 void AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 650 of file InventoryParserHelper.cpp.

References AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::FlightDateStruct::_itBookingClass, and AIRINV::BookingClassStruct::_nbOfGroupBookings.

22.166.4 Member Data Documentation

22.166.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

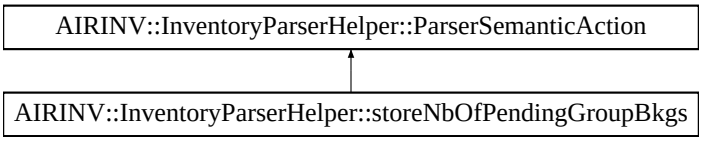
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.167 AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs:



Public Member Functions

- storeNbOfPendingGroupBkgs (FlightDateStruct &)
- void operator() (double iReal) const

Public Attributes

- FlightDateStruct & _flightDate

22.167.1 Detailed Description

Store the parsed number of pending group bookings (at booking class level).

Definition at line 365 of file InventoryParserHelper.hpp.

22.167.2 Constructor & Destructor Documentation

22.167.2.1 AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::storeNbOfPendingGroupBkgs (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 657 of file InventoryParserHelper.cpp.

22.167.3 Member Function Documentation

22.167.3.1 void AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 662 of file InventoryParserHelper.cpp.

References AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::FlightDateStruct::_itBookingClass, and AIRINV::BookingClassStruct::_nbOfPendingGroupBookings.

22.167.4 Member Data Documentation

22.167.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

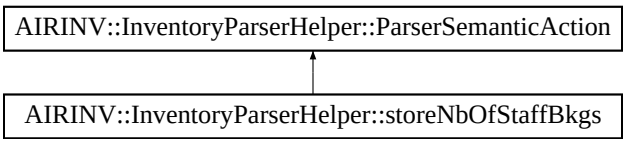
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.168 AIRINV::InventoryParserHelper::storeNbOfStaffBkgs Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeNbOfStaffBkgs:



Public Member Functions

- storeNbOfStaffBkgs (FlightDateStruct &)
- void operator() (double iReal) const

Public Attributes

- FlightDateStruct & _flightDate

22.168.1 Detailed Description

Store the parsed number of staff bookings (at booking class level).

Definition at line 373 of file InventoryParserHelper.hpp.

22.168.2 Constructor & Destructor Documentation

22.168.2.1 AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::storeNbOfStaffBkgs (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 668 of file InventoryParserHelper.cpp.

22.168.3 Member Function Documentation

22.168.3.1 void AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 673 of file InventoryParserHelper.cpp.

References AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::FlightDateStruct::_itBookingClass, and AIRINV::BookingClassStruct::_nbOfStaffBookings.

22.168.4 Member Data Documentation

22.168.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper-

::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

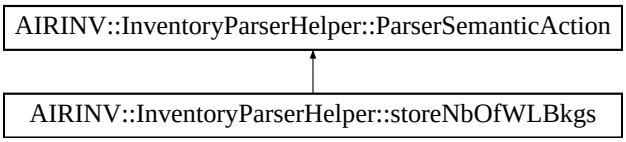
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.169 AIRINV::InventoryParserHelper::storeNbOfWLBkgs Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeNbOfWLBkgs:



Public Member Functions

- storeNbOfWLBkgs (FlightDateStruct &)
- void operator() (double iReal) const

Public Attributes

- FlightDateStruct & _flightDate

22.169.1 Detailed Description

Store the parsed number of wait-list bookings (at booking class level).

Definition at line 382 of file InventoryParserHelper.hpp.

22.169.2 Constructor & Destructor Documentation

22.169.2.1 AIRINV::InventoryParserHelper::storeNbOfWLBkgs::storeNbOfWLBkgs (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 679 of file InventoryParserHelper.cpp.

22.169.3 Member Function Documentation

22.169.3.1 void AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 684 of file InventoryParserHelper.cpp.

References AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::FlightDateStruct::_itBookingClass, and AIRINV::BookingClassStruct::_nbOfWLBookings.

22.169.4 Member Data Documentation

22.169.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

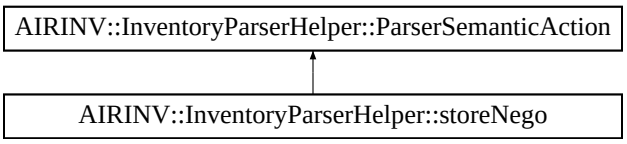
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.170 AIRINV::InventoryParserHelper::storeNego Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeNego:



Public Member Functions

- storeNego (FlightDateStruct &)
- void operator() (double iReal) const

Public Attributes

- FlightDateStruct & _flightDate

22.170.1 Detailed Description

Store the negotiated allotment (at booking class level).

Definition at line 325 of file InventoryParserHelper.hpp.

22.170.2 Constructor & Destructor Documentation

22.170.2.1 AIRINV::InventoryParserHelper::storeNego::storeNego (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 601 of file InventoryParserHelper.cpp.

22.170.3 Member Function Documentation

22.170.3.1 void AIRINV::InventoryParserHelper::storeNego::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 606 of file InventoryParserHelper.cpp.

References AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::FlightDateStruct::_itBookingClass, and AIRINV::BookingClassStruct::_nego.

22.170.4 Member Data Documentation

22.170.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper-

::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

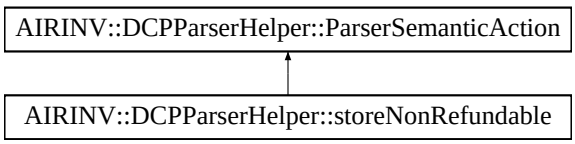
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.171 AIRINV::DCPParserHelper::storeNonRefundable Struct Reference

`#include <airinv/command/vault/DCPParserHelper.hpp>`

Inheritance diagram for AIRINV::DCPParserHelper::storeNonRefundable:



Public Member Functions

- storeNonRefundable (DCPRuleStruct &)
- void operator() (char, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- DCPRuleStruct & _DCPRule

22.171.1 Detailed Description

Store the parsed refundable option

Definition at line 168 of file DCPParserHelper.hpp.

22.171.2 Constructor & Destructor Documentation

22.171.2.1 AIRINV::DCPParserHelper::storeNonRefundable::storeNonRefundable (DCPRuleStruct & *ioDCPRule*)

Actor Constructor.

Definition at line 274 of file DCPParserHelper.cpp.

22.171.3 Member Function Documentation

22.171.3.1 void AIRINV::DCPParserHelper::storeNonRefundable::operator() (char *iNonRefundable,* boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 279 of file DCPParserHelper.cpp.

References AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule.

22.171.4 Member Data Documentation

22.171.4.1 DCPRuleStruct& AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule [inherited]

Actor Context.

Definition at line 34 of file DCPParserHelper.hpp.

Referenced by AIRINV::DCPParserHelper::storeDCPId::operator()(), AIRINV::DCPParserHelper::storeOrigin::operator()(), AIRINV::DCPParserHelper::storeDestination::operator()(), AIRINV::DCPParserHelper::storeDateRangeStart::operator()(), AIRINV::DCPParserHelper::storeDateRangeEnd::operator()(), AIRINV::DCPParserHelper::storeStartRangeTime::operator()(), AIRINV::DCPParserHelper::storeEndRangeTime::operator()(), AIRINV::DCPParserHelper::storePOS::operator()(), AIRINV::DCPParserHelper::storeCabinCode::operator()(), AIRINV::DCPParserHelper::storeChannel::operator()(), AIRINV::DCPParserHelper::storeAdvancePurchase::operator()(), AIRINV::DCPParserHelper::storeSaturdayStay::operator()(), AIRINV::DCPParserHelper::storeChangeFees::operator()(), operator()(), AIRINV::DCPParserHelper::storeMinimumStay::operator()(), AIRINV::DCPParserHelper::storeDCP::operator()(), AIRINV::DCPParserHelper::storeAirlineCode::operator()(), AIRINV::DCPParserHelper::storeClass::operator()(), and AIRINV::DCPParserHelper::doEndDCP::operator()().

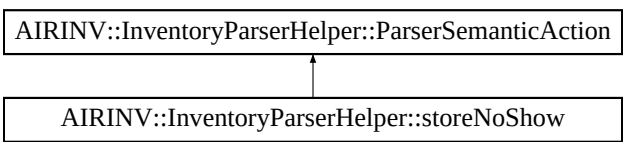
The documentation for this struct was generated from the following files:

- airinv/command/vault/DCPParserHelper.hpp
- airinv/command/vault/DCPParserHelper.cpp

22.172 AIRINV::InventoryParserHelper::storeNoShow Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeNoShow:



Public Member Functions

- storeNoShow (FlightDateStruct &)
- void operator() (double iReal) const

Public Attributes

- FlightDateStruct & _flightDate

22.172.1 Detailed Description

Store the parsed No-Show percentage (at booking class level).

Definition at line 333 of file InventoryParserHelper.hpp.

22.172.2 Constructor & Destructor Documentation

22.172.2.1 AIRINV::InventoryParserHelper::storeNoShow::storeNoShow (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 612 of file InventoryParserHelper.cpp.

22.172.3 Member Function Documentation

22.172.3.1 void AIRINV::InventoryParserHelper::storeNoShow::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 617 of file InventoryParserHelper.cpp.

References AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::FlightDateStruct::_itBookingClass, and AIRINV::BookingClassStruct::_noShowPercentage.

22.172.4 Member Data Documentation

22.172.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(),

AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

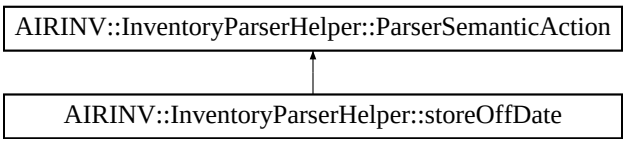
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.173 AIRINV::InventoryParserHelper::storeOffDate Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeOffDate:



Public Member Functions

- storeOffDate (FlightDateStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightDateStruct & _flightDate

22.173.1 Detailed Description

Store the off date.

Definition at line 133 of file InventoryParserHelper.hpp.

22.173.2 Constructor & Destructor Documentation

22.173.2.1 AIRINV::InventoryParserHelper::storeOffDate::storeOffDate (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 232 of file InventoryParserHelper.cpp.

22.173.3 Member Function Documentation

22.173.3.1 void AIRINV::InventoryParserHelper::storeOffDate::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 237 of file InventoryParserHelper.cpp.

References AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::FlightDateStruct::_itLeg, AIRINV::LegStruct::_offDate, and AIRINV::FlightDateStruct::getDate().

22.173.4 Member Data Documentation

22.173.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

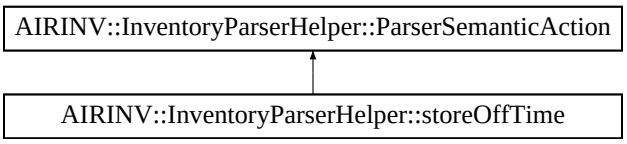
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.174 AIRINV::InventoryParserHelper::storeOffTime Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeOffTime:



Public Member Functions

- storeOffTime (FlightDateStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightDateStruct & _flightDate

22.174.1 Detailed Description

Store the off time.

Definition at line 141 of file InventoryParserHelper.hpp.

22.174.2 Constructor & Destructor Documentation

22.174.2.1 AIRINV::InventoryParserHelper::storeOffTime::storeOffTime (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 242 of file InventoryParserHelper.cpp.

22.174.3 Member Function Documentation

22.174.3.1 void AIRINV::InventoryParserHelper::storeOffTime::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 247 of file InventoryParserHelper.cpp.

References AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::FlightDateStruct::_itLeg, AIRINV::FlightDateStruct::_itSeconds, AIRINV::LegStruct::_offTime, and AIRINV::FlightDateStruct::getTime().

22.174.4 Member Data Documentation

22.174.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::Inventory-

ParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

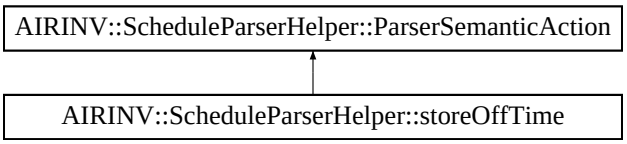
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.175 AIRINV::ScheduleParserHelper::storeOffTime Struct Reference

`#include <airinv/command/ScheduleParserHelper.hpp>`

Inheritance diagram for AIRINV::ScheduleParserHelper::storeOffTime:



Public Member Functions

- storeOffTime (FlightPeriodStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

22.175.1 Detailed Description

Store the off time.

Definition at line 117 of file ScheduleParserHelper.hpp.

22.175.2 Constructor & Destructor Documentation

22.175.2.1 AIRINV::ScheduleParserHelper::storeOffTime::storeOffTime (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 208 of file ScheduleParserHelper.cpp.

22.175.3 Member Function Documentation

22.175.3.1 void AIRINV::ScheduleParserHelper::storeOffTime::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 213 of file ScheduleParserHelper.cpp.

References AIRINV::LegStruct::_boardingDateOffset, AIRINV::FlightPeriodStruct::_dateOffset, AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod, AIRINV::FlightPeriodStruct::_itLeg, AIRINV::FlightPeriodStruct::_itSeconds, AIRINV::LegStruct::_offTime, and AIRINV::FlightPeriodStruct::getTime().

22.175.4 Member Data Documentation

22.175.4.1 FlightPeriodStruct& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRINV::ScheduleParserHelper::storeAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRINV::ScheduleParserHelper::storeDow::operator()(), AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeBoardingTime::operator()(), operator()(), AIRINV::ScheduleParserHelper::storeElapsedTime::operator()(), AIRINV::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeCapacity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeClasses::operator()(), AIRINV::ScheduleParserHelper::storeFamilyCode::operator()(), AIRINV::ScheduleParserHelper::storeFRAT5CurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFFDisutilityCurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFClasses::operator()(), and AIRINV::ScheduleParserHelper::doEndFlight::operator()().

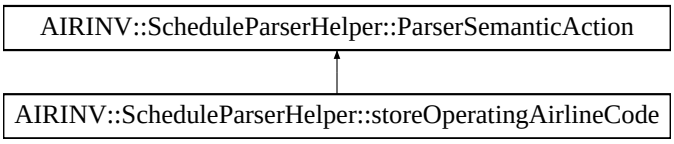
The documentation for this struct was generated from the following files:

- airinv/command/ScheduleParserHelper.hpp
- airinv/command/ScheduleParserHelper.cpp

22.176 AIRINV::ScheduleParserHelper::storeOperatingAirlineCode Struct Reference

`#include <airinv/command/ScheduleParserHelper.hpp>`

Inheritance diagram for AIRINV::ScheduleParserHelper::storeOperatingAirlineCode:



Public Member Functions

- storeOperatingAirlineCode (FlightPeriodStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

22.176.1 Detailed Description

Store the parsed operating airline code.

Definition at line 93 of file ScheduleParserHelper.hpp.

22.176.2 Constructor & Destructor Documentation

22.176.2.1 AIRINV::ScheduleParserHelper::storeOperatingAirlineCode::storeOperatingAirlineCode (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 162 of file ScheduleParserHelper.cpp.

22.176.3 Member Function Documentation

22.176.3.1 void AIRINV::ScheduleParserHelper::storeOperatingAirlineCode::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 167 of file ScheduleParserHelper.cpp.

References AIRINV::LegStruct::_airlineCode, AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod, and AIRINV::FlightPeriodStruct::_itLeg.

22.176.4 Member Data Documentation

22.176.4.1 FlightPeriodStruct& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRINV::ScheduleParserHelper::storeAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRINV::ScheduleParserHelper::storeDow::operator()(), AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeLegOffPoint::operator()(), operator()(), AIRINV::ScheduleParserHelper::storeOperatingFlightNumber::operator()(),

AIRINV::ScheduleParserHelper::storeBoardingTime::operator()(), AIRINV::ScheduleParserHelper::storeOffTime::operator()(), AIRINV::ScheduleParserHelper::storeElapsedTime::operator()(), AIRINV::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeCapacity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeClasses::operator()(), AIRINV::ScheduleParserHelper::storeFamilyCode::operator()(), AIRINV::ScheduleParserHelper::storeFRAT5CurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFFDisutilityCurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFClasses::operator()(), and AIRINV::ScheduleParserHelper::doEndFlight::operator()().

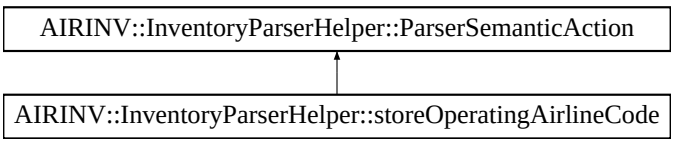
The documentation for this struct was generated from the following files:

- airinv/command/ScheduleParserHelper.hpp
- airinv/command/ScheduleParserHelper.cpp

22.177 AIRINV::InventoryParserHelper::storeOperatingAirlineCode Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeOperatingAirlineCode:



Public Member Functions

- storeOperatingAirlineCode (FlightDateStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightDateStruct & _flightDate

22.177.1 Detailed Description

Store the parsed operating airline code.

Definition at line 101 of file InventoryParserHelper.hpp.

22.177.2 Constructor & Destructor Documentation

22.177.2.1 AIRINV::InventoryParserHelper::storeOperatingAirlineCode::storeOperatingAirlineCode (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 176 of file InventoryParserHelper.cpp.

22.177.3 Member Function Documentation

22.177.3.1 void AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator() (iterator_t *iStr*, iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 181 of file InventoryParserHelper.cpp.

References AIRINV::LegStruct::_airlineCode, AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, and AIRINV::FlightDateStruct::_itLeg.

22.177.4 Member Data Documentation

22.177.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

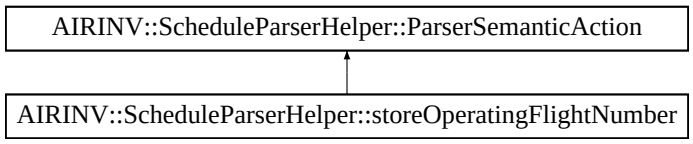
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.178 AIRINV::ScheduleParserHelper::storeOperatingFlightNumber Struct Reference

`#include <airinv/command/ScheduleParserHelper.hpp>`

Inheritance diagram for AIRINV::ScheduleParserHelper::storeOperatingFlightNumber:



Public Member Functions

- storeOperatingFlightNumber (FlightPeriodStruct &)
- void operator() (unsigned int iNumber) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

22.178.1 Detailed Description

Store the parsed operating flight number.

Definition at line 101 of file ScheduleParserHelper.hpp.

22.178.2 Constructor & Destructor Documentation

22.178.2.1 AIRINV::ScheduleParserHelper::storeOperatingFlightNumber::storeOperatingFlightNumber (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 178 of file ScheduleParserHelper.cpp.

22.178.3 Member Function Documentation

22.178.3.1 void AIRINV::ScheduleParserHelper::storeOperatingFlightNumber::operator() (unsigned int *iNumber*) const

Actor Function (functor).

Definition at line 183 of file ScheduleParserHelper.cpp.

References AIRINV::LegStruct::_flightNumber, AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod, and AIRINV::FlightPeriodStruct::_itLeg.

22.178.4 Member Data Documentation

22.178.4.1 FlightPeriodStruct& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRINV::ScheduleParserHelper::storeAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRINV::ScheduleParserHelper::storeDow::operator()(), AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeOperatingAirlineCode::operator()(), operator()(),

AIRINV::ScheduleParserHelper::storeBoardingTime::operator()(), AIRINV::ScheduleParserHelper::storeOffTime::operator()(), AIRINV::ScheduleParserHelper::storeElapsedTime::operator()(), AIRINV::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeCapacity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeClasses::operator()(), AIRINV::ScheduleParserHelper::storeFamilyCode::operator()(), AIRINV::ScheduleParserHelper::storeFRAT5CurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFFDisutilityCurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFClasses::operator()(), and AIRINV::ScheduleParserHelper::doEndFlight::operator()().

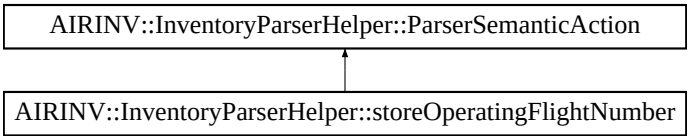
The documentation for this struct was generated from the following files:

- airinv/command/ScheduleParserHelper.hpp
- airinv/command/ScheduleParserHelper.cpp

22.179 AIRINV::InventoryParserHelper::storeOperatingFlightNumber Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeOperatingFlightNumber:



Public Member Functions

- storeOperatingFlightNumber (FlightDateStruct &)
- void operator() (unsigned int iNumber) const

Public Attributes

- FlightDateStruct & _flightDate

22.179.1 Detailed Description

Store the parsed operating flight number.

Definition at line 109 of file InventoryParserHelper.hpp.

22.179.2 Constructor & Destructor Documentation

22.179.2.1 AIRINV::InventoryParserHelper::storeOperatingFlightNumber::storeOperatingFlightNumber (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 192 of file InventoryParserHelper.cpp.

22.179.3 Member Function Documentation

22.179.3.1 void AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator() (unsigned int *iNumber*) const

Actor Function (functor).

Definition at line 197 of file InventoryParserHelper.cpp.

References AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::LegStruct::_flightNumber, and AIRINV::FlightDateStruct::_itLeg.

22.179.4 Member Data Documentation

22.179.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

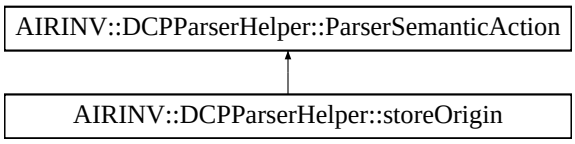
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.180 AIRINV::DCPParserHelper::storeOrigin Struct Reference

`#include <airinv/command/vault/DCPParserHelper.hpp>`

Inheritance diagram for AIRINV::DCPParserHelper::storeOrigin:



Public Member Functions

- storeOrigin (DCPRuleStruct &)
- void operator() (std::vector< char >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- DCPRuleStruct & _DCPRule

22.180.1 Detailed Description

Store the parsed origin.

Definition at line 48 of file DCPParserHelper.hpp.

22.180.2 Constructor & Destructor Documentation

22.180.2.1 AIRINV::DCPParserHelper::storeOrigin::storeOrigin (DCPRuleStruct & *ioDCPRule*)

Actor Constructor.

Definition at line 54 of file DCPParserHelper.cpp.

22.180.3 Member Function Documentation

22.180.3.1 void AIRINV::DCPParserHelper::storeOrigin::operator() (std::vector< char > *iChar*, boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 59 of file DCPParserHelper.cpp.

References AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule.

22.180.4 Member Data Documentation

22.180.4.1 DCPRuleStruct& AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule [inherited]

Actor Context.

Definition at line 34 of file DCPParserHelper.hpp.

Referenced by AIRINV::DCPParserHelper::storeDCPId::operator()(), operator()(), AIRINV::DCPParserHelper::storeDestination::operator()(), AIRINV::DCPParserHelper::storeDateRangeStart::operator()(), AIRINV::DCPParserHelper::storeDateRangeEnd::operator()(), AIRINV::DCPParserHelper::storeStartRangeTime::operator()(), AIRINV::DCPParserHelper::storeEndRangeTime::operator()(), AIRINV::DCPParserHelper::storePOS::operator()(), AIRINV::DCPParserHelper::storeCabinCode::operator()(), AIRINV::DCPParserHelper::storeChannel::operator()(), AIRINV::DCPParserHelper::storeAdvancePurchase::operator()(), AIRINV::DCPParserHelper::storeSaturdayStay::operator()(), AIRINV::DCPParserHelper::storeChangeFees::operator()(), AIRINV::DCPParserHelper::storeNonRefundable::operator()(), AIRINV::DCPParserHelper::storeMinimumStay::operator()(), AIRINV::DCPParserHelper-

::storeDCP::operator()(), AIRINV::DCPParserHelper::storeAirlineCode::operator()(), AIRINV::DCPParserHelper::storeClass::operator()(), and AIRINV::DCPParserHelper::doEndDCP::operator()().

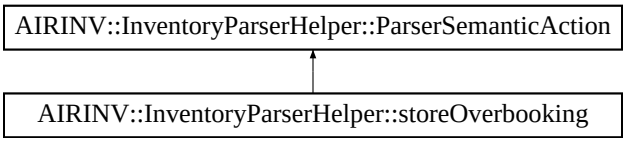
The documentation for this struct was generated from the following files:

- airinv/command/vault/DCPParserHelper.hpp
- airinv/command/vault/DCPParserHelper.cpp

22.181 AIRINV::InventoryParserHelper::storeOverbooking Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeOverbooking:



Public Member Functions

- storeOverbooking (FlightDateStruct &)
- void operator() (double iReal) const

Public Attributes

- FlightDateStruct & _flightDate

22.181.1 Detailed Description

Store the parsed Overbooking percentage (at booking class level).

Definition at line 341 of file InventoryParserHelper.hpp.

22.181.2 Constructor & Destructor Documentation

22.181.2.1 AIRINV::InventoryParserHelper::storeOverbooking::storeOverbooking (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 623 of file InventoryParserHelper.cpp.

22.181.3 Member Function Documentation

22.181.3.1 void AIRINV::InventoryParserHelper::storeOverbooking::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 628 of file InventoryParserHelper.cpp.

References AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::FlightDateStruct::_itBookingClass, and AIRINV::BookingClassStruct::_overbookingPercentage.

22.181.4 Member Data Documentation

22.181.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

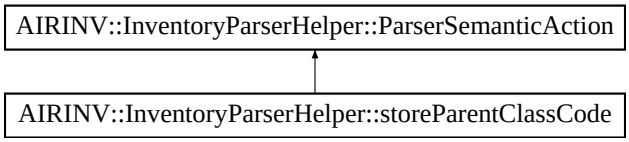
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.182 AIRINV::InventoryParserHelper::storeParentClassCode Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeParentClassCode:



Public Member Functions

- storeParentClassCode (FlightDateStruct &)
- void operator() (char iChar) const

Public Attributes

- FlightDateStruct & _flightDate

22.182.1 Detailed Description

Store the parsed class code of the parent sub-class.

Definition at line 293 of file InventoryParserHelper.hpp.

22.182.2 Constructor & Destructor Documentation

22.182.2.1 AIRINV::InventoryParserHelper::storeParentClassCode::storeParentClassCode (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 555 of file InventoryParserHelper.cpp.

22.182.3 Member Function Documentation

22.182.3.1 void AIRINV::InventoryParserHelper::storeParentClassCode::operator() (char *iChar*) const

Actor Function (functor).

Definition at line 560 of file InventoryParserHelper.cpp.

References AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::FlightDateStruct::_itBookingClass, and AIRINV::BookingClassStruct::_parentClassCode.

22.182.4 Member Data Documentation

22.182.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(),

AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

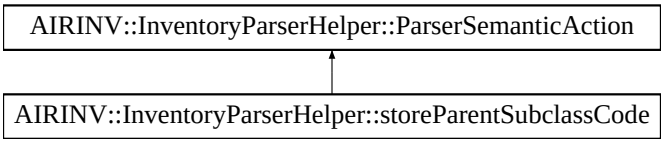
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.183 AIRINV::InventoryParserHelper::storeParentSubclassCode Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeParentSubclassCode:



Public Member Functions

- storeParentSubclassCode (FlightDateStruct &)
- void operator() (unsigned int iNumber) const

Public Attributes

- FlightDateStruct & _flightDate

22.183.1 Detailed Description

Store the parsed sub-class code of the parent sub-class.

Definition at line 301 of file InventoryParserHelper.hpp.

22.183.2 Constructor & Destructor Documentation

22.183.2.1 AIRINV::InventoryParserHelper::storeParentSubclassCode::storeParentSubclassCode (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 567 of file InventoryParserHelper.cpp.

22.183.3 Member Function Documentation

22.183.3.1 void AIRINV::InventoryParserHelper::storeParentSubclassCode::operator() (unsigned int *iNumber*) const

Actor Function (functor).

Definition at line 572 of file InventoryParserHelper.cpp.

References AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::FlightDateStruct::_itBookingClass, and AIRINV::BookingClassStruct::_parentSubclassCode.

22.183.4 Member Data Documentation

22.183.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

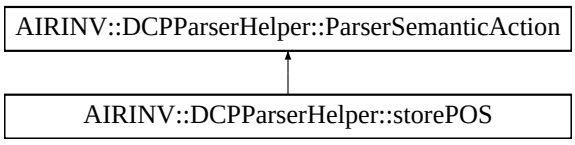
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.184 AIRINV::DCPParserHelper::storePOS Struct Reference

`#include <airinv/command/vault/DCPParserHelper.hpp>`

Inheritance diagram for AIRINV::DCPParserHelper::storePOS:



Public Member Functions

- storePOS (DCPRuleStruct &)
- void operator() (std::vector< char >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- DCPRuleStruct & _DCPRule

22.184.1 Detailed Description

Store the parsed customer position.

Definition at line 108 of file DCPParserHelper.hpp.

22.184.2 Constructor & Destructor Documentation

22.184.2.1 AIRINV::DCPParserHelper::storePOS::storePOS (DCPRuleStruct & *ioDCPRule*)

Actor Constructor.

Definition at line 150 of file DCPParserHelper.cpp.

22.184.3 Member Function Documentation

22.184.3.1 void AIRINV::DCPParserHelper::storePOS::operator() (std::vector< char > *iChar,* boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 155 of file DCPParserHelper.cpp.

References AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule.

22.184.4 Member Data Documentation

22.184.4.1 DCPRuleStruct& AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule [inherited]

Actor Context.

Definition at line 34 of file DCPParserHelper.hpp.

Referenced by AIRINV::DCPParserHelper::storeDCPId::operator()(), AIRINV::DCPParserHelper::storeOrigin::operator()(), AIRINV::DCPParserHelper::storeDestination::operator()(), AIRINV::DCPParserHelper::storeDateRangeStart::operator()(), AIRINV::DCPParserHelper::storeDateRangeEnd::operator()(), AIRINV::DCPParser-

Helper::storeStartRangeTime::operator()(), AIRINV::DCPParserHelper::storeEndRangeTime::operator()(), operator()(), AIRINV::DCPParserHelper::storeCabinCode::operator()(), AIRINV::DCPParserHelper::storeChannel-::operator()(), AIRINV::DCPParserHelper::storeAdvancePurchase::operator()(), AIRINV::DCPParserHelper::store-SaturdayStay::operator()(), AIRINV::DCPParserHelper::storeChangeFees::operator()(), AIRINV::DCPParser-Helper::storeNonRefundable::operator()(), AIRINV::DCPParserHelper::storeMinimumStay::operator()(), AIRINV-::DCPParserHelper::storeDCP::operator()(), AIRINV::DCPParserHelper::storeAirlineCode::operator()(), AIRINV::D-CPParserHelper::storeClass::operator()(), and AIRINV::DCPParserHelper::doEndDCP::operator()().

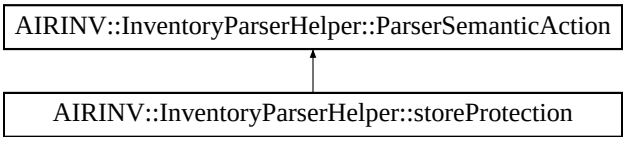
The documentation for this struct was generated from the following files:

- airinv/command/vault/DCPParserHelper.hpp
- airinv/command/vault/DCPParserHelper.cpp

22.185 AIRINV::InventoryParserHelper::storeProtection Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeProtection:



Public Member Functions

- storeProtection (FlightDateStruct &)
- void operator() (double iReal) const

Public Attributes

- FlightDateStruct & _flightDate

22.185.1 Detailed Description

Store the parsed protection (at booking class level).

Definition at line 317 of file InventoryParserHelper.hpp.

22.185.2 Constructor & Destructor Documentation

22.185.2.1 AIRINV::InventoryParserHelper::storeProtection::storeProtection (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 590 of file InventoryParserHelper.cpp.

22.185.3 Member Function Documentation

22.185.3.1 void AIRINV::InventoryParserHelper::storeProtection::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 595 of file InventoryParserHelper.cpp.

References AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::FlightDateStruct::_itBookingClass, and AIRINV::BookingClassStruct::_protection.

22.185.4 Member Data Documentation

22.185.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

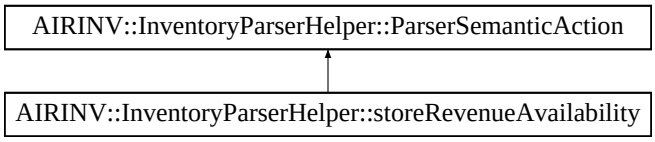
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.186 AIRINV::InventoryParserHelper::storeRevenueAvailability Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeRevenueAvailability:



Public Member Functions

- storeRevenueAvailability (FlightDateStruct &)
- void operator() (double iReal) const

Public Attributes

- FlightDateStruct & _flightDate

22.186.1 Detailed Description

Store the parsed number of net revenue availability (at booking class level).

Definition at line 417 of file InventoryParserHelper.hpp.

22.186.2 Constructor & Destructor Documentation

22.186.2.1 AIRINV::InventoryParserHelper::storeRevenueAvailability::storeRevenueAvailability (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 726 of file InventoryParserHelper.cpp.

22.186.3 Member Function Documentation

22.186.3.1 void AIRINV::InventoryParserHelper::storeRevenueAvailability::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 731 of file InventoryParserHelper.cpp.

References AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::FlightDateStruct::_itBookingClass, and AIRINV::BookingClassStruct::_netRevenueAvailability.

22.186.4 Member Data Documentation

22.186.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate-

::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

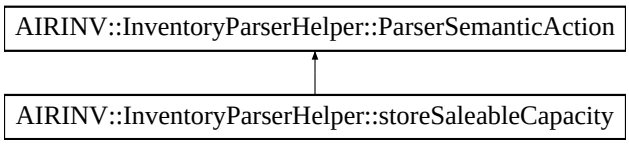
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.187 AIRINV::InventoryParserHelper::storeSaleableCapacity Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeSaleableCapacity:



Public Member Functions

- storeSaleableCapacity (FlightDateStruct &)
- void operator() (double iReal) const

Public Attributes

- FlightDateStruct & _flightDate

22.187.1 Detailed Description

Store the parsed saleable capacity.

Definition at line 157 of file InventoryParserHelper.hpp.

22.187.2 Constructor & Destructor Documentation

22.187.2.1 AIRINV::InventoryParserHelper::storeSaleableCapacity::storeSaleableCapacity (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 284 of file InventoryParserHelper.cpp.

22.187.3 Member Function Documentation

22.187.3.1 void AIRINV::InventoryParserHelper::storeSaleableCapacity::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 289 of file InventoryParserHelper.cpp.

References AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::FlightDateStruct::_itLegCabin, and AIRINV::LegCabinStruct::_saleableCapacity.

22.187.4 Member Data Documentation

22.187.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

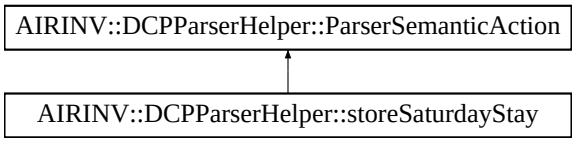
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.188 AIRINV::DCPParserHelper::storeSaturdayStay Struct Reference

`#include <airinv/command/vault/DCPParserHelper.hpp>`

Inheritance diagram for AIRINV::DCPParserHelper::storeSaturdayStay:



Public Member Functions

- storeSaturdayStay (DCPRuleStruct &)
- void operator() (char, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- DCPRuleStruct & _DCPRule

22.188.1 Detailed Description

Store the parsed saturday night.

Definition at line 148 of file DCPParserHelper.hpp.

22.188.2 Constructor & Destructor Documentation

22.188.2.1 AIRINV::DCPParserHelper::storeSaturdayStay::storeSaturdayStay (DCPRuleStruct & *ioDCPRule*)

Actor Constructor.

Definition at line 223 of file DCPParserHelper.cpp.

22.188.3 Member Function Documentation

22.188.3.1 void AIRINV::DCPParserHelper::storeSaturdayStay::operator() (char *iSaturdayStay,* boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 228 of file DCPParserHelper.cpp.

References AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule.

22.188.4 Member Data Documentation

22.188.4.1 DCPRuleStruct& AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule [inherited]

Actor Context.

Definition at line 34 of file DCPParserHelper.hpp.

Referenced by AIRINV::DCPParserHelper::storeDCPId::operator()(), AIRINV::DCPParserHelper::storeOrigin::operator()(), AIRINV::DCPParserHelper::storeDestination::operator()(), AIRINV::DCPParserHelper::storeDateRangeStart::operator()(), AIRINV::DCPParserHelper::storeDateRangeEnd::operator()(), AIRINV::DCPParserHelper::storeStartRangeTime::operator()(), AIRINV::DCPParserHelper::storeEndRangeTime::operator()(), AIRINV::DCPParserHelper::storePOS::operator()(), AIRINV::DCPParserHelper::storeCabinCode::operator()(), AIRINV::DCPParserHelper::storeChannel::operator()(), AIRINV::DCPParserHelper::storeAdvancePurchase::operator()(), operator()(), AIRINV::DCPParserHelper::storeChangeFees::operator()(), AIRINV::DCPParserHelper::storeNonRefundable::operator()(), AIRINV::DCPParserHelper::storeMinimumStay::operator()(), AIRINV::DCPParserHelper::storeDCP::operator()(), AIRINV::DCPParserHelper::storeAirlineCode::operator()(), AIRINV::DCPParserHelper::storeClass::operator()(), and AIRINV::DCPParserHelper::doEndDCP::operator()().

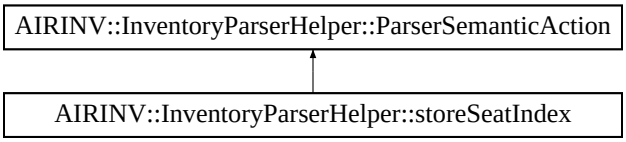
The documentation for this struct was generated from the following files:

- airinv/command/vault/DCPParserHelper.hpp
- airinv/command/vault/DCPParserHelper.cpp

22.189 AIRINV::InventoryParserHelper::storeSeatIndex Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeSeatIndex:



Public Member Functions

- storeSeatIndex (FlightDateStruct &)
- void operator() (double iReal) const

Public Attributes

- FlightDateStruct & _flightDate

22.189.1 Detailed Description

Store the parsed leg-cabin seat index.

Definition at line 237 of file InventoryParserHelper.hpp.

22.189.2 Constructor & Destructor Documentation

22.189.2.1 AIRINV::InventoryParserHelper::storeSeatIndex::storeSeatIndex (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 403 of file InventoryParserHelper.cpp.

22.189.3 Member Function Documentation

22.189.3.1 void AIRINV::InventoryParserHelper::storeSeatIndex::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 408 of file InventoryParserHelper.cpp.

References AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::FlightDateStruct::_itBucket, and AIRINV::BucketStruct::_seatIndex.

22.189.4 Member Data Documentation

22.189.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

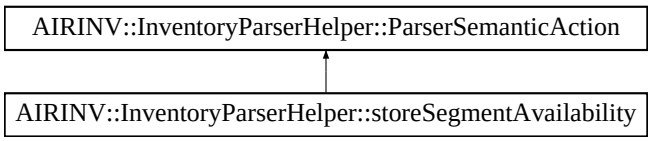
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.190 AIRINV::InventoryParserHelper::storeSegmentAvailability Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeSegmentAvailability:



Public Member Functions

- storeSegmentAvailability (FlightDateStruct &)
- void operator() (double iReal) const

Public Attributes

- FlightDateStruct & _flightDate

22.190.1 Detailed Description

Store the parsed number of segment availability (at booking class level).

Definition at line 408 of file InventoryParserHelper.hpp.

22.190.2 Constructor & Destructor Documentation

22.190.2.1 AIRINV::InventoryParserHelper::storeSegmentAvailability::storeSegmentAvailability (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 714 of file InventoryParserHelper.cpp.

22.190.3 Member Function Documentation

22.190.3.1 void AIRINV::InventoryParserHelper::storeSegmentAvailability::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 719 of file InventoryParserHelper.cpp.

References AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::FlightDateStruct::_itBookingClass, and AIRINV::BookingClassStruct::_segmentAvailability.

22.190.4 Member Data Documentation

22.190.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::store-

OperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

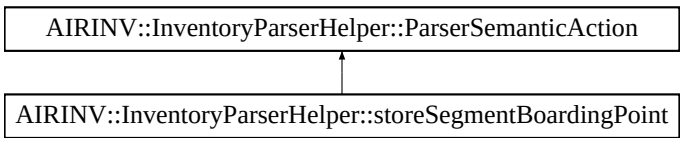
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.191 AIRINV::InventoryParserHelper::storeSegmentBoardingPoint Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeSegmentBoardingPoint:



Public Member Functions

- storeSegmentBoardingPoint (FlightDateStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightDateStruct & _flightDate

22.191.1 Detailed Description

Store the parsed segment boarding point.

Definition at line 245 of file InventoryParserHelper.hpp.

22.191.2 Constructor & Destructor Documentation

22.191.2.1 AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::storeSegmentBoardingPoint (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 415 of file InventoryParserHelper.cpp.

22.191.3 Member Function Documentation

22.191.3.1 void AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 420 of file InventoryParserHelper.cpp.

References AIRINV::SegmentStruct::_boardingPoint, AIRINV::LegCabinStruct::_bucketList, AIRINV::LegCabinStruct::_cabinCode, AIRINV::SegmentStruct::_cabinList, AIRINV::LegStruct::_cabinList, AIRINV::BookingClassStruct::_classCode, AIRINV::FareFamilyStruct::_classList, AIRINV::SegmentCabinStruct::_fareFamilies, AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::FlightDateStruct::_itBookingClass, AIRINV::FlightDateStruct::_itBucket, AIRINV::SegmentCabinStruct::_itFareFamily, AIRINV::FlightDateStruct::_itLeg, AIRINV::FlightDateStruct::_itLegCabin, AIRINV::FlightDateStruct::_itSegment, AIRINV::FlightDateStruct::_itSegmentCabin, AIRINV::FlightDateStruct::_legList, and AIRINV::FlightDateStruct::_segmentList.

22.191.4 Member Data Documentation

22.191.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParser-

Helper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

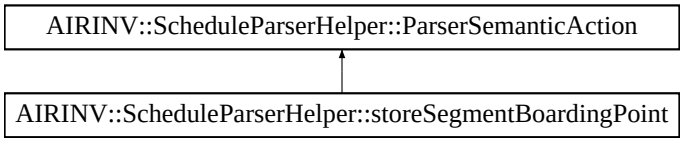
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.192 AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint Struct Reference

`#include <airinv/command/ScheduleParserHelper.hpp>`

Inheritance diagram for AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint:



Public Member Functions

- storeSegmentBoardingPoint (FlightPeriodStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

22.192.1 Detailed Description

Store the parsed segment boarding point.

Definition at line 160 of file ScheduleParserHelper.hpp.

22.192.2 Constructor & Destructor Documentation

22.192.2.1 AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::storeSegmentBoardingPoint (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 307 of file ScheduleParserHelper.cpp.

22.192.3 Member Function Documentation

22.192.3.1 void AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 312 of file ScheduleParserHelper.cpp.

References AIRINV::SegmentStruct::_boardingPoint, AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod, and AIRINV::FlightPeriodStruct::_itSegment.

22.192.4 Member Data Documentation

22.192.4.1 FlightPeriodStruct& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRINV::ScheduleParserHelper::storeAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRINV::ScheduleParserHelper::storeDow::operator()(), AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeBoardingTime::operator()(), AIRINV::ScheduleParserHelper::storeOffTime::operator()(), AIRINV::ScheduleParserHelper::storeElapsedTime::operator()(), AIRINV::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeCapacity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator()(), operator()(), AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeClasses::operator()(), AIRINV::ScheduleParserHelper::storeFamilyCode::operator()(), AIRINV::ScheduleParserHelper::storeFRAT5CurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFFDisutilityCurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFClasses::operator()(), and AIRINV::ScheduleParserHelper::doEndFlight::operator()().

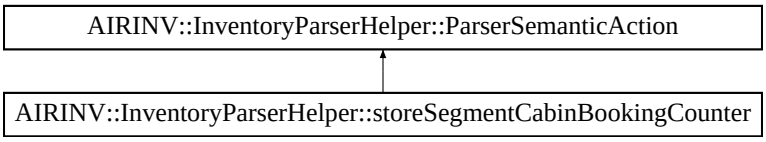
The documentation for this struct was generated from the following files:

- airinv/command/ScheduleParserHelper.hpp
- airinv/command/ScheduleParserHelper.cpp

22.193 AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter:



Public Member Functions

- storeSegmentCabinBookingCounter (FlightDateStruct &)
- void operator() (double iReal) const

Public Attributes

- FlightDateStruct & _flightDate

22.193.1 Detailed Description

Store the parsed segment cabin number of bookings.

Definition at line 269 of file InventoryParserHelper.hpp.

22.193.2 Constructor & Destructor Documentation

22.193.2.1 AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::storeSegmentCabinBookingCounter (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 513 of file InventoryParserHelper.cpp.

22.193.3 Member Function Documentation

22.193.3.1 void AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 518 of file InventoryParserHelper.cpp.

References AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::FlightDateStruct::_itSegmentCabin, and AIRINV::SegmentCabinStruct::_nbOfBookings.

22.193.4 Member Data Documentation

22.193.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

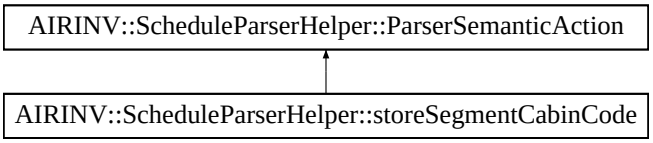
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.194 AIRINV::ScheduleParserHelper::storeSegmentCabinCode Struct Reference

`#include <airinv/command/ScheduleParserHelper.hpp>`

Inheritance diagram for AIRINV::ScheduleParserHelper::storeSegmentCabinCode:



Public Member Functions

- storeSegmentCabinCode (FlightPeriodStruct &)
- void operator() (char iChar) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

22.194.1 Detailed Description

Store the parsed segment cabin code.

Definition at line 176 of file ScheduleParserHelper.hpp.

22.194.2 Constructor & Destructor Documentation

22.194.2.1 AIRINV::ScheduleParserHelper::storeSegmentCabinCode::storeSegmentCabinCode (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 333 of file ScheduleParserHelper.cpp.

22.194.3 Member Function Documentation

22.194.3.1 void AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator() (char *iChar*) const

Actor Function (functor).

Definition at line 338 of file ScheduleParserHelper.cpp.

References AIRINV::SegmentCabinStruct::_cabinCode, AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod, and AIRINV::FlightPeriodStruct::_itSegmentCabin.

22.194.4 Member Data Documentation

22.194.4.1 FlightPeriodStruct& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRINV::ScheduleParserHelper::storeAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRINV::ScheduleParserHelper::storeDow::operator()(), AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeBoardingTime::operator()(), AIRINV::ScheduleParserHelper::storeOffTime::operator()(), AIRINV::ScheduleParserHelper::storeElapsedTime::operator()(), AIRINV::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeCapacity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator()(), operator()(), AIRINV::ScheduleParserHelper::storeClasses::operator()(), AIRINV::ScheduleParserHelper::storeFamilyCode::operator()(), AIRINV::ScheduleParserHelper::storeFRAT5CurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFFDisutilityCurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFClasses::operator()(), and AIRINV::ScheduleParserHelper::doEndFlight::operator()().

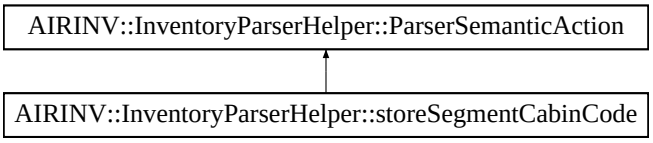
The documentation for this struct was generated from the following files:

- airinv/command/ScheduleParserHelper.hpp
- airinv/command/ScheduleParserHelper.cpp

22.195 AIRINV::InventoryParserHelper::storeSegmentCabinCode Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeSegmentCabinCode:



Public Member Functions

- storeSegmentCabinCode (FlightDateStruct &)
- void operator() (char iChar) const

Public Attributes

- FlightDateStruct & _flightDate

22.195.1 Detailed Description

Store the parsed segment cabin code.

Definition at line 261 of file InventoryParserHelper.hpp.

22.195.2 Constructor & Destructor Documentation

22.195.2.1 AIRINV::InventoryParserHelper::storeSegmentCabinCode::storeSegmentCabinCode (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 478 of file InventoryParserHelper.cpp.

22.195.3 Member Function Documentation

22.195.3.1 void AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator() (char *iChar*) const

Actor Function (functor).

Definition at line 483 of file InventoryParserHelper.cpp.

References AIRINV::SegmentCabinStruct::_cabinCode, AIRINV::SegmentStruct::_cabinList, AIRINV::BookingClassStruct::_classCode, AIRINV::FareFamilyStruct::_classList, AIRINV::SegmentCabinStruct::_fareFamilies, AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::FlightDateStruct::_itBookingClass, AIRINV::SegmentCabinStruct::_itFareFamily, AIRINV::FlightDateStruct::_itSegment, and AIRINV::FlightDateStruct::_itSegmentCabin.

22.195.4 Member Data Documentation

22.195.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

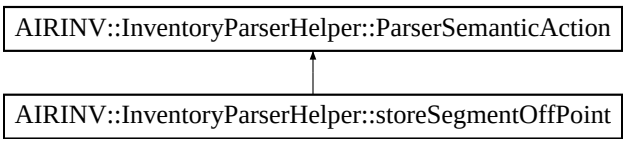
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.196 AIRINV::InventoryParserHelper::storeSegmentOffPoint Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeSegmentOffPoint:



Public Member Functions

- storeSegmentOffPoint (FlightDateStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightDateStruct & _flightDate

22.196.1 Detailed Description

Store the parsed segment off point.

Definition at line 253 of file InventoryParserHelper.hpp.

22.196.2 Constructor & Destructor Documentation

22.196.2.1 AIRINV::InventoryParserHelper::storeSegmentOffPoint::storeSegmentOffPoint (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 464 of file InventoryParserHelper.cpp.

22.196.3 Member Function Documentation

22.196.3.1 void AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 469 of file InventoryParserHelper.cpp.

References AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::FlightDateStruct::_itSegment, and AIRINV::SegmentStruct::_offPoint.

22.196.4 Member Data Documentation

22.196.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode-

::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

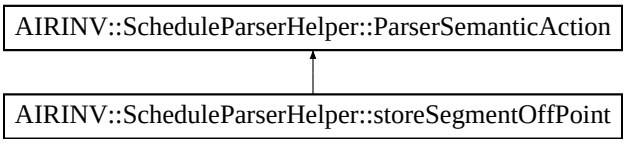
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.197 AIRINV::ScheduleParserHelper::storeSegmentOffPoint Struct Reference

`#include <airinv/command/ScheduleParserHelper.hpp>`

Inheritance diagram for AIRINV::ScheduleParserHelper::storeSegmentOffPoint:



Public Member Functions

- storeSegmentOffPoint (FlightPeriodStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

22.197.1 Detailed Description

Store the parsed segment off point.

Definition at line 168 of file ScheduleParserHelper.hpp.

22.197.2 Constructor & Destructor Documentation

22.197.2.1 AIRINV::ScheduleParserHelper::storeSegmentOffPoint::storeSegmentOffPoint (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 320 of file ScheduleParserHelper.cpp.

22.197.3 Member Function Documentation

22.197.3.1 void AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 325 of file ScheduleParserHelper.cpp.

References AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod, AIRINV::FlightPeriodStruct::_itSegment, and AIRINV::SegmentStruct::_offPoint.

22.197.4 Member Data Documentation

22.197.4.1 FlightPeriodStruct& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRINV::ScheduleParserHelper::storeAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRINV::ScheduleParserHelper::storeDow::operator()(), AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeBoardingTime::operator()(), AIRINV::ScheduleParserHelper::storeOffTime::operator()(), AIRINV::ScheduleParserHelper::storeElapsedTime::operator()(), AIRINV::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeCapacity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator()(), AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), operator()(), AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeClasses::operator()(), AIRINV::ScheduleParserHelper::storeFamilyCode::operator()(), AIRINV::ScheduleParserHelper::storeFRAT5CurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFFDisutilityCurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFClasses::operator()(), and AIRINV::ScheduleParserHelper::doEndFlight::operator()().

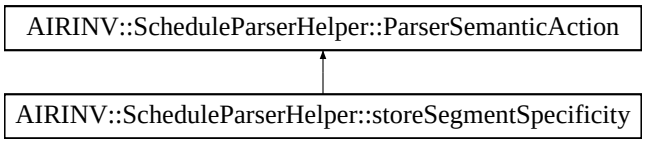
The documentation for this struct was generated from the following files:

- airinv/command/ScheduleParserHelper.hpp
- airinv/command/ScheduleParserHelper.cpp

22.198 AIRINV::ScheduleParserHelper::storeSegmentSpecificity Struct Reference

`#include <airinv/command/ScheduleParserHelper.hpp>`

Inheritance diagram for AIRINV::ScheduleParserHelper::storeSegmentSpecificity:



Public Member Functions

- storeSegmentSpecificity (FlightPeriodStruct &)
- void operator() (char iChar) const

Public Attributes

- FlightPeriodStruct & _flightPeriod

22.198.1 Detailed Description

Store whether or not the segment definitions are specific. Specific means that there is a definition for each segment. General (not specific) means that a single definition defines all the segments.

Definition at line 152 of file ScheduleParserHelper.hpp.

22.198.2 Constructor & Destructor Documentation

22.198.2.1 AIRINV::ScheduleParserHelper::storeSegmentSpecificity::storeSegmentSpecificity (FlightPeriodStruct & *ioFlightPeriod*)

Actor Constructor.

Definition at line 281 of file ScheduleParserHelper.cpp.

22.198.3 Member Function Documentation

22.198.3.1 void AIRINV::ScheduleParserHelper::storeSegmentSpecificity::operator() (char *iChar*) const

Actor Function (functor).

Definition at line 286 of file ScheduleParserHelper.cpp.

References AIRINV::FlightPeriodStruct::_airportList, AIRINV::FlightPeriodStruct::_airportOrderedList, AIRINV::FlightPeriodStruct::_areSegmentDefinitionsSpecific, AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod, and AIRINV::FlightPeriodStruct::buildSegments().

22.198.4 Member Data Documentation

22.198.4.1 FlightPeriodStruct& AIRINV::ScheduleParserHelper::ParserSemanticAction::_flightPeriod [inherited]

Actor Context.

Definition at line 33 of file ScheduleParserHelper.hpp.

Referenced by AIRINV::ScheduleParserHelper::storeAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeStart::operator()(), AIRINV::ScheduleParserHelper::storeDateRangeEnd::operator()(), AIRINV::ScheduleParserHelper::storeDow::operator()(),

AIRINV::ScheduleParserHelper::storeLegBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeLegOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::ScheduleParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::ScheduleParserHelper::storeBoardingTime::operator()(), AIRINV::ScheduleParserHelper::storeOffTime::operator()(), AIRINV::ScheduleParserHelper::storeElapsedTime::operator()(), AIRINV::ScheduleParserHelper::storeLegCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeCapacity::operator()(), operator()(), AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentOffPoint::operator()(), AIRINV::ScheduleParserHelper::storeSegmentCabinCode::operator()(), AIRINV::ScheduleParserHelper::storeClasses::operator()(), AIRINV::ScheduleParserHelper::storeFamilyCode::operator()(), AIRINV::ScheduleParserHelper::storeFRAT5CurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFFDisutilityCurveKey::operator()(), AIRINV::ScheduleParserHelper::storeFClasses::operator()(), and AIRINV::ScheduleParserHelper::doEndFlight::operator()().

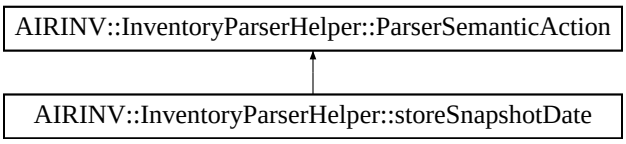
The documentation for this struct was generated from the following files:

- airinv/command/ScheduleParserHelper.hpp
- airinv/command/ScheduleParserHelper.cpp

22.199 AIRINV::InventoryParserHelper::storeSnapshotDate Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeSnapshotDate:



Public Member Functions

- storeSnapshotDate (FlightDateStruct &)
- void operator() (iterator_t iStr, iterator_t iStrEnd) const

Public Attributes

- FlightDateStruct & _flightDate

22.199.1 Detailed Description

Store the snapshot date.

Definition at line 37 of file InventoryParserHelper.hpp.

22.199.2 Constructor & Destructor Documentation

22.199.2.1 AIRINV::InventoryParserHelper::storeSnapshotDate::storeSnapshotDate (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 32 of file InventoryParserHelper.cpp.

22.199.3 Member Function Documentation

22.199.3.1 void AIRINV::InventoryParserHelper::storeSnapshotDate::operator() (iterator_t *iStr,* iterator_t *iStrEnd*) const

Actor Function (functor).

Definition at line 37 of file InventoryParserHelper.cpp.

References AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::FlightDateStruct::_flightDate, and AIRINV::FlightDateStruct::getDate().

22.199.4 Member Data Documentation

22.199.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

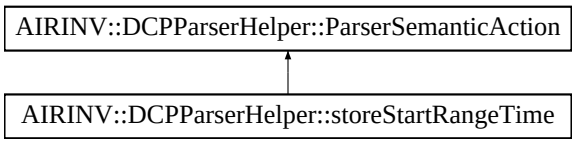
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.200 AIRINV::DCPParserHelper::storeStartRangeTime Struct Reference

`#include <airinv/command/vault/DCPParserHelper.hpp>`

Inheritance diagram for AIRINV::DCPParserHelper::storeStartRangeTime:



Public Member Functions

- storeStartRangeTime (DCPRuleStruct &)
- void operator() (boost::spirit::qi::unused_type, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- DCPRuleStruct & _DCPRule

22.200.1 Detailed Description

Store the parsed start range time.

Definition at line 88 of file DCPParserHelper.hpp.

22.200.2 Constructor & Destructor Documentation

22.200.2.1 AIRINV::DCPParserHelper::storeStartRangeTime::storeStartRangeTime (DCPRuleStruct & *ioDCPRule*)

Actor Constructor.

Definition at line 116 of file DCPParserHelper.cpp.

22.200.3 Member Function Documentation

22.200.3.1 void AIRINV::DCPParserHelper::storeStartRangeTime::operator() (boost::spirit::qi::unused_type , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 121 of file DCPParserHelper.cpp.

References AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule.

22.200.4 Member Data Documentation

22.200.4.1 DCPRuleStruct& AIRINV::DCPParserHelper::ParserSemanticAction::_DCPRule [inherited]

Actor Context.

Definition at line 34 of file DCPParserHelper.hpp.

Referenced by AIRINV::DCPParserHelper::storeDCPId::operator()(), AIRINV::DCPParserHelper::storeOrigin::operator()(), AIRINV::DCPParserHelper::storeDestination::operator()(), AIRINV::DCPParserHelper::storeDateRangeStart::operator()(), AIRINV::DCPParserHelper::storeDateRangeEnd::operator()(), operator()(), AIRINV::DCPParserHelper::storeEndRangeTime::operator()(), AIRINV::DCPParserHelper::storePOS::operator()(), AIRINV::DCPParserHelper::storeCabinCode::operator()(), AIRINV::DCPParserHelper::storeChannel::operator()(),

AIRINV::DCPParserHelper::storeAdvancePurchase::operator()(), AIRINV::DCPParserHelper::storeSaturdayStay-::operator()(), AIRINV::DCPParserHelper::storeChangeFees::operator()(), AIRINV::DCPParserHelper::storeNon-Refundable::operator()(), AIRINV::DCPParserHelper::storeMinimumStay::operator()(), AIRINV::DCPParserHelper-::storeDCP::operator()(), AIRINV::DCPParserHelper::storeAirlineCode::operator()(), AIRINV::DCPParserHelper-::storeClass::operator()(), and AIRINV::DCPParserHelper::doEndDCP::operator()().

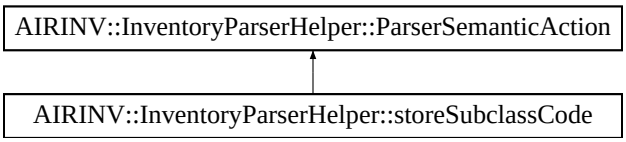
The documentation for this struct was generated from the following files:

- airinv/command/vault/DCPParserHelper.hpp
- airinv/command/vault/DCPParserHelper.cpp

22.201 AIRINV::InventoryParserHelper::storeSubclassCode Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeSubclassCode:



Public Member Functions

- storeSubclassCode (FlightDateStruct &)
- void operator() (unsigned int iNumber) const

Public Attributes

- FlightDateStruct & _flightDate

22.201.1 Detailed Description

Store the parsed sub-class code.

Definition at line 285 of file InventoryParserHelper.hpp.

22.201.2 Constructor & Destructor Documentation

22.201.2.1 AIRINV::InventoryParserHelper::storeSubclassCode::storeSubclassCode (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 543 of file InventoryParserHelper.cpp.

22.201.3 Member Function Documentation

22.201.3.1 void AIRINV::InventoryParserHelper::storeSubclassCode::operator() (unsigned int *iNumber*) const

Actor Function (functor).

Definition at line 548 of file InventoryParserHelper.cpp.

References AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::FlightDateStruct::_it-BookingClass, and AIRINV::BookingClassStruct::_subclassCode.

22.201.4 Member Data Documentation

22.201.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

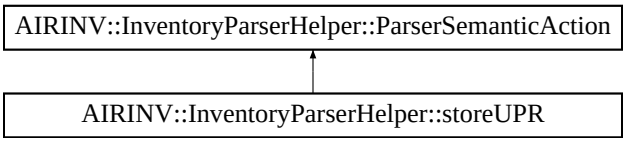
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.202 AIRINV::InventoryParserHelper::storeUPR Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeUPR:



Public Member Functions

- storeUPR (FlightDateStruct &)

- void operator() (double iReal) const

Public Attributes

- FlightDateStruct & _flightDate

22.202.1 Detailed Description

Store the parsed Unsold Protected (UPR).

Definition at line 173 of file InventoryParserHelper.hpp.

22.202.2 Constructor & Destructor Documentation

22.202.2.1 AIRINV::InventoryParserHelper::storeUPR::storeUPR (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 306 of file InventoryParserHelper.cpp.

22.202.3 Member Function Documentation

22.202.3.1 void AIRINV::InventoryParserHelper::storeUPR::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 311 of file InventoryParserHelper.cpp.

References AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::FlightDateStruct::_itLegCabin, and AIRINV::LegCabinStruct::_upr.

22.202.4 Member Data Documentation

22.202.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), AIRINV::InventoryParserHelper::storeYieldUpperRange::operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper-

::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

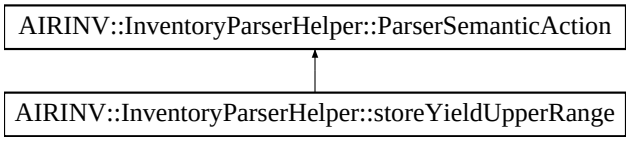
The documentation for this struct was generated from the following files:

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.203 AIRINV::InventoryParserHelper::storeYieldUpperRange Struct Reference

`#include <airinv/command/InventoryParserHelper.hpp>`

Inheritance diagram for AIRINV::InventoryParserHelper::storeYieldUpperRange:



Public Member Functions

- storeYieldUpperRange (FlightDateStruct &)
- void operator() (double iReal) const

Public Attributes

- FlightDateStruct & _flightDate

22.203.1 Detailed Description

Store the parsed Yield Upper Range value.

Definition at line 221 of file InventoryParserHelper.hpp.

22.203.2 Constructor & Destructor Documentation

22.203.2.1 AIRINV::InventoryParserHelper::storeYieldUpperRange::storeYieldUpperRange (FlightDateStruct & *ioFlightDate*)

Actor Constructor.

Definition at line 372 of file InventoryParserHelper.cpp.

22.203.3 Member Function Documentation

22.203.3.1 void AIRINV::InventoryParserHelper::storeYieldUpperRange::operator() (double *iReal*) const

Actor Function (functor).

Definition at line 377 of file InventoryParserHelper.cpp.

References AIRINV::LegCabinStruct::_bucketList, AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate, AIRINV::FlightDateStruct::_itBucket, AIRINV::FlightDateStruct::_itLegCabin, and AIRINV::BucketStruct::_yieldRangeUpperValue.

22.203.4 Member Data Documentation

22.203.4.1 FlightDateStruct& AIRINV::InventoryParserHelper::ParserSemanticAction::_flightDate [inherited]

Actor Context.

Definition at line 33 of file InventoryParserHelper.hpp.

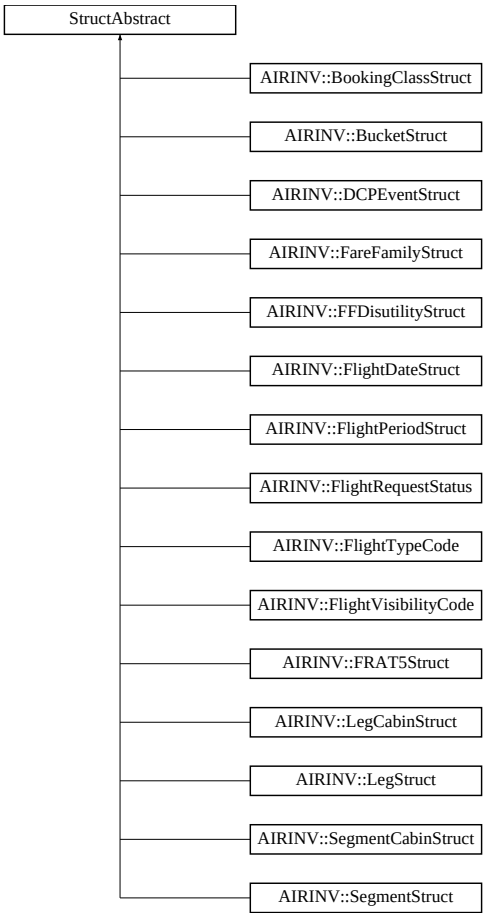
Referenced by AIRINV::InventoryParserHelper::storeSnapshotDate::operator()(), AIRINV::InventoryParserHelper::storeAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeFlightDate::operator()(), AIRINV::InventoryParserHelper::storeFlightTypeCode::operator()(), AIRINV::InventoryParserHelper::storeFlightVisibilityCode::operator()(), AIRINV::InventoryParserHelper::storeLegBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeLegOffPoint::operator()(), AIRINV::InventoryParserHelper::storeOperatingAirlineCode::operator()(), AIRINV::InventoryParserHelper::storeOperatingFlightNumber::operator()(), AIRINV::InventoryParserHelper::storeBoardingDate::operator()(), AIRINV::InventoryParserHelper::storeBoardingTime::operator()(), AIRINV::InventoryParserHelper::storeOffDate::operator()(), AIRINV::InventoryParserHelper::storeOffTime::operator()(), AIRINV::InventoryParserHelper::storeLegCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSaleableCapacity::operator()(), AIRINV::InventoryParserHelper::storeAU::operator()(), AIRINV::InventoryParserHelper::storeUPR::operator()(), AIRINV::InventoryParserHelper::storeBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeNAV::operator()(), AIRINV::InventoryParserHelper::storeGAV::operator()(), AIRINV::InventoryParserHelper::storeACP::operator()(), AIRINV::InventoryParserHelper::storeETB::operator()(), operator()(), AIRINV::InventoryParserHelper::storeBucketAvaibality::operator()(), AIRINV::InventoryParserHelper::storeSeatIndex::operator()(), AIRINV::InventoryParserHelper::storeSegmentBoardingPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentOffPoint::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinCode::operator()(), AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter::operator()(), AIRINV::InventoryParserHelper::storeClassCode::operator()(), AIRINV::InventoryParserHelper::storeSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeParentClassCode::operator()(), AIRINV::InventoryParserHelper::storeParentSubclassCode::operator()(), AIRINV::InventoryParserHelper::storeCumulatedProtection::operator()(), AIRINV::InventoryParserHelper::storeProtection::operator()(), AIRINV::InventoryParserHelper::storeNego::operator()(), AIRINV::InventoryParserHelper::storeNoShow::operator()(), AIRINV::InventoryParserHelper::storeOverbooking::operator()(), AIRINV::InventoryParserHelper::storeNbOfBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfStaffBkgs::operator()(), AIRINV::InventoryParserHelper::storeNbOfWLBkgs::operator()(), AIRINV::InventoryParserHelper::storeClassETB::operator()(), AIRINV::InventoryParserHelper::storeClassAvailability::operator()(), AIRINV::InventoryParserHelper::storeSegmentAvailability::operator()(), AIRINV::InventoryParserHelper::storeRevenueAvailability::operator()(), AIRINV::InventoryParserHelper::storeFamilyCode::operator()(), AIRINV::InventoryParserHelper::storeFClasses::operator()(), and AIRINV::InventoryParserHelper::doEndFlightDate::operator()().

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

- airinv/command/InventoryParserHelper.hpp
- airinv/command/InventoryParserHelper.cpp

22.204 StructAbstract Class Reference

Inheritance diagram for StructAbstract:

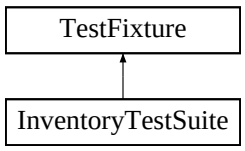


The documentation for this class was generated from the following file:

- airinv/bom/SegmentStruct.hpp

22.205 TestFixture Class Reference

Inheritance diagram for TestFixture:



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

- test/airinv/InventoryTestSuite.hpp

23 File Documentation

23.1 airinv/AIRINV_Master_Service.hpp File Reference

```
#include <string>
```

```
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_file.hpp>
#include <stdair/stdair_service_types.hpp>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/bom/BomIDTypes.hpp>
#include <airrac/AIRRAC_Types.hpp>
#include <sevmgr/SEVMGR_Types.hpp>
#include <airinv/AIRINV_Types.hpp>
```

Classes

- class AIRINV::AIRINV_Master_Service

  Interface for the AIRINV Services.

Namespaces

- namespace stdair

  Forward declarations.
- namespace AIRINV

23.2 AIRINV_Master_Service.hpp

```
00001 #ifndef __AIRINV_SVC_AIRINV_MASTER_SERVICE_HPP
00002 #define __AIRINV_SVC_AIRINV_MASTER_SERVICE_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_file.hpp>
00012 #include <stdair/stdair_service_types.hpp>
00013 #include <stdair/stdair_inventory_types.hpp>
00014 #include <stdair/stdair_maths_types.hpp>
00015 #include <stdair/bom/BomIDTypes.hpp>
00016 // AirRAC
00017 #include <airrac/AIRRAC_Types.hpp>
00018 // SEvMgr
00019 #include <sevmgr/SEVMGR_Types.hpp>
00020 // AirInv
00021 #include <airinv/AIRINV_Types.hpp>
00022
00023
00025 namespace stdair {
00026   class BomRoot;
00027   class AirlineFeatureSet;
00028   class Inventory;
00029   class JSONString;
00030   class STDAIR_Service;
00031   struct BasLogParams;
00032   struct BasDBParams;
00033   struct SnapshotStruct;
00034   struct RMEventStruct;
00035   struct TravelSolutionStruct;
00036 }
00037
00038 namespace AIRINV {
00039
00041   class AIRINV_Master_ServiceContext;
00042
00043
00047   class AIRINV_Master_Service {
00048   public:
00049     // ////////// Constructors and destructors //////////
00065     AIRINV_Master_Service (const stdair::BasLogParams&,
00066                            const stdair::BasDBParams&);
00067
```

```
00079     AIRINV_Master_Service (const stdair::BasLogParams&);
00080
00096     AIRINV_Master_Service (stdair::STDAIR_ServicePtr_T);
00097
00114     AIRINV_Master_Service (stdair::STDAIR_ServicePtr_T,
00115                            SEVMGR::SEVMGR_ServicePtr_T);
00116
00125     void parseAndLoad (const InventoryFilePath&);
00126
00139     void parseAndLoad (const stdair::ScheduleFilePath&,
00140                        const stdair::ODFilePath&,
00141                        const stdair::FRAT5FilePath&,
00142                        const stdair::FFDisutilityFilePath&,
00143                        const AIRRAC::YieldFilePath&);
00144
00148     ~AIRINV_Master_Service();
00149
00154     void initSnapshotAndRMEvents (const stdair::Date_T&,
      const stdair::Date_T&);
00155
00156
00157   public:
00158     // ////////// Business Methods /////////////
00166     void buildSampleBom();
00167
00171     void clonePersistentBom ();
00172
00177     void buildComplementaryLinks (stdair::BomRoot&);
00178
00182     void calculateAvailability (
      stdair::TravelSolutionStruct&);
00183
00192     bool sell (const std::string& iSegmentDateKey, const
      stdair::ClassCode_T&,
00193                const stdair::PartySize_T&);
00194
00202     bool sell (const stdair::BookingClassID_T&, const stdair::PartySize_T&)
      ;
00203
00213     bool cancel (const std::string& iSegmentDateKey, const
      stdair::ClassCode_T&,
00214                  const stdair::PartySize_T&);
00215
00223     bool cancel (const stdair::BookingClassID_T&, const
      stdair::PartySize_T&);
00224
00228     void takeSnapshots (const stdair::SnapshotStruct&);
00229
00233     void optimise (const stdair::RMEventStruct&);
00234
00235
00236   public:
00237
00238     // ///////////////// Export support methods /////////////////
00248     std::string jsonHandler (const stdair::JSONString&) const;
00249
00262     std::string jsonExportFlightDateList (const
      stdair::AirlineCode_T& iAirlineCode = "all",
00263                                           const stdair::FlightNumber_T&
      iFlightNumber = 0) const;
00264
00275     std::string jsonExportFlightDateObjects (const
      stdair::AirlineCode_T&,
00276                                              const stdair::FlightNumber_T&,
00277                                              const stdair::Date_T&
      iDepartureDate) const;
00278
00279
00280   public:
00281     // ///////////////// Display support methods /////////////////
00295     std::string list (const stdair::AirlineCode_T& iAirlineCode = "all",
00296                       const stdair::FlightNumber_T& iFlightNumber = 0) const;
00297
00307     bool check (const stdair::AirlineCode_T&, const stdair::FlightNumber_T
      &,
00308                 const stdair::Date_T& iDepartureDate) const;
00309
00317     std::string csvDisplay() const;
00318
00330     std::string csvDisplay (const stdair::AirlineCode_T&,
00331                             const stdair::FlightNumber_T&,
00332                             const stdair::Date_T& iDepartureDate) const;
00333
00334
00335   private:
00336     // /////// Construction and Destruction helper methods ///////
00340     AIRINV_Master_Service();
```

```
00341
00345     AIRINV_Master_Service (const AIRINV_Master_Service
    &);
00346
00356     stdair::STDAIR_ServicePtr_T initStdAirService (const stdair::BasLogParams&,
00357                                                    const stdair::BasDBParams&);
00358
00367     stdair::STDAIR_ServicePtr_T initStdAirService (const stdair::BasLogParams&)
    ;
00368
00377     void addStdAirService (stdair::STDAIR_ServicePtr_T,
00378                            const bool iOwnStdairService);
00379
00385     void addSEVMGRService (SEVMGR::SEVMGR_ServicePtr_T ioSEVMGR_ServicePtr,
00386                            const bool iOwnSEVMGRService);
00387
00388
00393     void initServiceContext();
00394
00401     void initSlaveAirinvService();
00402
00406     void finalise();
00407
00408
00409   private:
00410     // ///////// Service Context /////////
00414     AIRINV_Master_ServiceContext*
    _airinvMasterServiceContext;
00415   };
00416 }
00417 #endif // __AIRINV_SVC_AIRINV_MASTER_SERVICE_HPP
```

23.3 airinv/AIRINV_Service.hpp File Reference

```
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_service_types.hpp>
#include <stdair/basic/JSonCommand.hpp>
#include <stdair/bom/RMEventTypes.hpp>
#include <stdair/bom/BomIDTypes.hpp>
#include <airrac/AIRRAC_Types.hpp>
#include <sevmgr/SEVMGR_Types.hpp>
```

Classes

- class AIRINV::AIRINV_Service

  Interface for the AIRINV Services.

Namespaces

- namespace stdair

  Forward declarations.
- namespace AIRINV

23.4 AIRINV_Service.hpp

```
00001 #ifndef __AIRINV_SVC_AIRINV_SERVICE_HPP
00002 #define __AIRINV_SVC_AIRINV_SERVICE_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_service_types.hpp>
00012 #include <stdair/basic/JSonCommand.hpp>
```

```
00013 #include <stdair/bom/RMEventTypes.hpp>
00014 #include <stdair/bom/BomIDTypes.hpp>
00015 // AirRAC
00016 #include <airrac/AIRRAC_Types.hpp>
00017 // SEvMgr
00018 #include <sevmgr/SEVMGR_Types.hpp>
00019
00021 namespace stdair {
00022   class AirlineFeatureSet;
00023   class STDAIR_Service;
00024   class Inventory;
00025   class JSONString;
00026   struct TravelSolutionStruct;
00027   struct BasLogParams;
00028   struct BasDBParams;
00029 }
00030
00031 namespace AIRINV {
00032
00034   class AIRINV_ServiceContext;
00035
00036
00040   class AIRINV_Service {
00041   public:
00042     // ////////// Constructors and destructors //////////
00058     AIRINV_Service (const stdair::BasLogParams&, const
      stdair::BasDBParams&);
00059
00071     AIRINV_Service (const stdair::BasLogParams&);
00072
00088     AIRINV_Service (stdair::STDAIR_ServicePtr_T);
00089
00106     AIRINV_Service (stdair::STDAIR_ServicePtr_T,
00107                     SEVMGR::SEVMGR_ServicePtr_T);
00108
00117     void parseAndLoad (const AIRINV::InventoryFilePath
      &);
00118
00131     void parseAndLoad (const stdair::ScheduleFilePath&,
00132                        const stdair::ODFilePath&,
00133                        const stdair::FRAT5FilePath&,
00134                        const stdair::FFDisutilityFilePath&,
00135                        const AIRRAC::YieldFilePath&);
00136
00140     ~AIRINV_Service();
00141
00142
00143   public:
00144     // ////////// Business Methods /////////////
00152     void buildSampleBom();
00153
00157     void clonePersistentBom ();
00158
00164     void buildComplementaryLinks (stdair::BomRoot&);
00165
00171     stdair::RMEventList_T initRMEvents (const stdair::Date_T&
      iStartDate,
00172                                         const stdair::Date_T& iEndDate);
00173
00177     void calculateAvailability (
      stdair::TravelSolutionStruct&);
00178
00187     bool sell (const std::string& iSegmentDateKey, const
      stdair::ClassCode_T&,
00188                const stdair::PartySize_T&);
00189
00197     bool sell (const stdair::BookingClassID_T&, const stdair::PartySize_T&)
      ;
00198
00208     bool cancel (const std::string& iSegmentDateKey, const
      stdair::ClassCode_T&,
00209                  const stdair::PartySize_T&);
00210
00218     bool cancel (const stdair::BookingClassID_T&, const
      stdair::PartySize_T&);
00219
00223     void takeSnapshots (const stdair::AirlineCode_T&,
00224                         const stdair::DateTime_T&);
00225
00229     void optimise (const stdair::AirlineCode_T&,
00230                    const stdair::KeyDescription_T&,
00231                    const stdair::DateTime_T&);
00232
00233   public:
00234     // ////////////////// Export support methods /////////////////
00235
00245     std::string jsonHandler (const stdair::JSONString&) const;
```

```
00246
00259     std::string jsonExportFlightDateList (const
      stdair::AirlineCode_T& iAirlineCode = "all",
00260                                           const stdair::FlightNumber_T&
      iFlightNumber = 0) const;
00271     std::string jsonExportFlightDateObjects (const
      stdair::AirlineCode_T&,
00272                                              const stdair::FlightNumber_T&,
00273                                              const stdair::Date_T&
      iDepartureDate) const;
00274
00275   public:
00276     // ///////////////// Display support methods /////////////////
00290     std::string list (const stdair::AirlineCode_T& iAirlineCode = "all",
00291                       const stdair::FlightNumber_T& iFlightNumber = 0) const;
00292
00302     bool check (const stdair::AirlineCode_T&, const stdair::FlightNumber_T
      &,
00303                 const stdair::Date_T& iDepartureDate) const;
00304
00312     std::string csvDisplay() const;
00313
00325     std::string csvDisplay (const stdair::AirlineCode_T&,
00326                             const stdair::FlightNumber_T&,
00327                             const stdair::Date_T& iDepartureDate) const;
00328
00329
00330   private:
00331     // /////// Construction and Destruction helper methods ///////
00335     AIRINV_Service ();
00336
00340     AIRINV_Service (const AIRINV_Service&);
00341
00351     stdair::STDAIR_ServicePtr_T initStdAirService (const stdair::BasLogParams&,
00352                                                    const stdair::BasDBParams&);
00353
00362     stdair::STDAIR_ServicePtr_T initStdAirService (const stdair::BasLogParams&)
      ;
00363
00367     void initRMOLService();
00368
00372     void initAIRRACService();
00373
00377     void initSEVMGRService();
00378
00387     void addStdAirService (stdair::STDAIR_ServicePtr_T,
00388                            const bool iOwnStdairService);
00389
00395     void addSEVMGRService (SEVMGR::SEVMGR_ServicePtr_T,
00396                            const bool iOwnSEVMGRService);
00397
00402     void initServiceContext();
00403
00410     void initAirinvService();
00411
00415     void finalise();
00416
00417
00418   private:
00419     // ///////// Service Context /////////
00423     AIRINV_ServiceContext* _airinvServiceContext;
00424   };
00425 }
00426 #endif // __AIRINV_SVC_AIRINV_SERVICE_HPP
```

23.5 airinv/AIRINV_Types.hpp File Reference

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

Classes

- class AIRINV::InventoryFileParsingFailedException
- class AIRINV::ScheduleFileParsingFailedException

- class AIRINV::MissingPartnerFlightDateWithinScheduleFile
- class AIRINV::FRAT5FileParsingFailedException
- class AIRINV::FFDisutilityFileParsingFailedException
- class AIRINV::SegmentDateNotFoundException
- class AIRINV::InventoryInputFileNotFoundException
- class AIRINV::ScheduleInputFileNotFoundException
- class AIRINV::FRAT5InputFileNotFoundException
- class AIRINV::FFDisutilityInputFileNotFoundException
- class AIRINV::FlightDateDuplicationException
- class AIRINV::BookingException
- class AIRINV::InventoryNotFoundException
- class AIRINV::FlightDateNotFoundException
- class AIRINV::InventoryFilePath

Namespaces

- namespace AIRINV

Typedefs

- typedef boost::shared_ptr < AIRINV_Service > AIRINV::AIRINV_ServicePtr_T
- typedef boost::shared_ptr < AIRINV_Master_Service > AIRINV::AIRINV_Master_ServicePtr_T
- typedef std::map< const stdair::AirlineCode_T, AIRINV_ServicePtr_T > AIRINV::AIRINV_ServicePtr_Map_T
- typedef std::map< const stdair::DTD_T, double > AIRINV::FRAT5Curve_T

23.6 AIRINV_Types.hpp

```
00001 #ifndef __AIRINV_AIRINV_TYPES_HPP
00002 #define __AIRINV_AIRINV_TYPES_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <map>
00009 // Boost
00010 #include <boost/shared_ptr.hpp>
00011 // StdAir
00012 #include <stdair/stdair_exceptions.hpp>
00013 #include <stdair/stdair_file.hpp>
00014 #include <stdair/stdair_inventory_types.hpp>
00015
00016 namespace AIRINV {
00017
00018   // Forward declarations
00019   class AIRINV_Service;
00020   class AIRINV_Master_Service;
00021
00022
00023   // ///////// Exceptions //////////
00024
00028   class InventoryFileParsingFailedException
00029     : public stdair::ParsingFileFailedException {
00030   public:
00034     InventoryFileParsingFailedException (
      const std::string& iWhat)
00035       : stdair::ParsingFileFailedException (iWhat) {}
00036   };
00037
00041   class ScheduleFileParsingFailedException
00042     : public stdair::ParsingFileFailedException {
00043   public:
```

```
00047     ScheduleFileParsingFailedException (const
       std::string& iWhat)
00048       : stdair::ParsingFileFailedException (iWhat) {}
00049   };
00050
00054   class MissingPartnerFlightDateWithinScheduleFile
00055     : public ScheduleFileParsingFailedException
       {
00056   public:
00060     MissingPartnerFlightDateWithinScheduleFile
       (const std::string& iWhat)
00061       : ScheduleFileParsingFailedException (
      iWhat) {}
00062   };
00063
00067   class FRAT5FileParsingFailedException
00068     : public stdair::ParsingFileFailedException {
00069   public:
00073     FRAT5FileParsingFailedException (const
      std::string& iWhat)
00074       : stdair::ParsingFileFailedException (iWhat) {}
00075   };
00076
00080   class FFDisutilityFileParsingFailedException
00081     : public stdair::ParsingFileFailedException {
00082   public:
00086     FFDisutilityFileParsingFailedException
       (const std::string& iWhat)
00087       : stdair::ParsingFileFailedException (iWhat) {}
00088   };
00089
00094   class SegmentDateNotFoundException : public
      stdair::ParserException {
00095   public:
00099     SegmentDateNotFoundException (const std::string
      & iWhat)
00100       : stdair::ParserException (iWhat) {}
00101   };
00102
00106   class InventoryInputFileNotFoundException
       : public stdair::FileNotFoundException {
00107   public:
00111     InventoryInputFileNotFoundException (
      const std::string& iWhat)
00112       : stdair::FileNotFoundException (iWhat) {}
00113   };
00114
00118   class ScheduleInputFileNotFoundException :
      public stdair::FileNotFoundException {
00119   public:
00123     ScheduleInputFileNotFoundException (const
       std::string& iWhat)
00124       : stdair::FileNotFoundException (iWhat) {}
00125   };
00126
00130   class FRAT5InputFileNotFoundException : public
       stdair::FileNotFoundException {
00131   public:
00135     FRAT5InputFileNotFoundException (const
      std::string& iWhat)
00136       : stdair::FileNotFoundException (iWhat) {}
00137   };
00138
00142   class FFDisutilityInputFileNotFoundException
       : public stdair::FileNotFoundException {
00143   public:
00147     FFDisutilityInputFileNotFoundException
       (const std::string& iWhat)
00148       : stdair::FileNotFoundException (iWhat) {}
00149   };
00150
00154   class FlightDateDuplicationException : public
      stdair::ObjectCreationgDuplicationException {
00155   public:
00159     FlightDateDuplicationException (const
      std::string& iWhat)
00160       : stdair::ObjectCreationgDuplicationException
       (iWhat) {}
00161   };
00162
00166   class BookingException : public stdair::RootException {
00167   };
00168
00172   class InventoryNotFoundException : public
      stdair::ObjectNotFoundException {
00173   public:
00177     InventoryNotFoundException (const std::string&
```

```
      iWhat)
00178       : stdair::ObjectNotFoundException (iWhat) {}
00179   };
00180
00184   class FlightDateNotFoundException : public
      stdair::ObjectNotFoundException {
00185   public:
00189     FlightDateNotFoundException (const std::string&
      iWhat)
00190       : stdair::ObjectNotFoundException (iWhat) {}
00191   };
00192
00193
00194   // //////// Type definitions /////////
00198   class InventoryFilePath : public stdair::InputFilePath {
00199   public:
00203     explicit InventoryFilePath (const stdair::Filename_T&
      iFilename)
00204       : stdair::InputFilePath (iFilename) {}
00205   };
00206
00210   typedef boost::shared_ptr<AIRINV_Service> AIRINV_ServicePtr_T
      ;
00211
00215   typedef boost::shared_ptr<AIRINV_Master_Service> AIRINV_Master_ServicePtr_T
      ;
00216
00221   typedef std::map<const stdair::AirlineCode_T,
00222                    AIRINV_ServicePtr_T>
      AIRINV_ServicePtr_Map_T;
00223
00227   typedef std::map<const stdair::DTD_T, double> FRAT5Curve_T;
00228
00229 }
00230 #endif // __AIRINV_AIRINV_TYPES_HPP
```

23.7 airinv/basic/BasConst.cpp File Reference

```
#include <airinv/basic/BasConst_General.hpp>
#include <airinv/basic/BasConst_Curves.hpp>
#include <airinv/basic/BasConst_AIRINV_Service.hpp>
```

Namespaces

- namespace AIRINV

Variables

- const std::string AIRINV::DEFAULT_AIRLINE_CODE = "BA"
- const FRAT5Curve_T AIRINV::DEFAULT_PICKUP_FRAT5_CURVE

23.8 BasConst.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 #include <airinv/basic/BasConst_General.hpp>
00005 #include <airinv/basic/BasConst_Curves.hpp>
00006 #include <airinv/basic/BasConst_AIRINV_Service.hpp
      >
00007
00008 namespace AIRINV {
00009
00011   const std::string DEFAULT_AIRLINE_CODE = "BA";
00012
00014   const FRAT5Curve_T DEFAULT_PICKUP_FRAT5_CURVE
       =
00015     DefaultMap::createPickupFRAT5Curve();
00016   FRAT5Curve_T DefaultMap::createPickupFRAT5Curve
      () {
00017     FRAT5Curve_T oCurve;
00018     oCurve[365] = 1.1; oCurve[63] = 1.2;
```

```
00019     oCurve[49] = 1.3;  oCurve[35] = 1.4;
00020     oCurve[23] = 1.7;  oCurve[16] = 2.0;
00021     oCurve[10] = 2.3;  oCurve[5]  = 2.44;
00022     oCurve[1]  = 2.5;  oCurve[0]  = 2.5;
00023
00024     return oCurve;
00025   };
00026
00027 }
```

23.9 airinv/basic/BasConst_AIRINV_Service.hpp File Reference

```
#include <string>
```

Namespaces

- namespace AIRINV

23.10 BasConst_AIRINV_Service.hpp

```
00001 #ifndef __AIRINV_BAS_BASCONST_AIRINV_SERVICE_HPP
00002 #define __AIRINV_BAS_BASCONST_AIRINV_SERVICE_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 #include <string>
00008
00009 namespace AIRINV {
00010
00012   extern const std::string DEFAULT_AIRLINE_CODE;
00013
00014 }
00015 #endif // __AIRINV_BAS_BASCONST_AIRINV_SERVICE_HPP
```

23.11 airinv/basic/BasConst_Curves.hpp File Reference

```
#include <airinv/AIRINV_Types.hpp>
```

Classes

- struct AIRINV::DefaultMap

Namespaces

- namespace AIRINV

23.12 BasConst_Curves.hpp

```
00001 #ifndef __AIRINV_BAS_BASCONST_CURVES_HPP
00002 #define __AIRINV_BAS_BASCONST_CURVES_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // AIRINV
00008 #include <airinv/AIRINV_Types.hpp>
00009
00010 namespace AIRINV {
00011
00013   extern const FRAT5Curve_T DEFAULT_PICKUP_FRAT5_CURVE
      ;
```

```
00014
00016   struct DefaultMap {
00017     static FRAT5Curve_T createPickupFRAT5Curve
      ();
00018   };
00019 }
00020 #endif // __AIRINV_BAS_BASCONST_CURVES_HPP
```

23.13 airinv/basic/BasConst_General.hpp File Reference

Namespaces

- namespace AIRINV

23.14 BasConst_General.hpp

```
00001 #ifndef __AIRINV_BAS_BASCONST_GENERAL_HPP
00002 #define __AIRINV_BAS_BASCONST_GENERAL_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007
00008 namespace AIRINV {
00009
00010 }
00011 #endif // __AIRINV_BAS_BASCONST_GENERAL_HPP
```

23.15 airinv/basic/BasParserTypes.hpp File Reference

```
#include <string>
#include <boost/spirit/home/classic/core.hpp>
#include <boost/spirit/home/classic/attribute.hpp>
#include <boost/spirit/home/classic/utility/functor_parser.hpp>
#include <boost/spirit/home/classic/utility/loops.hpp>
#include <boost/spirit/home/classic/utility/chset.hpp>
#include <boost/spirit/home/classic/utility/confix.hpp>
#include <boost/spirit/home/classic/iterator/file_iterator.hpp>
#include <boost/spirit/home/classic/actor/push_back_actor.hpp>
#include <boost/spirit/home/classic/actor/assign_actor.hpp>
```

Namespaces

- namespace AIRINV

Typedefs

- typedef char AIRINV::char_t
- typedef boost::spirit::classic::file_iterator < char_t > AIRINV::iterator_t
- typedef boost::spirit::classic::scanner < iterator_t > AIRINV::scanner_t
- typedef boost::spirit::classic::rule < scanner_t > AIRINV::rule_t

- typedef boost::spirit::classic::int_parser < unsigned int, 10, 1, 1 > AIRINV::int1_p_t
- typedef boost::spirit::classic::uint_parser < unsigned int, 10, 2, 2 > AIRINV::uint2_p_t
- typedef boost::spirit::classic::uint_parser < unsigned int, 10, 1, 2 > AIRINV::uint1_2_p_t
- typedef boost::spirit::classic::uint_parser < unsigned int, 10, 1, 3 > AIRINV::uint1_3_p_t
- typedef boost::spirit::classic::uint_parser < unsigned int, 10, 4, 4 > AIRINV::uint4_p_t
- typedef boost::spirit::classic::uint_parser < unsigned int, 10, 1, 4 > AIRINV::uint1_4_p_t
- typedef boost::spirit::classic::chset < char_t > AIRINV::chset_t
- typedef boost::spirit::classic::impl::loop_traits < chset_t, unsigned int, unsigned int >::type AIRINV::repeat_p_t
- typedef boost::spirit::classic::bounded < uint2_p_t, unsigned int > AIRINV::bounded2_p_t
- typedef boost::spirit::classic::bounded < uint1_2_p_t, unsigned int > AIRINV::bounded1_2_p_t
- typedef boost::spirit::classic::bounded < uint1_3_p_t, unsigned int > AIRINV::bounded1_3_p_t
- typedef boost::spirit::classic::bounded < uint4_p_t, unsigned int > AIRINV::bounded4_p_t
- typedef boost::spirit::classic::bounded < uint1_4_p_t, unsigned int > AIRINV::bounded1_4_p_t

23.16 BasParserTypes.hpp

```
00001 #ifndef __AIRINV_BAS_BASCOMPARSERTYPES_HPP
00002 #define __AIRINV_BAS_BASCOMPARSERTYPES_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // Boost
00010 //#define BOOST_SPIRIT_DEBUG
00011 #include <boost/spirit/home/classic/core.hpp>
00012 #include <boost/spirit/home/classic/attribute.hpp>
00013 #include <boost/spirit/home/classic/utility/functor_parser.hpp>
00014 #include <boost/spirit/home/classic/utility/loops.hpp>
00015 #include <boost/spirit/home/classic/utility/chset.hpp>
00016 #include <boost/spirit/home/classic/utility/confix.hpp>
00017 #include <boost/spirit/home/classic/iterator/file_iterator.hpp>
00018 #include <boost/spirit/home/classic/actor/push_back_actor.hpp>
00019 #include <boost/spirit/home/classic/actor/assign_actor.hpp>
00020
```

```
00021 namespace AIRINV {
00022
00023   // ////////////////////////////////////////////////////////////////////
00024   //
00025   //  Definition of Basic Types
00026   //
00027   // ////////////////////////////////////////////////////////////////////
00028   // For a file, the parsing unit is the character (char). For a string,
00029   // it is a "char const *".
00030   // typedef char const* iterator_t;
00031   typedef char char_t;
00032
00033   // The types of iterator, scanner and rule are then derived from
00034   // the parsing unit.
00035   typedef boost::spirit::classic::file_iterator<char_t> iterator_t;
00036   typedef boost::spirit::classic::scanner<iterator_t> scanner_t;
00037   typedef boost::spirit::classic::rule<scanner_t> rule_t;
00038
00039   // ////////////////////////////////////////////////////////////////////
00040   //
00041   //  Parser related types
00042   //
00043   // ////////////////////////////////////////////////////////////////////
00045   typedef boost::spirit::classic::int_parser<unsigned int, 10, 1, 1> int1_p_t
      ;
00046
00048   typedef boost::spirit::classic::uint_parser<unsigned int, 10, 2, 2> uint2_p_t
      ;
00049
00051   typedef boost::spirit::classic::uint_parser<unsigned int, 10, 1, 2>
      uint1_2_p_t;
00052
00054   typedef boost::spirit::classic::uint_parser<unsigned int, 10, 1, 3>
      uint1_3_p_t;
00055
00057   typedef boost::spirit::classic::uint_parser<unsigned int, 10, 4, 4> uint4_p_t
      ;
00058
00060   typedef boost::spirit::classic::uint_parser<unsigned int, 10, 1, 4>
      uint1_4_p_t;
00061
00063   typedef boost::spirit::classic::chset<char_t> chset_t;
00064
00067   typedef boost::spirit::classic::impl::loop_traits<chset_t,
00068                                                     unsigned int,
00069                                                     unsigned int>::type repeat_p_t
      ;
00070
00072   typedef boost::spirit::classic::bounded<uint2_p_t, unsigned int> bounded2_p_t
      ;
00073   typedef boost::spirit::classic::bounded<uint1_2_p_t, unsigned int>
      bounded1_2_p_t;
00074   typedef boost::spirit::classic::bounded<uint1_3_p_t, unsigned int>
      bounded1_3_p_t;
00075   typedef boost::spirit::classic::bounded<uint4_p_t, unsigned int> bounded4_p_t
      ;
00076   typedef boost::spirit::classic::bounded<uint1_4_p_t, unsigned int>
      bounded1_4_p_t;
00077 }
00078 #endif // __AIRINV_BAS_BASCOMPARSERTYPES_HPP
```

23.17 airinv/basic/FlightRequestStatus.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/service/Logger.hpp>
#include <airinv/AIRINV_Types.hpp>
#include <airinv/FlightRequestStatus.hpp>
```

Namespaces

- namespace AIRINV

23.18 FlightRequestStatus.cpp

```
00001 // //////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/service/Logger.hpp>
00009 // Airinv
00010 #include <airinv/AIRINV_Types.hpp>
00011 #include <airinv/FlightRequestStatus.hpp>
00012
00013 namespace AIRINV {
00014
00015   // //////////////////////////////////////////////////////////////////
00016   const std::string FlightRequestStatus::_labels[LAST_VALUE] =
00017     { "OK", "Not Found", "Internal Error"};
00018
00019   const std::string FlightRequestStatus::_codeLabels[LAST_VALUE] =
00020     { "OK", "NF", "IE" };
00021
00022
00023   // //////////////////////////////////////////////////////////////////
00024   FlightRequestStatus::
00025   FlightRequestStatus (const EN_FlightRequestStatus
      & iFlightRequestStatus)
00026     : _code (iFlightRequestStatus) {
00027   }
00028
00029   // //////////////////////////////////////////////////////////////////
00030   FlightRequestStatus::FlightRequestStatus
       (const std::string& iCode) {
00031     _code = LAST_VALUE;
00032
00033     if (iCode == "OK") {
00034       _code = OK;
00035
00036     } else if (iCode == "NF") {
00037       _code = NOT_FOUND;
00038
00039     } else if (iCode == "IE") {
00040       _code = INTERNAL_ERROR;
00041
00042     }
00043
00044     if (_code == LAST_VALUE) {
00045       const std::string& lLabels = describeLabels();
00046       STDAIR_LOG_ERROR ("The flight request status '" << iCode
00047                         << "' is not known. Known flight request status: "
00048                         << lLabels);
00049       throw stdair::CodeConversionException ("The flight request status '"
00050                                              + iCode
00051                                              + "' is not known. Known flight
       request status: "
00052                                              + lLabels);
00053     }
00054   }
00055
00056   // //////////////////////////////////////////////////////////////////
00057   const std::string& FlightRequestStatus::
00058   getLabel (const EN_FlightRequestStatus& iCode
      ) {
00059     return _labels[iCode];
00060   }
00061
00062   // //////////////////////////////////////////////////////////////////
00063   const std::string& FlightRequestStatus::
00064   getCodeLabel (const EN_FlightRequestStatus
      & iCode) {
00065     return _codeLabels[iCode];
00066   }
00067
00068   // //////////////////////////////////////////////////////////////////
00069   std::string FlightRequestStatus::describeLabels
      () {
00070     std::ostringstream ostr;
00071     for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
00072       if (idx != 0) {
00073         ostr << ", ";
00074       }
00075       ostr << _labels[idx];
00076     }
00077     return ostr.str();
00078   }
00079
```

```
00080   // ////////////////////////////////////////////////////////////////////
00081   FlightRequestStatus::EN_FlightRequestStatus
      FlightRequestStatus::
00082   getCode() const {
00083     return _code;
00084   }
00085
00086   // ////////////////////////////////////////////////////////////////////
00087   const std::string FlightRequestStatus::describe(
      ) const {
00088     std::ostringstream ostr;
00089     ostr << _labels[_code];
00090     return ostr.str();
00091   }
00092
00093 }
```

23.19 airinv/basic/FlightTypeCode.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/service/Logger.hpp>
#include <airinv/AIRINV_Types.hpp>
#include <airinv/basic/FlightTypeCode.hpp>
```

Namespaces

- namespace AIRINV

23.20 FlightTypeCode.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/service/Logger.hpp>
00009 // Airinv
00010 #include <airinv/AIRINV_Types.hpp>
00011 #include <airinv/basic/FlightTypeCode.hpp>
00012
00013 namespace AIRINV {
00014
00015   // ////////////////////////////////////////////////////////////////////
00016   const std::string FlightTypeCode::_labels[LAST_VALUE] =
00017     { "Domestic", "International", "Ground Handling"};
00018
00019   const std::string FlightTypeCode::_codeLabels[LAST_VALUE] =
00020     { "DOM", "INT", "GRD" };
00021
00022
00023   // ////////////////////////////////////////////////////////////////////
00024   FlightTypeCode::FlightTypeCode (const
      EN_FlightTypeCode& iFlightTypeCode)
00025     : _code (iFlightTypeCode) {
00026   }
00027
00028   // ////////////////////////////////////////////////////////////////////
00029   FlightTypeCode::FlightTypeCode (const
      std::string& iCode) {
00030     _code = LAST_VALUE;
00031
00032     if (iCode == "DOM") {
00033       _code = DOMESTIC;
00034
00035     } else if (iCode == "INT") {
00036       _code = INTERNATIONAL;
00037
00038     } else if (iCode == "GRD") {
00039       _code = GROUND_HANDLING;
00040     }
00041
```

```
00042     if (_code == LAST_VALUE) {
00043       const std::string& lLabels = describeLabels();
00044       STDAIR_LOG_ERROR ("The flight type code '" << iCode
00045                         << "' is not known. Known flight type codes: "
00046                         << lLabels);
00047       throw stdair::CodeConversionException ("The flight type code '" + iCode
00048                                              + "' is not known. Known flight
      type codes: "
00049                                              + lLabels);
00050     }
00051   }
00052
00053   // ////////////////////////////////////////////////////////////////////
00054   const std::string& FlightTypeCode::getLabel (const
      EN_FlightTypeCode& iCode) {
00055     return _labels[iCode];
00056   }
00057
00058   // ////////////////////////////////////////////////////////////////////
00059   const std::string& FlightTypeCode::
00060   getCodeLabel (const EN_FlightTypeCode& iCode)
      {
00061     return _codeLabels[iCode];
00062   }
00063
00064   // ////////////////////////////////////////////////////////////////////
00065   std::string FlightTypeCode::describeLabels() {
00066     std::ostringstream ostr;
00067     for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
00068       if (idx != 0) {
00069         ostr << ", ";
00070       }
00071       ostr << _labels[idx];
00072     }
00073     return ostr.str();
00074   }
00075
00076   // ////////////////////////////////////////////////////////////////////
00077   FlightTypeCode::EN_FlightTypeCode
      FlightTypeCode::getCode() const {
00078     return _code;
00079   }
00080
00081   // ////////////////////////////////////////////////////////////////////
00082   const std::string FlightTypeCode::describe() const {
00083     std::ostringstream ostr;
00084     ostr << _labels[_code];
00085     return ostr.str();
00086   }
00087
00088 }
```

23.21 airinv/basic/FlightTypeCode.hpp File Reference

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

Classes

- struct AIRINV::FlightTypeCode

Namespaces

- namespace AIRINV

23.22 FlightTypeCode.hpp

```
00001 #ifndef __AIRINV_BAS_FLIGHTTYPECODE_HPP
00002 #define __AIRINV_BAS_FLIGHTTYPECODE_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
```

```
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/basic/StructAbstract.hpp>
00011
00012 namespace AIRINV {
00013
00015   struct FlightTypeCode : public stdair::StructAbstract {
00016   public:
00017     typedef enum {
00018       DOMESTIC = 0,
00019       INTERNATIONAL,
00020       GROUND_HANDLING,
00021       LAST_VALUE
00022     } EN_FlightTypeCode;
00023
00025     static const std::string& getLabel (const EN_FlightTypeCode
      &);
00026
00028     static const std::string& getCodeLabel (const EN_FlightTypeCode
      &);
00029
00031     static std::string describeLabels();
00032
00034     EN_FlightTypeCode getCode() const;
00035
00037     const std::string describe() const;
00038
00039
00040   public:
00042     FlightTypeCode (const EN_FlightTypeCode&);
00044     FlightTypeCode (const std::string& iCode);
00045
00046
00047   private:
00049     static const std::string _labels[LAST_VALUE];
00051     static const std::string _codeLabels[LAST_VALUE];
00052
00053
00054   private:
00055     // //////// Attributes /////////
00057     EN_FlightTypeCode _code;
00058   };
00059
00060 }
00061 #endif // __AIRINV_BAS_FLIGHTTYPECODE_HPP
```

23.23 airinv/basic/FlightVisibilityCode.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/service/Logger.hpp>
#include <airinv/AIRINV_Types.hpp>
#include <airinv/basic/FlightVisibilityCode.hpp>
```

Namespaces

- namespace AIRINV

23.24 FlightVisibilityCode.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/service/Logger.hpp>
00009 // Airinv
00010 #include <airinv/AIRINV_Types.hpp>
00011 #include <airinv/basic/FlightVisibilityCode.hpp
      >
00012
```

```
00013 namespace AIRINV {
00014
00015   // ////////////////////////////////////////////////////////////////////
00016   const std::string FlightVisibilityCode::_labels[LAST_VALUE] =
00017     { "Normal", "Hidden", "Pseudo"};
00018
00019   const std::string FlightVisibilityCode::_codeLabels[LAST_VALUE] =
00020     { "NOR", "HID", "PSD" };
00021
00022
00023   // ////////////////////////////////////////////////////////////////////
00024   FlightVisibilityCode::
00025   FlightVisibilityCode (const EN_FlightVisibilityCode
      & iFlightVisibilityCode)
00026     : _code (iFlightVisibilityCode) {
00027   }
00028
00029   // ////////////////////////////////////////////////////////////////////
00030   FlightVisibilityCode::FlightVisibilityCode
       (const std::string& iCode) {
00031     _code = LAST_VALUE;
00032
00033     if (iCode == "NOR") {
00034       _code = NORMAL;
00035
00036     } else if (iCode == "HID") {
00037       _code = HIDDEN;
00038
00039     } else if (iCode == "PSD") {
00040       _code = PSEUDO;
00041     }
00042
00043     if (_code == LAST_VALUE) {
00044       const std::string& lLabels = describeLabels();
00045       STDAIR_LOG_ERROR ("The flight visibility code '" << iCode
00046                         << "' is not known. Known flight visibility codes: "
00047                         << lLabels);
00048       throw stdair::CodeConversionException ("The flight visibility code '"
00049                                              + iCode
00050                                              + "' is not known. Known flight
       visibility codes: "
00051                                              + lLabels);
00052     }
00053   }
00054
00055   // ////////////////////////////////////////////////////////////////////
00056   const std::string& FlightVisibilityCode::
00057   getLabel (const EN_FlightVisibilityCode&
      iCode) {
00058     return _labels[iCode];
00059   }
00060
00061   // ////////////////////////////////////////////////////////////////////
00062   const std::string& FlightVisibilityCode::
00063   getCodeLabel (const EN_FlightVisibilityCode
      & iCode) {
00064     return _codeLabels[iCode];
00065   }
00066
00067   // ////////////////////////////////////////////////////////////////////
00068   std::string FlightVisibilityCode::describeLabels
      () {
00069     std::ostringstream ostr;
00070     for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
00071       if (idx != 0) {
00072         ostr << ", ";
00073       }
00074       ostr << _labels[idx];
00075     }
00076     return ostr.str();
00077   }
00078
00079   // ////////////////////////////////////////////////////////////////////
00080   FlightVisibilityCode::EN_FlightVisibilityCode
       FlightVisibilityCode::
00081   getCode() const {
00082     return _code;
00083   }
00084
00085   // ////////////////////////////////////////////////////////////////////
00086   const std::string FlightVisibilityCode::describe
      () const {
00087     std::ostringstream ostr;
00088     ostr << _labels[_code];
00089     return ostr.str();
00090   }
00091
```

```
00092 }
```

23.25 airinv/basic/FlightVisibilityCode.hpp File Reference

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

Classes

- struct AIRINV::FlightVisibilityCode

Namespaces

- namespace AIRINV

23.26 FlightVisibilityCode.hpp

```
00001 #ifndef __AIRINV_BAS_FLIGHTVISIBILITYCODE_HPP
00002 #define __AIRINV_BAS_FLIGHTVISIBILITYCODE_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/basic/StructAbstract.hpp>
00011
00012 namespace AIRINV {
00013
00015   struct FlightVisibilityCode : public
      stdair::StructAbstract {
00016   public:
00017     typedef enum {
00018       NORMAL = 0,
00019       HIDDEN,
00020       PSEUDO,
00021       LAST_VALUE
00022     } EN_FlightVisibilityCode;
00023
00025     static const std::string& getLabel (const EN_FlightVisibilityCode
      &);
00026
00028     static const std::string& getCodeLabel (const
      EN_FlightVisibilityCode&);
00029
00031     static std::string describeLabels();
00032
00034     EN_FlightVisibilityCode getCode() const;
00035
00037     const std::string describe() const;
00038
00039
00040   public:
00042     FlightVisibilityCode (const EN_FlightVisibilityCode
      &);
00044     FlightVisibilityCode (const std::string& iCode);
00045
00046
00047   private:
00049     static const std::string _labels[LAST_VALUE];
00051     static const std::string _codeLabels[LAST_VALUE];
00052
00053
00054   private:
00055     // //////// Attributes /////////
00057     EN_FlightVisibilityCode _code;
00058   };
00059
00060 }
00061 #endif // __AIRINV_BAS_FLIGHTVISIBILITYCODE_HPP
```

23.27 airinv/batches/airinv_parseInventory.cpp File Reference

23.28 airinv_parseInventory.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/program_options.hpp>
00013 #include <boost/tokenizer.hpp>
00014 // StdAir
00015 #include <stdair/basic/BasLogParams.hpp>
00016 #include <stdair/basic/BasDBParams.hpp>
00017 #include <stdair/service/Logger.hpp>
00018 // AirInv
00019 #include <airinv/AIRINV_Master_Service.hpp>
00020 #include <airinv/config/airinv-paths.hpp>
00021
00022 // //////// Constants //////
00026 const std::string K_AIRINV_DEFAULT_LOG_FILENAME ("airinv_parseInventory.log");
00027
00031 const std::string K_AIRINV_DEFAULT_INVENTORY_FILENAME (STDAIR_SAMPLE_DIR
00032                                                         "/invdump01.csv");
00036 const std::string K_AIRINV_DEFAULT_SCHEDULE_FILENAME (STDAIR_SAMPLE_DIR
00037                                                       "/schedule01.csv");
00041 const std::string K_AIRINV_DEFAULT_OND_FILENAME (STDAIR_SAMPLE_DIR
00042                                                  "/ond01.csv");
00046 const std::string K_AIRINV_DEFAULT_FRAT5_INPUT_FILENAME (STDAIR_SAMPLE_DIR
00047                                                       "/frat5.csv");
00051 const std::string K_AIRINV_DEFAULT_FF_DISUTILITY_INPUT_FILENAME (
      STDAIR_SAMPLE_DIR
00052                                                                  "
      /ffDisutility.csv");
00053
00057 const std::string K_AIRINV_DEFAULT_YIELD_FILENAME (STDAIR_SAMPLE_DIR
00058                                                    "/yieldstore01.csv");
00059
00063 const std::string K_AIRINV_DEFAULT_SEGMENT_DATE_KEY ("SV,5,2010-03-11,KBP,JFK")
      ;
00064
00068 const stdair::ClassCode_T K_AIRINV_DEFAULT_CLASS_CODE ("Y");
00069
00073 const stdair::PartySize_T K_AIRINV_DEFAULT_PARTY_SIZE (2);
00074
00079 const bool K_AIRINV_DEFAULT_BUILT_IN_INPUT = false;
00080
00085 const bool K_AIRINV_DEFAULT_FOR_SCHEDULE = false;
00086
00090 const int K_AIRINV_EARLY_RETURN_STATUS = 99;
00091
00092 // ///////// Parsing of Options & Configuration /////////
00093 // A helper function to simplify the main part.
00094 template<class T> std::ostream& operator<< (std::ostream& os,
00095                                             const std::vector<T>& v) {
00096   std::copy (v.begin(), v.end(), std::ostream_iterator<T> (std::cout, " "));
00097   return os;
00098 }
00099
00103 int readConfiguration (int argc, char* argv[],
00104                        bool& ioIsBuiltin, bool& ioIsForSchedule,
00105                        stdair::Filename_T& ioInventoryFilename,
00106                        stdair::Filename_T& ioScheduleInputFilename,
00107                        stdair::Filename_T& ioODInputFilename,
00108                        stdair::Filename_T& ioFRAT5Filename,
00109                        stdair::Filename_T& ioFFDisutilityFilename,
00110                        stdair::Filename_T& ioYieldInputFilename,
00111                        std::string& ioSegmentDateKey,
00112                        stdair::ClassCode_T& ioClassCode,
00113                        stdair::PartySize_T& ioPartySize,
00114                        std::string& ioLogFilename) {
00115   // Default for the built-in input
00116   ioIsBuiltin = K_AIRINV_DEFAULT_BUILT_IN_INPUT;
00117
00118   // Default for the inventory or schedule option
00119   ioIsForSchedule = K_AIRINV_DEFAULT_FOR_SCHEDULE;
00120
00121   // Declare a group of options that will be allowed only on command line
00122   boost::program_options::options_description generic ("Generic options");
00123   generic.add_options()
00124     ("prefix", "print installation prefix")
00125     ("version,v", "print version string")
```

```
00126     ("help,h", "produce help message");
00127
00128   // Declare a group of options that will be allowed both on command
00129   // line and in config file
00130
00131   boost::program_options::options_description config ("Configuration");
00132   config.add_options()
00133     ("builtin,b",
00134      "The sample BOM tree can be either built-in or parsed from an input file.
       That latter must then be given with the -i/--inventory or -s/--schedule option")
00135     ("for_schedule,f",
00136      "The BOM tree should be built from a schedule file (instead of from an
       inventory dump)")
00137     ("inventory,i",
00138      boost::program_options::value< std::string >(&ioInventoryFilename)->
      default_value(K_AIRINV_DEFAULT_INVENTORY_FILENAME),
00139      "(CSV) input file for the inventory")
00140     ("schedule,s",
00141      boost::program_options::value< std::string >(&ioScheduleInputFilename)->
      default_value(K_AIRINV_DEFAULT_SCHEDULE_FILENAME),
00142      "(CSV) input file for the schedule")
00143     ("ond,o",
00144      boost::program_options::value< std::string >(&ioODInputFilename)->
      default_value(K_AIRINV_DEFAULT_OND_FILENAME),
00145      "(CSV) input file for the O&D")
00146     ("frat5,F",
00147      boost::program_options::value< std::string >(&ioFRAT5Filename)->
      default_value(K_AIRINV_DEFAULT_FRAT5_INPUT_FILENAME),
00148      "(CSV) input file for the FRAT5 Curve")
00149     ("ff_disutility,D",
00150      boost::program_options::value< std::string >(&ioFFDisutilityFilename)->
      default_value(K_AIRINV_DEFAULT_FF_DISUTILITY_INPUT_FILENAME),
00151      "(CSV) input file for the FF disutility Curve")
00152     ("yield,y",
00153      boost::program_options::value< std::string >(&ioYieldInputFilename)->
      default_value(K_AIRINV_DEFAULT_YIELD_FILENAME),
00154      "(CSV) input file for the yield")
00155     ("segment_date_key,k",
00156      boost::program_options::value< std::string >(&ioSegmentDateKey)->
      default_value(K_AIRINV_DEFAULT_SEGMENT_DATE_KEY),
00157      "Segment-date key")
00158     ("class_code,c",
00159      boost::program_options::value< stdair::ClassCode_T >(&ioClassCode)->
      default_value(K_AIRINV_DEFAULT_CLASS_CODE),
00160      "Class code")
00161     ("party_size,p",
00162      boost::program_options::value< stdair::PartySize_T >(&ioPartySize)->
      default_value(K_AIRINV_DEFAULT_PARTY_SIZE),
00163      "Party size")
00164     ("log,l",
00165      boost::program_options::value< std::string >(&ioLogFilename)->
      default_value(K_AIRINV_DEFAULT_LOG_FILENAME),
00166      "Filename for the logs")
00167     ;
00168
00169   // Hidden options, will be allowed both on command line and
00170   // in config file, but will not be shown to the user.
00171   boost::program_options::options_description hidden ("Hidden options");
00172   hidden.add_options()
00173     ("copyright",
00174      boost::program_options::value< std::vector<std::string> >(),
00175      "Show the copyright (license)");
00176
00177   boost::program_options::options_description cmdline_options;
00178   cmdline_options.add(generic).add(config).add(hidden);
00179
00180   boost::program_options::options_description config_file_options;
00181   config_file_options.add(config).add(hidden);
00182   boost::program_options::options_description visible ("Allowed options");
00183   visible.add(generic).add(config);
00184
00185   boost::program_options::positional_options_description p;
00186   p.add ("copyright", -1);
00187
00188   boost::program_options::variables_map vm;
00189   boost::program_options::
00190     store (boost::program_options::command_line_parser (argc, argv).
00191            options (cmdline_options).positional(p).run(), vm);
00192
00193   std::ifstream ifs ("airinv.cfg");
00194   boost::program_options::store (parse_config_file (ifs, config_file_options),
00195                                  vm);
00196   boost::program_options::notify (vm);
00197
00198   if (vm.count ("help")) {
00199     std::cout << visible << std::endl;
00200     return K_AIRINV_EARLY_RETURN_STATUS;
```

```
00201   }
00202
00203   if (vm.count ("version")) {
00204     std::cout << PACKAGE_NAME << ", version " << PACKAGE_VERSION
      << std::endl;
00205     return K_AIRINV_EARLY_RETURN_STATUS;
00206   }
00207
00208   if (vm.count ("prefix")) {
00209     std::cout << "Installation prefix: " << PREFIXDIR << std::endl;
00210     return K_AIRINV_EARLY_RETURN_STATUS;
00211   }
00212
00213   if (vm.count ("builtin")) {
00214     ioIsBuiltin = true;
00215   }
00216   const std::string isBuiltinStr = (ioIsBuiltin == true)?"yes":"no";
00217   std::cout << "The BOM should be built-in? " << isBuiltinStr << std::endl;
00218
00219   if (vm.count ("for_schedule")) {
00220     ioIsForSchedule = true;
00221   }
00222   const std::string isForScheduleStr = (ioIsForSchedule == true)?"yes":"no";
00223   std::cout << "The BOM should be built from schedule? " << isForScheduleStr
00224             << std::endl;
00225
00226   if (ioIsBuiltin == false) {
00227
00228     if (ioIsForSchedule == false) {
00229       // The BOM tree should be built from parsing an inventory dump
00230       if (vm.count ("inventory")) {
00231         ioInventoryFilename = vm["inventory"].as< std::string >();
00232         std::cout << "Input inventory filename is: " << ioInventoryFilename
00233                   << std::endl;
00234
00235       } else {
00236         // The built-in option is not selected. However, no inventory dump
00237         // file is specified
00238         std::cerr << "Either one among the -b/--builtin, -i/--inventory or "
00239                   << " -f/--for_schedule and -s/--schedule options "
00240                   << "must be specified" << std::endl;
00241       }
00242
00243     } else {
00244       // The BOM tree should be built from parsing a schedule (and O&D) file
00245       if (vm.count ("schedule")) {
00246         ioScheduleInputFilename = vm["schedule"].as< std::string >();
00247         std::cout << "Input schedule filename is: " << ioScheduleInputFilename
00248                   << std::endl;
00249
00250       } else {
00251         // The built-in option is not selected. However, no schedule file
00252         // is specified
00253         std::cerr << "Either one among the -b/--builtin, -i/--inventory or "
00254                   << " -f/--for_schedule and -s/--schedule options "
00255                   << "must be specified" << std::endl;
00256       }
00257
00258       if (vm.count ("ond")) {
00259         ioODInputFilename = vm["ond"].as< std::string >();
00260         std::cout << "Input O&D filename is: " << ioODInputFilename <<
      std::endl;
00261       }
00262
00263       if (vm.count ("frat5")) {
00264         ioFRAT5Filename = vm["frat5"].as< std::string >();
00265         std::cout << "FRAT5 input filename is: " << ioFRAT5Filename <<
      std::endl;
00266
00267       }
00268
00269       if (vm.count ("ff_disutility")) {
00270         ioFFDisutilityFilename = vm["ff_disutility"].as< std::string >();
00271         std::cout << "FF disutility input filename is: "
00272                   << ioFFDisutilityFilename << std::endl;
00273
00274       }
00275
00276       if (vm.count ("yield")) {
00277         ioYieldInputFilename = vm["yield"].as< std::string >();
00278         std::cout << "Input yield filename is: "
00279                   << ioYieldInputFilename << std::endl;
00280       }
00281     }
00282   }
00283
00284   if (vm.count ("log")) {
```

```
00285     ioLogFilename = vm["log"].as< std::string >();
00286     std::cout << "Log filename is: " << ioLogFilename << std::endl;
00287   }
00288
00289   return 0;
00290 }
00291
00292
00293 // ///////// M A I N ////////////
00294 int main (int argc, char* argv[]) {
00295
00296   // State whether the BOM tree should be built-in or parsed from an
00297   // input file
00298   bool isBuiltin;
00299   bool isForSchedule;
00300
00301   // Input file names
00302   stdair::Filename_T lInventoryFilename;
00303   stdair::Filename_T lScheduleInputFilename;
00304   stdair::Filename_T lODInputFilename;
00305   stdair::Filename_T lFRAT5InputFilename;
00306   stdair::Filename_T lFFDisutilityInputFilename;
00307   stdair::Filename_T lYieldInputFilename;
00308
00309   // Parameters for the sale
00310   std::string lSegmentDateKey;
00311   stdair::ClassCode_T lClassCode;
00312   stdair::PartySize_T lPartySize;
00313
00314   // Output log File
00315   stdair::Filename_T lLogFilename;
00316
00317   // Call the command-line option parser
00318   const int lOptionParserStatus =
00319     readConfiguration (argc, argv, isBuiltin, isForSchedule, lInventoryFilename
    ,
00320                        lScheduleInputFilename, lODInputFilename,
00321                        lFRAT5InputFilename, lFFDisutilityInputFilename,
00322                        lYieldInputFilename, lSegmentDateKey, lClassCode,
00323                        lPartySize, lLogFilename);
00324
00325   if (lOptionParserStatus == K_AIRINV_EARLY_RETURN_STATUS) {
00326     return 0;
00327   }
00328
00329   // Set the log parameters
00330   std::ofstream logOutputFile;
00331   // Open and clean the log outputfile
00332   logOutputFile.open (lLogFilename.c_str());
00333   logOutputFile.clear();
00334
00335   // Initialise the inventory service
00336   const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
00337   AIRINV::AIRINV_Master_Service airinvService (
    lLogParams);
00338
00339   // DEBUG
00340   STDAIR_LOG_DEBUG ("Welcome to AirInv");
00341
00342   // Check wether or not a (CSV) input file should be read
00343   if (isBuiltin == true) {
00344
00345     // Build the sample BOM tree for RMOL
00346     airinvService.buildSampleBom();
00347
00348     // Define a specific segment-date key for the sample BOM tree
00349     lSegmentDateKey = "BA,9,2011-06-10,LHR,SYD";
00350
00351   } else {
00352     if (isForSchedule == true) {
00353       // Build the BOM tree from parsing a schedule file (and O&D list)
00354       stdair::ScheduleFilePath lScheduleFilePath (lScheduleInputFilename);
00355       stdair::ODFilePath lODFilePath (lODInputFilename);
00356       stdair::FRAT5FilePath lFRAT5FilePath (lFRAT5InputFilename);
00357       stdair::FFDisutilityFilePath lFFDisutilityFilePath (
    lFFDisutilityInputFilename);
00358       AIRRAC::YieldFilePath lYieldFilePath (lYieldInputFilename);
00359       airinvService.parseAndLoad (lScheduleFilePath, lODFilePath,
00360                                   lFRAT5FilePath, lFFDisutilityFilePath,
00361                                   lYieldFilePath);
00362
00363       if (lSegmentDateKey == K_AIRINV_DEFAULT_SEGMENT_DATE_KEY) {
00364         // Define a specific segment-date key for the schedule-based inventory
00365         lSegmentDateKey = "SQ,11,2010-01-15,SIN,BKK";
00366       }
00367
00368     } else {
```

```
00369       // Build the BOM tree from parsing an inventory dump file
00370       AIRINV::InventoryFilePath lInventoryFilePath (
    lInventoryFilename);
00371       airinvService.parseAndLoad (lInventoryFilePath);
00372     }
00373   }
00374
00375   // Make a booking
00376   const bool isSellSuccessful =
00377     airinvService.sell (lSegmentDateKey, lClassCode, lPartySize);
00378
00379   // DEBUG
00380   STDAIR_LOG_DEBUG ("Sale ('" << lSegmentDateKey << "', " << lClassCode << ": "
00381                     << lPartySize << ") successful? " << isSellSuccessful);
00382
00383   // DEBUG: Display the whole BOM tree
00384   const std::string& lCSVDump = airinvService.csvDisplay();
00385   STDAIR_LOG_DEBUG (lCSVDump);
00386
00387   // Close the Log outputFile
00388   logOutputFile.close();
00389
00390   /*
00391     Note: as that program is not intended to be run on a server in
00392     production, it is better not to catch the exceptions. When it
00393     happens (that an exception is thrownded), that way we get the
00394     call stack.
00395   */
00396
00397   return 0;
00398 }
```

23.29 airinv/batches/parseInventory.cpp File Reference

23.30 parseInventory.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/program_options.hpp>
00013 #include <boost/tokenizer.hpp>
00014 // StdAir
00015 #include <stdair/basic/BasLogParams.hpp>
00016 #include <stdair/basic/BasDBParams.hpp>
00017 #include <stdair/service/Logger.hpp>
00018 // AirInv
00019 #include <airinv/AIRINV_Master_Service.hpp>
00020 #include <airinv/config/airinv-paths.hpp>
00021
00022 // //////// Constants //////
00026 const std::string K_AIRINV_DEFAULT_LOG_FILENAME ("parseInventory.log");
00027
00031 const std::string K_AIRINV_DEFAULT_INVENTORY_FILENAME (STDAIR_SAMPLE_DIR
00032                                                         "/invdump01.csv");
00036 const std::string K_AIRINV_DEFAULT_SCHEDULE_FILENAME (STDAIR_SAMPLE_DIR
00037                                                       "/schedule01.csv");
00041 const std::string K_AIRINV_DEFAULT_OND_FILENAME (STDAIR_SAMPLE_DIR
00042                                                  "/ond01.csv");
00043
00047 const std::string K_AIRINV_DEFAULT_YIELD_FILENAME (STDAIR_SAMPLE_DIR
00048                                                    "/yieldstore01.csv");
00049
00053 const std::string K_AIRINV_DEFAULT_SEGMENT_DATE_KEY ("SV,5,2010-03-11,KBP,JFK")
    ;
00054
00058 const stdair::ClassCode_T K_AIRINV_DEFAULT_CLASS_CODE ("Y");
00059
00063 const stdair::PartySize_T K_AIRINV_DEFAULT_PARTY_SIZE (2);
00064
00069 const bool K_AIRINV_DEFAULT_BUILT_IN_INPUT = false;
00070
00075 const bool K_AIRINV_DEFAULT_FOR_SCHEDULE = false;
00076
00080 const int K_AIRINV_EARLY_RETURN_STATUS = 99;
00081
00082 // ///////// Parsing of Options & Configuration /////////
00083 // A helper function to simplify the main part.
00084 template<class T> std::ostream& operator<< (std::ostream& os,
```

```
00085                                           const std::vector<T>& v) {
00086   std::copy (v.begin(), v.end(), std::ostream_iterator<T> (std::cout, " "));
00087   return os;
00088 }
00089
00093 int readConfiguration (int argc, char* argv[],
00094                        bool& ioIsBuiltin, bool& ioIsForSchedule,
00095                        stdair::Filename_T& ioInventoryFilename,
00096                        stdair::Filename_T& ioScheduleInputFilename,
00097                        stdair::Filename_T& ioODInputFilename,
00098                        stdair::Filename_T& ioYieldInputFilename,
00099                        std::string& ioSegmentDateKey,
00100                        stdair::ClassCode_T& ioClassCode,
00101                        stdair::PartySize_T& ioPartySize,
00102                        std::string& ioLogFilename) {
00103   // Default for the built-in input
00104   ioIsBuiltin = K_AIRINV_DEFAULT_BUILT_IN_INPUT;
00105
00106   // Default for the inventory or schedule option
00107   ioIsForSchedule = K_AIRINV_DEFAULT_FOR_SCHEDULE;
00108
00109   // Declare a group of options that will be allowed only on command line
00110   boost::program_options::options_description generic ("Generic options");
00111   generic.add_options()
00112     ("prefix", "print installation prefix")
00113     ("version,v", "print version string")
00114     ("help,h", "produce help message");
00115
00116   // Declare a group of options that will be allowed both on command
00117   // line and in config file
00118
00119   boost::program_options::options_description config ("Configuration");
00120   config.add_options()
00121     ("builtin,b",
00122      "The sample BOM tree can be either built-in or parsed from an input file.
      That latter must then be given with the -i/--inventory or -s/--schedule option")
00123     ("for_schedule,f",
00124      "The BOM tree should be built from a schedule file (instead of from an
      inventory dump)")
00125     ("inventory,i",
00126      boost::program_options::value< std::string >(&ioInventoryFilename)->
     default_value(K_AIRINV_DEFAULT_INVENTORY_FILENAME),
00127      "(CSV) input file for the inventory")
00128     ("schedule,s",
00129      boost::program_options::value< std::string >(&ioScheduleInputFilename)->
     default_value(K_AIRINV_DEFAULT_SCHEDULE_FILENAME),
00130      "(CSV) input file for the schedule")
00131     ("ond,o",
00132      boost::program_options::value< std::string >(&ioODInputFilename)->
     default_value(K_AIRINV_DEFAULT_OND_FILENAME),
00133      "(CSV) input file for the O&D")
00134     ("yield,y",
00135      boost::program_options::value< std::string >(&ioYieldInputFilename)->
     default_value(K_AIRINV_DEFAULT_YIELD_FILENAME),
00136      "(CSV) input file for the yield")
00137     ("segment_date_key,k",
00138      boost::program_options::value< std::string >(&ioSegmentDateKey)->
     default_value(K_AIRINV_DEFAULT_SEGMENT_DATE_KEY),
00139      "Segment-date key")
00140     ("class_code,c",
00141      boost::program_options::value< stdair::ClassCode_T >(&ioClassCode)->
     default_value(K_AIRINV_DEFAULT_CLASS_CODE),
00142      "Class code")
00143     ("party_size,p",
00144      boost::program_options::value< stdair::PartySize_T >(&ioPartySize)->
     default_value(K_AIRINV_DEFAULT_PARTY_SIZE),
00145      "Party size")
00146     ("log,l",
00147      boost::program_options::value< std::string >(&ioLogFilename)->
     default_value(K_AIRINV_DEFAULT_LOG_FILENAME),
00148      "Filename for the logs")
00149     ;
00150
00151   // Hidden options, will be allowed both on command line and
00152   // in config file, but will not be shown to the user.
00153   boost::program_options::options_description hidden ("Hidden options");
00154   hidden.add_options()
00155     ("copyright",
00156      boost::program_options::value< std::vector<std::string> >(),
00157      "Show the copyright (license)");
00158
00159   boost::program_options::options_description cmdline_options;
00160   cmdline_options.add(generic).add(config).add(hidden);
00161
00162   boost::program_options::options_description config_file_options;
00163   config_file_options.add(config).add(hidden);
00164   boost::program_options::options_description visible ("Allowed options");
```

```
00165   visible.add(generic).add(config);
00166
00167   boost::program_options::positional_options_description p;
00168   p.add ("copyright", -1);
00169
00170   boost::program_options::variables_map vm;
00171   boost::program_options::
00172     store (boost::program_options::command_line_parser (argc, argv).
00173            options (cmdline_options).positional(p).run(), vm);
00174
00175   std::ifstream ifs ("airinv.cfg");
00176   boost::program_options::store (parse_config_file (ifs, config_file_options),
00177                                  vm);
00178   boost::program_options::notify (vm);
00179
00180   if (vm.count ("help")) {
00181     std::cout << visible << std::endl;
00182     return K_AIRINV_EARLY_RETURN_STATUS;
00183   }
00184
00185   if (vm.count ("version")) {
00186     std::cout << PACKAGE_NAME << ", version " << PACKAGE_VERSION
       << std::endl;
00187     return K_AIRINV_EARLY_RETURN_STATUS;
00188   }
00189
00190   if (vm.count ("prefix")) {
00191     std::cout << "Installation prefix: " << PREFIXDIR << std::endl;
00192     return K_AIRINV_EARLY_RETURN_STATUS;
00193   }
00194
00195   if (vm.count ("builtin")) {
00196     ioIsBuiltin = true;
00197   }
00198   const std::string isBuiltinStr = (ioIsBuiltin == true)?"yes":"no";
00199   std::cout << "The BOM should be built-in? " << isBuiltinStr << std::endl;
00200
00201   if (vm.count ("for_schedule")) {
00202     ioIsForSchedule = true;
00203   }
00204   const std::string isForScheduleStr = (ioIsForSchedule == true)?"yes":"no";
00205   std::cout << "The BOM should be built from schedule? " << isForScheduleStr
00206             << std::endl;
00207
00208   if (ioIsBuiltin == false) {
00209
00210     if (ioIsForSchedule == false) {
00211       // The BOM tree should be built from parsing an inventory dump
00212       if (vm.count ("inventory")) {
00213         ioInventoryFilename = vm["inventory"].as< std::string >();
00214         std::cout << "Input inventory filename is: " << ioInventoryFilename
00215                   << std::endl;
00216
00217       } else {
00218         // The built-in option is not selected. However, no inventory dump
00219         // file is specified
00220         std::cerr << "Either one among the -b/--builtin, -i/--inventory or "
00221                   << " -f/--for_schedule and -s/--schedule options "
00222                   << "must be specified" << std::endl;
00223       }
00224
00225     } else {
00226       // The BOM tree should be built from parsing a schedule (and O&D) file
00227       if (vm.count ("schedule")) {
00228         ioScheduleInputFilename = vm["schedule"].as< std::string >();
00229         std::cout << "Input schedule filename is: " << ioScheduleInputFilename
00230                   << std::endl;
00231
00232       } else {
00233         // The built-in option is not selected. However, no schedule file
00234         // is specified
00235         std::cerr << "Either one among the -b/--builtin, -i/--inventory or "
00236                   << " -f/--for_schedule and -s/--schedule options "
00237                   << "must be specified" << std::endl;
00238       }
00239
00240       if (vm.count ("ond")) {
00241         ioODInputFilename = vm["ond"].as< std::string >();
00242         std::cout << "Input O&D filename is: " << ioODInputFilename <<
       std::endl;
00243       }
00244
00245       if (vm.count ("yield")) {
00246         ioYieldInputFilename = vm["yield"].as< std::string >();
00247         std::cout << "Input yield filename is: "
00248                   << ioYieldInputFilename << std::endl;
00249       }
```

```
00250     }
00251   }
00252
00253   if (vm.count ("log")) {
00254     ioLogFilename = vm["log"].as< std::string >();
00255     std::cout << "Log filename is: " << ioLogFilename << std::endl;
00256   }
00257
00258   return 0;
00259 }
00260
00261
00262 // ///////// M A I N ////////////
00263 int main (int argc, char* argv[]) {
00264
00265   // State whether the BOM tree should be built-in or parsed from an
00266   // input file
00267   bool isBuiltin;
00268   bool isForSchedule;
00269
00270   // Input file names
00271   stdair::Filename_T lInventoryFilename;
00272   stdair::Filename_T lScheduleInputFilename;
00273   stdair::Filename_T lODInputFilename;
00274   stdair::Filename_T lYieldInputFilename;
00275
00276   // Parameters for the sale
00277   std::string lSegmentDateKey;
00278   stdair::ClassCode_T lClassCode;
00279   stdair::PartySize_T lPartySize;
00280
00281   // Output log File
00282   stdair::Filename_T lLogFilename;
00283
00284   // Call the command-line option parser
00285   const int lOptionParserStatus =
00286     readConfiguration (argc, argv, isBuiltin, isForSchedule, lInventoryFilename
      ,
00287                        lScheduleInputFilename, lODInputFilename,
00288                        lYieldInputFilename, lSegmentDateKey, lClassCode,
00289                        lPartySize, lLogFilename);
00290
00291   if (lOptionParserStatus == K_AIRINV_EARLY_RETURN_STATUS) {
00292     return 0;
00293   }
00294
00295   // Set the log parameters
00296   std::ofstream logOutputFile;
00297   // Open and clean the log outputfile
00298   logOutputFile.open (lLogFilename.c_str());
00299   logOutputFile.clear();
00300
00301   // Initialise the inventory service
00302   const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
00303   AIRINV::AIRINV_Master_Service airinvService (
      lLogParams);
00304
00305   // DEBUG
00306   STDAIR_LOG_DEBUG ("Welcome to AirInv");
00307
00308   // Check wether or not a (CSV) input file should be read
00309   if (isBuiltin == true) {
00310
00311     // Build the sample BOM tree for RMOL
00312     airinvService.buildSampleBom();
00313
00314     // Define a specific segment-date key for the sample BOM tree
00315     //lSegmentDateKey = "BA,9,2011-06-10,LHR,SYD";
00316     lSegmentDateKey = "SQ,11,2010-02-08,SIN,BKK";
00317
00318   } else {
00319     if (isForSchedule == true) {
00320       // Build the BOM tree from parsing a schedule file (and O&D list)
00321       stdair::ScheduleFilePath lScheduleFilePath (lScheduleInputFilename);
00322       stdair::ODFilePath lODFilePath (lODInputFilename);
00323       AIRRAC::YieldFilePath lYieldFilePath (lYieldInputFilename);
00324       airinvService.parseAndLoad (lScheduleFilePath, lODFilePath,
00325                                   lYieldFilePath);
00326
00327       if (lSegmentDateKey == K_AIRINV_DEFAULT_SEGMENT_DATE_KEY) {
00328         // Define a specific segment-date key for the schedule-based inventory
00329         lSegmentDateKey = "SQ,11,2010-01-15,SIN,BKK";
00330       }
00331
00332     } else {
00333       // Build the BOM tree from parsing an inventory dump file
00334       AIRINV::InventoryFilePath lInventoryFilePath (
```

```
    lInventoryFilename);
00335       airinvService.parseAndLoad (lInventoryFilePath);
00336     }
00337   }
00338
00339   // Make a booking
00340   const bool isSellSuccessful =
00341     airinvService.sell (lSegmentDateKey, lClassCode, lPartySize);
00342
00343   // DEBUG
00344   STDAIR_LOG_DEBUG ("Sale ('" << lSegmentDateKey << "', " << lClassCode << ": "
00345                     << lPartySize << ") successful? " << isSellSuccessful);
00346
00347   // DEBUG: Display the whole BOM tree
00348   const std::string& lCSVDump = airinvService.csvDisplay();
00349   STDAIR_LOG_DEBUG (lCSVDump);
00350
00351   // Close the Log outputFile
00352   logOutputFile.close();
00353
00354   /*
00355     Note: as that program is not intended to be run on a server in
00356     production, it is better not to catch the exceptions. When it
00357     happens (that an exception is thrownded), that way we get the
00358     call stack.
00359   */
00360
00361   return 0;
00362 }
```

23.31 airinv/bom/AirportList.hpp File Reference

```
#include <set>
#include <vector>
#include <stdair/stdair_basic_types.hpp>
```

Namespaces

- namespace AIRINV

Typedefs

- typedef std::set
  < stdair::AirportCode_T > AIRINV::AirportList_T
- typedef std::vector
  < stdair::AirportCode_T > AIRINV::AirportOrderedList_T

23.32 AirportList.hpp

```
00001 #ifndef __AIRINV_BOM_AIRPORTLIST_HPP
00002 #define __AIRINV_BOM_AIRPORTLIST_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <set>
00009 #include <vector>
00010 // STDAIR
00011 #include <stdair/stdair_basic_types.hpp>
00012
00013 namespace AIRINV {
00014
00016   typedef std::set<stdair::AirportCode_T> AirportList_T;
00017   typedef std::vector<stdair::AirportCode_T> AirportOrderedList_T
    ;
00018
00019 }
00020 #endif // __AIRINV_BOM_AIRPORTLIST_HPP
```

23.33 airinv/bom/BomAbstract.cpp File Reference

```
#include <airinv/bom/BomAbstract.hpp>
```

Namespaces

- namespace AIRINV

23.34 BomAbstract.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // AIRINV
00005 #include <airinv/bom/BomAbstract.hpp>
00006
00007 namespace AIRINV {
00008
00009 }
```

23.35 airinv/bom/BomAbstract.hpp File Reference

```
#include <iosfwd>
#include <string>
```

Classes

- class AIRINV::BomAbstract

Namespaces

- namespace AIRINV

Functions

- template<class charT , class traits >
  std::basic_ostream< charT,
  traits > & operator<< (std::basic_ostream< charT, traits > &ioOut, const AIRINV::BomAbstract &iBom)
- template<class charT , class traits >
  std::basic_istream< charT,
  traits > & operator>> (std::basic_istream< charT, traits > &ioIn, AIRINV::BomAbstract &ioBom)

23.35.1 Function Documentation

23.35.1.1 template<class charT , class traits > std::basic_ostream<charT, traits>& operator<< (std::basic_ostream< charT, traits > & *ioOut,* const AIRINV::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 56 of file BomAbstract.hpp.

23.35.1.2 template<class charT , class traits > std::basic_istream<charT, traits>& operator>> (std::basic_istream< charT, traits > & *ioIn,* AIRINV::BomAbstract & *ioBom*) [inline]

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (pp655-657) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 84 of file BomAbstract.hpp.

References AIRINV::BomAbstract::fromStream().

23.36 BomAbstract.hpp

```
00001 #ifndef __AIRINV_BOM_BOMABSTRACT_HPP
00002 #define __AIRINV_BOM_BOMABSTRACT_HPP
00003
00004 // //////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010
00011 namespace AIRINV {
00012
00014   class BomAbstract {
00015     friend class FacBomAbstract;
00016   public:
00017     // ////////// Display support methods /////////
00020     virtual void toStream (std::ostream& ioOut) const = 0;
00021
00024     virtual void fromStream (std::istream& ioIn) = 0;
00025
00027     virtual std::string toString() const = 0;
00028
00031     virtual std::string describeKey() const = 0;
00032
00035     virtual std::string describeShortKey() const = 0;
00036
00037
00038   protected:
00040     BomAbstract() {}
00041     BomAbstract(const BomAbstract&) {}
00042
00044     virtual ~BomAbstract() {}
00045   };
00046 }
00047
00053 template <class charT, class traits>
00054 inline
00055 std::basic_ostream<charT, traits>&
00056 operator<< (std::basic_ostream<charT, traits>& ioOut,
00057             const AIRINV::BomAbstract& iBom) {
00063   std::basic_ostringstream<charT,traits> ostr;
00064   ostr.copyfmt (ioOut);
00065   ostr.width (0);
00066
00067   // Fill string stream
00068   iBom.toStream (ostr);
00069
00070   // Print string stream
00071   ioOut << ostr.str();
00072
00073   return ioOut;
00074 }
00075
00081 template <class charT, class traits>
00082 inline
00083 std::basic_istream<charT, traits>&
00084 operator>> (std::basic_istream<charT, traits>& ioIn,
00085             AIRINV::BomAbstract& ioBom) {
00086   // Fill Bom object with input stream
00087   ioBom.fromStream (ioIn);
00088   return ioIn;
00089 }
00090
00091 #endif // __AIRINV_BOM_BOMABSTRACT_HPP
```

23.37 airinv/bom/BomRootHelper.cpp File Reference

```
#include <cassert>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/Inventory.hpp>
#include <airinv/bom/BomRootHelper.hpp>
#include <airinv/bom/InventoryHelper.hpp>
```

Namespaces

- namespace AIRINV

23.38 BomRootHelper.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // STDAIR
00007 #include <stdair/bom/BomManager.hpp>
00008 #include <stdair/bom/BomRoot.hpp>
00009 #include <stdair/bom/Inventory.hpp>
00010 // AIRINV
00011 #include <airinv/bom/BomRootHelper.hpp>
00012 #include <airinv/bom/InventoryHelper.hpp>
00013
00014 namespace AIRINV {
00015   // ////////////////////////////////////////////////////////////////////
00016   void BomRootHelper::fillFromRouting (const
      stdair::BomRoot& iBomRoot) {
00017     const stdair::InventoryList_T& lInventoryList =
00018       stdair::BomManager::getList<stdair::Inventory> (iBomRoot);
00019
00020     // Browse the list of inventories and update each inventory.
00021     for (stdair::InventoryList_T::const_iterator itInventory =
00022            lInventoryList.begin();
00023          itInventory != lInventoryList.end(); ++itInventory) {
00024       const stdair::Inventory* lCurrentInventory_ptr = *itInventory;
00025       assert (lCurrentInventory_ptr != NULL);
00026       InventoryHelper::fillFromRouting (*
      lCurrentInventory_ptr);
00027     }
00028   }
00029
00030 }
```

23.39 airinv/bom/BomRootHelper.hpp File Reference

Classes

- class AIRINV::BomRootHelper

Namespaces

- namespace stdair

  Forward declarations.
- namespace AIRINV

23.40 BomRootHelper.hpp

```
00001 #ifndef __AIRINV_BOM_BOMROOTHELPER_HPP
00002 #define __AIRINV_BOM_BOMROOTHELPER_HPP
```

```
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007
00008 // Forward declarations.
00009 namespace stdair {
00010   class BomRoot;
00011 }
00012
00013 namespace AIRINV {
00016   class BomRootHelper {
00017   public:
00018     // ////////// Business Methods /////////
00021     static void fillFromRouting (const stdair::BomRoot&);
00022   };
00023
00024 }
00025 #endif // __AIRINV_BOM_BOMROOTHELPER_HPP
```

23.41 airinv/bom/BookingClassHelper.cpp File Reference

```
#include <cassert>
#include <stdair/bom/BookingClass.hpp>
#include <airinv/bom/BookingClassHelper.hpp>
```

Namespaces

- namespace AIRINV

23.42 BookingClassHelper.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // STDAIR
00007 #include <stdair/bom/BookingClass.hpp>
00008 // AIRINV
00009 #include <airinv/bom/BookingClassHelper.hpp>
00010
00011 namespace AIRINV {
00012
00013 }
```

23.43 airinv/bom/BookingClassHelper.hpp File Reference

Classes

- class AIRINV::BookingClassHelper

Namespaces

- namespace stdair

  Forward declarations.
- namespace AIRINV

23.44 BookingClassHelper.hpp

```
00001 #ifndef __AIRINV_BOM_BOOKINGCLASSHELPER_HPP
00002 #define __AIRINV_BOM_BOOKINGCLASSHELPER_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
```

```
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // StdAir
00008 //#include <stdair/stdair_basic_types.hpp>
00009
00010 // Forward declarations
00011 namespace stdair {
00012   class BookingClass;
00013 }
00014
00015 namespace AIRINV {
00016
00019   class BookingClassHelper {
00020
00021   };
00022
00023 }
00024 #endif // __AIRINV_BOM_BOOKINGCLASSHELPER_HPP
```

23.45 airinv/bom/BookingClassStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <airinv/bom/BookingClassStruct.hpp>
```

Namespaces

- namespace AIRINV

23.46 BookingClassStruct.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/BookingClass.hpp>
00010 // AirInv
00011 #include <airinv/bom/BookingClassStruct.hpp>
00012
00013 namespace AIRINV {
00014
00015   // ////////////////////////////////////////////////////////////////////
00016   BookingClassStruct::BookingClassStruct
      () {
00017   }
00018
00019   // ////////////////////////////////////////////////////////////////////
00020   stdair::ClassCode_T BookingClassStruct::getFullSubclassCode
      () const {
00021     std::ostringstream oStr;
00022     oStr << _classCode << _subclassCode;
00023     return oStr.str();
00024   }
00025
00026   // ////////////////////////////////////////////////////////////////////
00027   const std::string BookingClassStruct::describe()
       const {
00028     std::ostringstream ostr;
00029     ostr << "            " << _classCode << _subclassCode
00030          << " (" << _parentClassCode << _parentSubclassCode
       << ")"
00031          << ", " << _cumulatedProtection << ":" <<
      _protection
00032          << ", " << _nego
00033          << ", " << _noShowPercentage << ":" <<
      _overbookingPercentage
00034          << ", " << _nbOfBookings << ":" << _nbOfGroupBookings
00035          << ":" << _nbOfPendingGroupBookings << ":" <<
```

```
      _nbOfStaffBookings
00036          << ":" << _nbOfWLBookings << ":" << _etb
00037          << ", " << _netClassAvailability  << ":" <<
      _segmentAvailability
00038          << ":" << _netRevenueAvailability
00039          << std::endl;
00040     return ostr.str();
00041   }
00042
00043   // ////////////////////////////////////////////////////////////////////
00044   void BookingClassStruct::fill (stdair::BookingClass&
      ioBookingClass) const {
00045     // Set the Yield Range Upper Value
00046     // ioBookingClass.setYieldRangeValue (_yieldRangeUpperValue);
00047
00048     // Set the Availability
00049     // ioBookingClass.setAvailability (_availability);
00050
00051     // Set the number of seats
00052     // ioBookingClass.setNbOfSeats (_nbOfSeats);
00053
00054     // Set the Seat Index
00055     // ioBookingClass.setSeatIndex (_seatIndex);
00056   }
00057
00058 }
```

23.47 airinv/bom/BookingClassStruct.hpp File Reference

```
#include <string>
#include <vector>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <airinv/AIRINV_Types.hpp>
```

Classes

- struct AIRINV::BookingClassStruct

Namespaces

- namespace stdair

  Forward declarations.
- namespace AIRINV

Typedefs

- typedef std::vector
  < BookingClassStruct > AIRINV::BookingClassStructList_T

23.48 BookingClassStruct.hpp

```
00001 #ifndef __AIRINV_BOM_BOOKINGCLASSSTRUCT_HPP
00002 #define __AIRINV_BOM_BOOKINGCLASSSTRUCT_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/basic/StructAbstract.hpp>
00013 // AirInv
00014 #include <airinv/AIRINV_Types.hpp>
```

```
00015
00016 // Forward declarations
00017 namespace stdair {
00018   class BookingClass;
00019 }
00020
00021 namespace AIRINV {
00022
00024   struct BookingClassStruct : public stdair::StructAbstract {
00025     // Attributes
00026     stdair::ClassCode_T _classCode;
00027     stdair::SubclassCode_T _subclassCode;
00028     stdair::ClassCode_T _parentClassCode;
00029     stdair::SubclassCode_T _parentSubclassCode;
00030     stdair::AuthorizationLevel_T _cumulatedProtection;
00031     stdair::AuthorizationLevel_T _protection;
00032     stdair::NbOfSeats_T _nego;
00033     stdair::OverbookingRate_T _noShowPercentage;
00034     stdair::OverbookingRate_T _overbookingPercentage;
00035     stdair::NbOfBookings_T _nbOfBookings;
00036     stdair::NbOfBookings_T _nbOfGroupBookings;
00037     stdair::NbOfBookings_T _nbOfPendingGroupBookings;
00038     stdair::NbOfBookings_T _nbOfStaffBookings;
00039     stdair::NbOfBookings_T _nbOfWLBookings;
00040     stdair::NbOfBookings_T _etb;
00041     stdair::Availability_T _netClassAvailability;
00042     stdair::Availability_T _segmentAvailability;
00043     stdair::Availability_T _netRevenueAvailability;
00044
00046     stdair::ClassCode_T getFullSubclassCode() const;
00047
00050     void fill (stdair::BookingClass&) const;
00051
00053     const std::string describe() const;
00054
00056     BookingClassStruct();
00057   };
00058
00060   typedef std::vector<BookingClassStruct> BookingClassStructList_T
      ;
00061
00062 }
00063 #endif // __AIRINV_BOM_BOOKINGCLASSSTRUCT_HPP
```

23.49 airinv/bom/BucketStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/Bucket.hpp>
#include <airinv/bom/BucketStruct.hpp>
```

Namespaces

- namespace AIRINV

23.50 BucketStruct.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/Bucket.hpp>
00010 // AirInv
00011 #include <airinv/bom/BucketStruct.hpp>
00012
00013 namespace AIRINV {
00014
00015   // ////////////////////////////////////////////////////////////////////
00016   BucketStruct::BucketStruct() : _nbOfSeats (0.0) {
```

```
00017   }
00018
00019   // ////////////////////////////////////////////////////////////////////
00020   const std::string BucketStruct::describe() const {
00021     std::ostringstream ostr;
00022     ostr << "            " << _yieldRangeUpperValue << ":"
      << _availability
00023          << ":" << _nbOfSeats  << ":" << _seatIndex
00024          << std::endl;
00025     return ostr.str();
00026   }
00027
00028   // ////////////////////////////////////////////////////////////////////
00029   void BucketStruct::fill (stdair::Bucket& ioBucket) const {
00030     // Set the Yield Range Upper Value
00031     ioBucket.setYieldRangeUpperValue (_yieldRangeUpperValue
      );
00032
00033     // Set the Availability
00034     ioBucket.setAvailability (_availability);
00035
00036     // Set the number of sold seats
00037     ioBucket.setSoldSeats (_nbOfSeats);
00038   }
00039
00040 }
```

23.51 airinv/bom/BucketStruct.hpp File Reference

```
#include <string>
#include <vector>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <airinv/AIRINV_Types.hpp>
```

Classes

- struct AIRINV::BucketStruct

  Utility Structure for the parsing of Bucket structures.

Namespaces

- namespace stdair

  Forward declarations.
- namespace AIRINV

Typedefs

- typedef std::vector< BucketStruct > AIRINV::BucketStructList_T

23.52 BucketStruct.hpp

```
00001 #ifndef __AIRINV_BOM_BUCKETSTRUCT_HPP
00002 #define __AIRINV_BOM_BUCKETSTRUCT_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/basic/StructAbstract.hpp>
00013 // AirInv
00014 #include <airinv/AIRINV_Types.hpp>
```

```
00015
00017 namespace stdair {
00018   class Bucket;
00019 }
00020
00021 namespace AIRINV {
00022
00026   struct BucketStruct : public stdair::StructAbstract {
00027     // Attributes
00028     stdair::Yield_T _yieldRangeUpperValue;
00029     stdair::CabinCapacity_T _availability;
00030     stdair::NbOfSeats_T _nbOfSeats;
00031     stdair::SeatIndex_T _seatIndex;
00032
00034     void fill (stdair::Bucket&) const;
00035
00037     const std::string describe() const;
00038
00040     BucketStruct();
00041   };
00042
00044   typedef std::vector<BucketStruct> BucketStructList_T;
00045
00046 }
00047 #endif // __AIRINV_BOM_BUCKETSTRUCT_HPP
```

23.53 airinv/bom/DCPEventStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <vector>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/service/Logger.hpp>
#include <airinv/AIRINV_Types.hpp>
#include <airinv/bom/DCPEventStruct.hpp>
```

Namespaces

- namespace AIRINV

23.54 DCPEventStruct.cpp

```
00001 // //////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 #include <vector>
00008 // StdAir
00009 #include <stdair/basic/BasConst_General.hpp>
00010 #include <stdair/service/Logger.hpp>
00011 // AirInv
00012 #include <airinv/AIRINV_Types.hpp>
00013 #include <airinv/bom/DCPEventStruct.hpp>
00014
00015 namespace AIRINV {
00016
00017   // ////////////////////////////////////////////////////////////////////
00018   DCPEventStruct::DCPEventStruct ()
00019     : _origin(""),
00020       _destination(""),
00021       _dateRangeStart(stdair::DEFAULT_DATE),
00022       _dateRangeEnd(stdair::DEFAULT_DATE),
00023       _timeRangeStart(stdair::DEFAULT_EPSILON_DURATION),
00024       _timeRangeEnd(stdair::DEFAULT_EPSILON_DURATION),
00025       _cabinCode (""),
00026       _pos (""),
00027       _advancePurchase(0),
00028       _saturdayStay("T"),
00029       _changeFees("T"),
00030       _nonRefundable("T"),
00031       _minimumStay(0),
```

```
00032       _DCP(0),
00033       _airlineCode(""),
00034       _classCode("") {
00035   }
00036
00037   // ////////////////////////////////////////////////////////////////////
00038   stdair::Date_T DCPEventStruct::getDate() const {
00039     _itYear.check(); _itMonth.check(); _itDay.check();
00040     return stdair::Date_T (_itYear._value, _itMonth._value,
      _itDay._value);
00041   }
00042
00043   // ////////////////////////////////////////////////////////////////////
00044   stdair::Duration_T DCPEventStruct::getTime() const {
00045     _itHours.check(); _itMinutes.check(); _itSeconds
      .check();
00046     return boost::posix_time::hours (_itHours._value)
00047       + boost::posix_time::minutes (_itMinutes._value)
00048       + boost::posix_time::seconds (_itSeconds._value);
00049   }
00050
00051
00052   // ////////////////////////////////////////////////////////////////////
00053   const std::string DCPEventStruct::describe () const {
00054     std::ostringstream ostr;
00055     ostr << "DCPEvent: "
00056          << _origin << "-" << _destination
00057          << ", POS(" << _pos << "), ["
00058          << _dateRangeStart << "/" << _dateRangeEnd
       << "] - ["
00059          << boost::posix_time::to_simple_string(_timeRangeStart)
       << "/"
00060          << boost::posix_time::to_simple_string(_timeRangeEnd) <<
      "]\n    "
00061          << "-Cabin code- " << _cabinCode << "\n    "
00062          << "-Channel-    " << _channel << "\n    "
00063          << "-Conditions- " << _saturdayStay << ", " <<
      _changeFees << ", "
00064          << _nonRefundable << ", " << _advancePurchase
       << ", "
00065          << _minimumStay << "\n    "
00066          << "-DCP-       " << _DCP << "\n           ";
00067     assert (_airlineCodeList.size() == _classCodeList
      .size());
00068     stdair::ClassList_StringList_T::const_iterator lItCurrentClassCode =
00069       _classCodeList.begin();
00070     stdair::AirlineCode_T lAirlineCode;
00071     std::string lClassCode;
00072     for (stdair::AirlineCodeList_T::const_iterator lItCurrentAirlineCode =
00073            _airlineCodeList.begin();
00074          lItCurrentAirlineCode != _airlineCodeList.end();
00075          lItCurrentAirlineCode++) {
00076       lAirlineCode = *lItCurrentAirlineCode;
00077       lClassCode = *lItCurrentClassCode;
00078       ostr << lAirlineCode << ", " << lClassCode;
00079       ostr << "        ";
00080       lItCurrentClassCode++;
00081     }
00082     ostr << std::endl;
00083     return ostr.str();
00084   }
00085
00086  // ////////////////////////////////////////////////////////////////////
00087   const stdair::AirlineCode_T& DCPEventStruct::getFirstAirlineCode
       () const {
00088     assert (_airlineCodeList.size() > 0);
00089     stdair::AirlineCodeList_T::const_iterator itFirstAirlineCode =
00090       _airlineCodeList.begin();
00091     return *itFirstAirlineCode;
00092   }
00093
00094   // ////////////////////////////////////////////////////////////////////
00095   void DCPEventStruct::beginAirline () {
00096     _itCurrentAirlineCode = _airlineCodeList
      .begin();
00097   }
00098
00099   // ////////////////////////////////////////////////////////////////////
00100   bool DCPEventStruct::hasNotReachedEndAirline
       () const {
00101     bool result = (_itCurrentAirlineCode !=
      _airlineCodeList.end());
00102     return result;
00103   }
00104
00105   // ////////////////////////////////////////////////////////////////////
00106   stdair::AirlineCode_T DCPEventStruct::getCurrentAirlineCode
```

```
      () const {
00107     assert (_itCurrentAirlineCode != _airlineCodeList
      .end());
00108       return (*_itCurrentAirlineCode);
00109   }
00110
00111   // ////////////////////////////////////////////////////////////////////
00112   void DCPEventStruct::iterateAirline () {
00113     if (_itCurrentAirlineCode != _classCodeList
      .end()) {
00114       _itCurrentAirlineCode++;
00115     }
00116   }
00117
00118   // ////////////////////////////////////////////////////////////////////
00119   const std::string& DCPEventStruct::getFirstClassCode
      () const {
00120     assert (_classCodeList.size() > 0);
00121     stdair::ClassList_StringList_T::const_iterator itFirstClassCode =
00122       _classCodeList.begin();
00123     return *itFirstClassCode;
00124   }
00125
00126   // ////////////////////////////////////////////////////////////////////
00127   void DCPEventStruct::beginClassCode () {
00128     _itCurrentClassCode = _classCodeList.begin
      ();
00129   }
00130
00131   // ////////////////////////////////////////////////////////////////////
00132   bool DCPEventStruct::hasNotReachedEndClassCode
      () const {
00133     bool result = (_itCurrentClassCode != _classCodeList
      .end());
00134     return result;
00135   }
00136
00137   // ////////////////////////////////////////////////////////////////////
00138   std::string DCPEventStruct::getCurrentClassCode
      () const {
00139     assert (_itCurrentClassCode != _classCodeList
      .end());
00140     return (*_itCurrentClassCode);
00141   }
00142
00143
00144   // ////////////////////////////////////////////////////////////////////
00145   void DCPEventStruct::iterateClassCode () {
00146     if (_itCurrentClassCode != _classCodeList.
      end()) {
00147       _itCurrentClassCode++;
00148     }
00149   }
00150
00151 }
00152
```

23.55 airinv/bom/DCPEventStruct.hpp File Reference

```
#include <string>
#include <vector>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/basic/BasParserTypes.hpp>
#include <airinv/AIRINV_Types.hpp>
```

Classes

- struct AIRINV::DCPEventStruct

Namespaces

- namespace AIRINV

23.56 DCPEventStruct.hpp

```
00001 #ifndef __AIRINV_BOM_DCPEVENTSTRUCT_HPP
00002 #define __AIRINV_BOM_DCPEVENTSTRUCT_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010 // StdAir
00011 #include <stdair/stdair_demand_types.hpp>
00012 #include <stdair/stdair_inventory_types.hpp>
00013 #include <stdair/basic/StructAbstract.hpp>
00014 #include <stdair/basic/BasParserTypes.hpp>
00015 // AirInv
00016 #include <airinv/AIRINV_Types.hpp>
00017
00018 namespace AIRINV {
00019
00021   struct DCPEventStruct : public stdair::StructAbstract {
00022   public:
00023
00025     DCPEventStruct ();
00026
00028     stdair::Date_T getDate() const;
00029
00031     stdair::Duration_T getTime() const;
00032
00034     const std::string describe() const;
00035
00037     const unsigned int getAirlineListSize () const {
00038       return _airlineCodeList.size();
00039     }
00040
00042     const unsigned int getClassCodeListSize () const {
00043       return _classCodeList.size();
00044     }
00045
00047     const stdair::AirlineCode_T& getFirstAirlineCode ()
      const;
00048
00052     void beginAirline ();
00053
00056     bool hasNotReachedEndAirline () const;
00057
00059     stdair::AirlineCode_T getCurrentAirlineCode () const;
00060
00063     void iterateAirline ();
00064
00066     const std::string& getFirstClassCode () const;
00067
00071     void beginClassCode ();
00072
00075     bool hasNotReachedEndClassCode () const;
00076
00078     std::string getCurrentClassCode () const;
00079
00082     void iterateClassCode ();
00083
00084   public:
00085     // ////////////////// Attributes /////////////////
00087     stdair::year_t _itYear;
00088     stdair::month_t _itMonth;
00089     stdair::day_t _itDay;
00090
00092     //long _itHours;
00093     stdair::hour_t _itHours;
00094     stdair::minute_t _itMinutes;
00095     stdair::second_t _itSeconds;
00096
00098     stdair::AirlineCodeList_T::iterator _itCurrentAirlineCode
      ;
00099
00101     stdair::ClassList_StringList_T::iterator _itCurrentClassCode
      ;
00102
00104     stdair::AirportCode_T _origin;
00105
00107     stdair::AirportCode_T _destination;
00108
00110     stdair::Date_T _dateRangeStart;
00111
00113     stdair::Date_T _dateRangeEnd;
00114
00116     stdair::Duration_T _timeRangeStart;
```

```
00117
00119     stdair::Duration_T _timeRangeEnd;
00120
00122     stdair::CabinCode_T _cabinCode;
00123
00125     stdair::CityCode_T _pos;
00126
00128     stdair::ChannelLabel_T _channel;
00129
00131     stdair::DayDuration_T _advancePurchase;
00132
00134     stdair::SaturdayStay_T _saturdayStay;
00135
00137     stdair::ChangeFees_T _changeFees;
00138
00140     stdair::NonRefundable_T _nonRefundable;
00141
00143     stdair::DayDuration_T _minimumStay;
00144
00146     stdair::PriceValue_T _DCP;
00147
00149     stdair::AirlineCode_T _airlineCode;
00150
00152     stdair::ClassCode_T _classCode;
00153
00155     stdair::AirlineCodeList_T _airlineCodeList;
00156
00158     //unsigned long int _nbOfAirlines;
00159
00161     stdair::ClassList_StringList_T _classCodeList;
00162
00163   };
00164
00165 }
00166 #endif // __AIRINV_BOM_DCPEVENTSTRUCT_HPP
```

23.57 airinv/bom/FareFamilyStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <airinv/bom/FareFamilyStruct.hpp>
```

Namespaces

- namespace AIRINV

23.58 FareFamilyStruct.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/FareFamily.hpp>
00010 // AirInv
00011 #include <airinv/bom/FareFamilyStruct.hpp>
00012
00013 namespace AIRINV {
00014
00015   // ////////////////////////////////////////////////////////////////////
00016   FareFamilyStruct::FareFamilyStruct()
00017     : _familyCode (stdair::DEFAULT_NULL_FARE_FAMILY_CODE),
00018       _classes (stdair::DEFAULT_NULL_CLASS_CODE) {
00019   }
00020
00021   // ////////////////////////////////////////////////////////////////////
00022   FareFamilyStruct::
00023   FareFamilyStruct (const stdair::FamilyCode_T& iFamilyCode,
00024                     const stdair::CurveKey_T& iFRAT5Key,
```

```
00025                     const stdair::CurveKey_T& iFFDisutilityKey,
00026                     const stdair::ClassList_String_T& iClasses)
00027     : _familyCode (iFamilyCode), _frat5CurveKey (iFRAT5Key),
00028       _ffDisutilityCurveKey (iFFDisutilityKey), _classes (iClasses) {
00029   }
00030
00031   // ////////////////////////////////////////////////////////////////////
00032   const std::string FareFamilyStruct::describe()
      const {
00033     std::ostringstream ostr;
00034
00035     ostr << "        " << _familyCode << " "
00036          << _frat5CurveKey << " " << _ffDisutilityCurveKey
00037          << " " << _classes << ", ";
00038
00039     for (BookingClassStructList_T::const_iterator itBkgClass= _classList
      .begin();
00040          itBkgClass != _classList.end(); ++itBkgClass) {
00041       const BookingClassStruct& lBkgClass = *itBkgClass;
00042       ostr << lBkgClass.describe();
00043     }
00044     if (_classList.empty() == false) {
00045       ostr << std::endl;
00046     }
00047
00048     return ostr.str();
00049   }
00050
00051   // ////////////////////////////////////////////////////////////////////
00052   void FareFamilyStruct::fill (stdair::FareFamily&
      ioFareFamily) const {
00053     // Set attributes
00054     // ioFareFamily.setSomeAttribute (_someAttribute);
00055   }
00056
00057 }
```

23.59 airinv/bom/FareFamilyStruct.hpp File Reference

```
#include <string>
#include <vector>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <airinv/bom/BookingClassStruct.hpp>
```

Classes

- struct AIRINV::FareFamilyStruct

  Utility Structure for the parsing of fare family details.

Namespaces

- namespace stdair

  Forward declarations.
- namespace AIRINV

Typedefs

- typedef std::vector
  < FareFamilyStruct > AIRINV::FareFamilyStructList_T

23.60 FareFamilyStruct.hpp

```
00001 #ifndef __AIRINV_BOM_FAREFAMILYSTRUCT_HPP
00002 #define __AIRINV_BOM_FAREFAMILYSTRUCT_HPP
```

```
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/basic/StructAbstract.hpp>
00013 // AirInv
00014 #include <airinv/bom/BookingClassStruct.hpp>
00015
00017 namespace stdair {
00018   class FareFamily;
00019 }
00020
00021 namespace AIRINV {
00022
00026   struct FareFamilyStruct : public stdair::StructAbstract {
00027     // Attributes
00028     stdair::FamilyCode_T _familyCode;
00029     stdair::CurveKey_T _frat5CurveKey;
00030     stdair::CurveKey_T _ffDisutilityCurveKey;
00031     stdair::ClassList_String_T _classes;
00032     BookingClassStructList_T _classList;
00033
00037     FareFamilyStruct();
00041     FareFamilyStruct (const stdair::FamilyCode_T&,
00042                       const stdair::CurveKey_T&, const stdair::CurveKey_T&,
00043                       const stdair::ClassList_String_T&);
00044
00048     void fill (stdair::FareFamily&) const;
00049
00053     const std::string describe() const;
00054   };
00055
00059   typedef std::vector<FareFamilyStruct> FareFamilyStructList_T
      ;
00060
00061 }
00062 #endif // __AIRINV_BOM_FAREFAMILYSTRUCT_HPP
```

23.61 airinv/bom/FFDisutilityStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/service/Logger.hpp>
#include <airinv/bom/FFDisutilityStruct.hpp>
```

Namespaces

- namespace AIRINV

23.62 FFDisutilityStruct.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/service/Logger.hpp>
00009 // AIRINV
00010 #include <airinv/bom/FFDisutilityStruct.hpp>
00011
00012 namespace AIRINV {
00013
00014   // ////////////////////////////////////////////////////////////////////
00015   FFDisutilityStruct::FFDisutilityStruct(
      ) {
00016   }
00017
00018   // ////////////////////////////////////////////////////////////////////
```

```
00019   FFDisutilityStruct::~FFDisutilityStruct
      () {
00020   }
00021
00022   // ////////////////////////////////////////////////////////////////////
00023   const std::string FFDisutilityStruct::describe()
       const {
00024     std::ostringstream oStr;
00025     oStr << _key << "; ";
00026     for (stdair::FFDisutilityCurve_T::const_reverse_iterator itFFDisutility =
00027            _curve.rbegin(); itFFDisutility != _curve.rend(); ++
      itFFDisutility) {
00028       const stdair::DTD_T& lDTD = itFFDisutility->first;
00029       const double& lFFDisutility = itFFDisutility->second;
00030       oStr << lDTD << ":" << lFFDisutility << ";";
00031     }
00032
00033     return oStr.str();
00034   }
00035
00036 }
```

23.63 airinv/bom/FFDisutilityStruct.hpp File Reference

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

Classes

- struct AIRINV::FFDisutilityStruct

Namespaces

- namespace AIRINV

23.64 FFDisutilityStruct.hpp

```
00001 #ifndef __AIRINV_BOM_FFDISUTILITYSTRUCT_HPP
00002 #define __AIRINV_BOM_FFDISUTILITYSTRUCT_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/stdair_rm_types.hpp>
00011
00012 namespace AIRINV {
00013
00015   struct FFDisutilityStruct : public stdair::StructAbstract {
00016
00017   public:
00018     // ////////////////// Display Support Methods /////////////////
00020     const std::string describe() const;
00021
00022
00023   public:
00024     // ///////////////// Constructors and destructors ///////////////
00026     FFDisutilityStruct();
00028     ~FFDisutilityStruct();
00029   private:
00031     FFDisutilityStruct (const FFDisutilityStruct
      &);
00032
00033
00034   public:
00035     // ////////////// Attributes ////////////////////
00037     std::string _key;
00038
00040     stdair::FFDisutilityCurve_T _curve;
00041
00042   public:
```

```
00043     // ////////////////// Staging ////////////////////
00045     stdair::DTD_T _dtd;
00046   };
00047
00048 }
00049 #endif // __AIRINV_BOM_FFDISUTILITYSTRUCT_HPP
```

23.65 airinv/bom/FlightDateHelper.cpp File Reference

```
#include <cassert>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <airinv/bom/FlightDateHelper.hpp>
#include <airinv/bom/SegmentDateHelper.hpp>
#include <airinv/bom/SegmentCabinHelper.hpp>
```

Namespaces

- namespace AIRINV

23.66 FlightDateHelper.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // STDAIR
00007 #include <stdair/basic/BasConst_Inventory.hpp>
00008 #include <stdair/bom/BomManager.hpp>
00009 #include <stdair/bom/FlightDate.hpp>
00010 #include <stdair/bom/SegmentDate.hpp>
00011 #include <stdair/bom/SegmentCabin.hpp>
00012 #include <stdair/bom/LegCabin.hpp>
00013 // AIRINV
00014 #include <airinv/bom/FlightDateHelper.hpp>
00015 #include <airinv/bom/SegmentDateHelper.hpp>
00016 #include <airinv/bom/SegmentCabinHelper.hpp>
00017
00018 namespace AIRINV {
00019
00020   // ////////////////////////////////////////////////////////////////////
00021   void FlightDateHelper::
00022   updateBookingControls (stdair::FlightDate&
      ioFlightDate) {
00023
00024     // Parse the segment-cabin list and build the pseudo bid price vector.
00025     const stdair::SegmentDateList_T& lSDList =
00026       stdair::BomManager::getList<stdair::SegmentDate> (ioFlightDate);
00027     for (stdair::SegmentDateList_T::const_iterator itSD = lSDList.begin();
00028          itSD != lSDList.end(); ++itSD) {
00029       const stdair::SegmentDate* lSD_ptr = *itSD;
00030       assert (lSD_ptr != NULL);
00031
00032       //
00033       const stdair::SegmentCabinList_T& lSCList =
00034         stdair::BomManager::getList<stdair::SegmentCabin> (*lSD_ptr);
00035       for (stdair::SegmentCabinList_T::const_iterator itSC = lSCList.begin();
00036            itSC != lSCList.end(); ++itSC) {
00037         stdair::SegmentCabin* lSC_ptr = *itSC;
00038         assert (lSC_ptr != NULL);
00039
00040         // Build the pseudo bid price vector for the segment-cabin.
00041         SegmentCabinHelper::buildPseudoBidPriceVector
       (*lSC_ptr);
00042
00043         // Update the booking controls using the pseudo bid price vector.
00044         SegmentCabinHelper::
```

```
00045           updateBookingControlsUsingPseudoBidPriceVector
       (*lSC_ptr);
00046       }
00047     }
00048   }
00049
00050   // ////////////////////////////////////////////////////////////////////
00051   void FlightDateHelper::fillFromRouting(const
       stdair::FlightDate& iFlightDate){
00052     const stdair::SegmentDateList_T& lSegmentDateList =
00053       stdair::BomManager::getList<stdair::SegmentDate> (iFlightDate);
00054
00055     // Browse the list of segment-dates and update each segment-date.
00056     for (stdair::SegmentDateList_T::const_iterator itSegmentDate =
00057            lSegmentDateList.begin();
00058          itSegmentDate != lSegmentDateList.end(); ++itSegmentDate) {
00059       stdair::SegmentDate* lCurrentSegmentDate_ptr = *itSegmentDate;
00060       assert (lCurrentSegmentDate_ptr != NULL);
00061       SegmentDateHelper::fillFromRouting (*
      lCurrentSegmentDate_ptr);
00062     }
00063   }
00064
00065   // ////////////////////////////////////////////////////////////////////
00066   void FlightDateHelper::
00067   updateAvailability (const stdair::FlightDate& iFlightDate
      ,
00068                       const stdair::SegmentCabin& iSegmentCabin,
00069                       const stdair::PartySize_T& iNbOfBookings) {
00070     // Update the commited space of the member leg-cabins.
00071     const stdair::LegCabinList_T& lLegCabinList =
00072       stdair::BomManager::getList<stdair::LegCabin> (iSegmentCabin);
00073     for (stdair::LegCabinList_T::const_iterator itLegCabin =
00074            lLegCabinList.begin();
00075          itLegCabin != lLegCabinList.end(); ++itLegCabin) {
00076       stdair::LegCabin* lLegCabin_ptr = *itLegCabin;
00077       assert (lLegCabin_ptr != NULL);
00078       lLegCabin_ptr->updateFromReservation (iNbOfBookings);
00079     }
00080
00081     // Update the availability pool of all the segment-cabin which belong to
       the
00082     // same flight-date.
00083     const stdair::CabinCode_T& lCabinCode = iSegmentCabin.getCabinCode();
00084     FlightDateHelper::updateAvailabilityPool
       (iFlightDate, lCabinCode);
00085
00086     // Recaculate the availability of all classes of the given cabin (code).
00087     FlightDateHelper::recalculateAvailability
       (iFlightDate, lCabinCode);
00088   }
00089
00090   // ////////////////////////////////////////////////////////////////////
00091   void FlightDateHelper::
00092   updateAvailabilityPool (const stdair::FlightDate&
      iFlightDate,
00093                           const stdair::CabinCode_T& iCabinCode){
00094     const stdair::SegmentDateList_T& lSegmentDateList =
00095       stdair::BomManager::getList<stdair::SegmentDate> (iFlightDate);
00096     for (stdair::SegmentDateList_T::const_iterator itSegmentDate =
00097            lSegmentDateList.begin(); itSegmentDate != lSegmentDateList.end();
00098          ++itSegmentDate) {
00099       const stdair::SegmentDate* lSegmentDate_ptr = *itSegmentDate;
00100       assert (lSegmentDate_ptr != NULL);
00101       stdair::SegmentCabin& lSegmentCabin =
00102         stdair::BomManager::getObject<stdair::SegmentCabin> (*lSegmentDate_ptr,
00103                                                              iCabinCode);
00104
00105       // Update the availability pool of the segment-cabin to the minimal
00106       // availability pool of the member leg-cabins.
00107       const stdair::LegCabinList_T& lLegCabinList =
00108         stdair::BomManager::getList<stdair::LegCabin> (lSegmentCabin);
00109       stdair::Availability_T lAvailabilityPool = stdair::MAXIMAL_AVAILABILITY;
00110       for (stdair::LegCabinList_T::const_iterator itLegCabin =
00111              lLegCabinList.begin();
00112            itLegCabin != lLegCabinList.end(); ++itLegCabin) {
00113         const stdair::LegCabin* lLegCabin_ptr = *itLegCabin;
00114         assert (lLegCabin_ptr != NULL);
00115         const stdair::Availability_T& lLegCabinAvailabilityPool =
00116           lLegCabin_ptr->getAvailabilityPool();
00117         if (lAvailabilityPool > lLegCabinAvailabilityPool) {
00118           lAvailabilityPool = lLegCabinAvailabilityPool;
00119         }
00120       }
00121       lSegmentCabin.setAvailabilityPool (lAvailabilityPool);
00122     }
00123   }
```

```
00124
00125   // ////////////////////////////////////////////////////////////////////
00126   void FlightDateHelper::
00127   recalculateAvailability (const stdair::FlightDate&
      iFlightDate,
00128                            const stdair::CabinCode_T& iCabinCode){
00129     const stdair::SegmentDateList_T& lSegmentDateList =
00130       stdair::BomManager::getList<stdair::SegmentDate> (iFlightDate);
00131     for (stdair::SegmentDateList_T::const_iterator itSegmentDate =
00132            lSegmentDateList.begin(); itSegmentDate != lSegmentDateList.end();
00133          ++itSegmentDate) {
00134       const stdair::SegmentDate* lSegmentDate_ptr = *itSegmentDate;
00135       assert (lSegmentDate_ptr != NULL);
00136       stdair::SegmentCabin& lSegmentCabin =
00137         stdair::BomManager::getObject<stdair::SegmentCabin> (*lSegmentDate_ptr,
00138                                                              iCabinCode);
00139       SegmentCabinHelper::updateAvailabilities
       (lSegmentCabin);
00140     }
00141   }
00142
00143   // ////////////////////////////////////////////////////////////////////
00144   void FlightDateHelper::
00145   recalculateAvailability (const stdair::FlightDate&
      iFlightDate) {
00146     const stdair::SegmentDateList_T& lSegmentDateList =
00147       stdair::BomManager::getList<stdair::SegmentDate> (iFlightDate);
00148     for (stdair::SegmentDateList_T::const_iterator itSegmentDate =
00149            lSegmentDateList.begin(); itSegmentDate != lSegmentDateList.end();
00150          ++itSegmentDate) {
00151       const stdair::SegmentDate* lSegmentDate_ptr = *itSegmentDate;
00152       assert (lSegmentDate_ptr != NULL);
00153       const stdair::SegmentCabinList_T& lSegmentCabinList =
00154         stdair::BomManager::getList<stdair::SegmentCabin> (*lSegmentDate_ptr);
00155       for (stdair::SegmentCabinList_T::const_iterator itSegmentCabin =
00156              lSegmentCabinList.begin();
00157            itSegmentCabin != lSegmentCabinList.end(); ++itSegmentCabin) {
00158         const stdair::SegmentCabin* lSegmentCabin_ptr = *itSegmentCabin;
00159         assert (lSegmentCabin_ptr != NULL);
00160         SegmentCabinHelper::updateAvailabilities
       (*lSegmentCabin_ptr);
00161       }
00162     }
00163   }
00164
00165 }
```

23.67 airinv/bom/FlightDateHelper.hpp File Reference

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

Classes

- class AIRINV::FlightDateHelper

Namespaces

- namespace stdair

  Forward declarations.
- namespace AIRINV

23.68 FlightDateHelper.hpp

```
00001 #ifndef __AIRINV_BOM_FLIGHTDATEHELPER_HPP
00002 #define __AIRINV_BOM_FLIGHTDATEHELPER_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009
```

```
00010 // Forward declarations
00011 namespace stdair {
00012   class FlightDate;
00013 }
00014
00015 namespace AIRINV {
00016
00019   class FlightDateHelper {
00020   public:
00021     // ////////// Business Methods /////////
00024     static void fillFromRouting (const stdair::FlightDate&);
00025
00027     static void updateAvailability (const stdair::FlightDate&
      ,
00028                                     const stdair::SegmentCabin&,
00029                                     const stdair::PartySize_T& iNbOfBookings);
00030
00033     static void updateAvailabilityPool (const
      stdair::FlightDate&,
00034                                         const stdair::CabinCode_T&);
00035
00038     static void recalculateAvailability (const
      stdair::FlightDate&,
00039                                          const stdair::CabinCode_T&);
00040
00042     static void updateBookingControls (stdair::FlightDate&
      );
00043
00046     static void recalculateAvailability (const
      stdair::FlightDate&);
00047   };
00048
00049 }
00050 #endif // __AIRINV_BOM_FLIGHTDATEHELPER_HPP
```

23.69 airinv/bom/FlightDateStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/service/Logger.hpp>
#include <airinv/AIRINV_Types.hpp>
#include <airinv/bom/FlightDateStruct.hpp>
```

Namespaces

- namespace AIRINV

23.70 FlightDateStruct.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 // AIRINV
00011 #include <airinv/AIRINV_Types.hpp>
00012 #include <airinv/bom/FlightDateStruct.hpp>
00013
00014 namespace AIRINV {
00015
00016   // ////////////////////////////////////////////////////////////////////
00017   FlightDateStruct::FlightDateStruct ()
00018     : _flightDate (stdair::DEFAULT_DATE),
00019       _flightTypeCode (FlightTypeCode::DOMESTIC),
00020       _flightVisibilityCode (FlightVisibilityCode::NORMAL),
00021       _itSeconds (0), _legAlreadyDefined (false) {
00022   }
00023
00024   // ////////////////////////////////////////////////////////////////////
```

```
00025   stdair::Date_T FlightDateStruct::getDate() const {
00026     return stdair::Date_T (_itYear + 2000, _itMonth, _itDay
      );
00027   }
00028
00029   // ////////////////////////////////////////////////////////////////////
00030   stdair::Duration_T FlightDateStruct::getTime() const
       {
00031     return boost::posix_time::hours (_itHours)
00032       + boost::posix_time::minutes (_itMinutes)
00033       + boost::posix_time::seconds (_itSeconds);
00034   }
00035
00036   // ////////////////////////////////////////////////////////////////////
00037   const std::string FlightDateStruct::describe()
       const {
00038     std::ostringstream ostr;
00039     ostr << _airlineCode << _flightNumber << ", " <<
      _flightDate
00040          << " (" << _flightTypeCode;
00041     if (_flightVisibilityCode.getCode() !=
      FlightVisibilityCode::NORMAL) {
00042       ostr << "/" << _flightVisibilityCode;
00043     }
00044     ostr << ")" << std::endl;
00045
00046     for (LegStructList_T::const_iterator itLeg = _legList.begin();
00047          itLeg != _legList.end(); ++itLeg) {
00048       const LegStruct& lLeg = *itLeg;
00049       ostr << lLeg.describe();
00050     }
00051
00052     for (SegmentStructList_T::const_iterator itSegment = _segmentList
      .begin();
00053          itSegment != _segmentList.end(); ++itSegment) {
00054       const SegmentStruct& lSegment = *itSegment;
00055       ostr << lSegment.describe();
00056     }
00057
00058     //ostr << "[Debug] - Staging Leg: ";
00059     //ostr << _itLeg.describe();
00060     //ostr << "[Debug] - Staging Cabin: ";
00061     //ostr << _itCabin.describe();
00062
00063     return ostr.str();
00064   }
00065
00066   // ////////////////////////////////////////////////////////////////////
00067   void FlightDateStruct::addAirport (const
      stdair::AirportCode_T& iAirport) {
00068     AirportList_T::const_iterator itAirport = _airportList.find (
      iAirport);
00069     if (itAirport == _airportList.end()) {
00070       // Add the airport code to the airport set
00071       const bool insertSuccessful = _airportList.insert (iAirport).
      second;
00072
00073       if (insertSuccessful == false) {
00074         // TODO: throw an exception
00075       }
00076
00077       // Add the airport code to the airport vector
00078       _airportOrderedList.push_back (iAirport);
00079     }
00080   }
00081
00082   // ////////////////////////////////////////////////////////////////////
00083   void FlightDateStruct::buildSegments () {
00084     // The list of airports encompasses all the airports on which
00085     // the flight takes off or lands. Moreover, that list is
00086     // time-ordered: the first airport is the initial departure of
00087     // the flight, and the last airport is the eventual point of
00088     // rest of the flight.
00089     // Be l the size of the ordered list of airports.
00090     // We want to generate all the segment combinations from the legs
00091     // and, hence, from all the possible (time-ordered) airport pairs.
00092     // Thus, we both iterator on i=0...l-1 and j=i+1...l
00093     assert (_airportOrderedList.size() >= 2);
00094
00095     _segmentList.clear();
00096     for (AirportOrderedList_T::const_iterator itAirport_i =
00097            _airportOrderedList.begin();
00098          itAirport_i != _airportOrderedList.end()-1; ++
      itAirport_i) {
00099       for (AirportOrderedList_T::const_iterator itAirport_j = itAirport_i + 1;
00100            itAirport_j != _airportOrderedList.end(); ++
      itAirport_j) {
```

```
00101         SegmentStruct lSegmentStruct;
00102         lSegmentStruct._boardingPoint = *itAirport_i;
00103         lSegmentStruct._offPoint = *itAirport_j;
00104 
00105         _segmentList.push_back (lSegmentStruct);
00106       }
00107     }
00108 
00109     // Clear the lists of airports, so that it is ready for the next flight
00110     _airportList.clear();
00111     _airportOrderedList.clear();
00112   }
00113 
00114   // ////////////////////////////////////////////////////////////////////
00115   void FlightDateStruct::
00116   addSegmentCabin (const SegmentStruct& iSegment,
00117                    const SegmentCabinStruct& iCabin) {
00118     // Retrieve the Segment structure corresponding to the (boarding, off)
      point
00119     // pair.
00120     SegmentStructList_T::iterator itSegment = _segmentList.begin();
00121     for ( ; itSegment != _segmentList.end(); ++itSegment) {
00122       const SegmentStruct& lSegment = *itSegment;
00123 
00124       const stdair::AirportCode_T& lBoardingPoint = iSegment._boardingPoint
     ;
00125       const stdair::AirportCode_T& lOffPoint = iSegment._offPoint;
00126       if (lSegment._boardingPoint == lBoardingPoint
00127           && lSegment._offPoint == lOffPoint) {
00128         break;
00129       }
00130     }
00131 
00132     // If the segment key (airport pair) given in the schedule input file
00133     // does not correspond to the leg (boarding, off) points, throw an
      exception
00134     // so that the user knows the schedule input file is corrupted.
00135     if (itSegment == _segmentList.end()) {
00136       STDAIR_LOG_ERROR ("Within the inventory input file, there is a "
00137                         << "flight for which the airports of segments "
00138                         << "and those of the legs do not correspond.");
00139       throw SegmentDateNotFoundException ("Within
      the inventory input file, "
00140                                           "there is a flight for which the "
00141                                           "airports of segments and those of "
00142                                           "the legs do not correspond.");
00143     }
00144 
00145     // Add the Cabin structure to the Segment Cabin structure.
00146     assert (itSegment != _segmentList.end());
00147     SegmentStruct& lSegment = *itSegment;
00148     lSegment._cabinList.push_back (iCabin);
00149   }
00150 
00151   // ////////////////////////////////////////////////////////////////////
00152   void FlightDateStruct::
00153   addSegmentCabin (const SegmentCabinStruct&
      iCabin) {
00154     // Iterate on all the Segment structures (as they get the same cabin
00155     // definitions)
00156 
00157     for (SegmentStructList_T::iterator itSegment = _segmentList.
      begin();
00158          itSegment != _segmentList.end(); ++itSegment) {
00159       SegmentStruct& lSegment = *itSegment;
00160 
00161       lSegment._cabinList.push_back (iCabin);
00162     }
00163   }
00164 
00165   // ////////////////////////////////////////////////////////////////////
00166   void FlightDateStruct::
00167   addFareFamily (const SegmentStruct& iSegment,
00168                  const SegmentCabinStruct& iCabin,
00169                  const FareFamilyStruct& iFareFamily) {
00170     // Retrieve the Segment structure corresponding to the (boarding, off)
      point
00171     // pair.
00172     SegmentStructList_T::iterator itSegment = _segmentList.begin();
00173     for ( ; itSegment != _segmentList.end(); ++itSegment) {
00174       const SegmentStruct& lSegment = *itSegment;
00175 
00176       const stdair::AirportCode_T& lBoardingPoint = iSegment._boardingPoint
     ;
00177       const stdair::AirportCode_T& lOffPoint = iSegment._offPoint;
00178       if (lSegment._boardingPoint == lBoardingPoint
00179           && lSegment._offPoint == lOffPoint) {
```

```
00180         break;
00181       }
00182     }
00183
00184     // If the segment key (airport pair) given in the schedule input file
00185     // does not correspond to the leg (boarding, off) points, throw an
      exception
00186     // so that the user knows the schedule input file is corrupted.
00187     if (itSegment == _segmentList.end()) {
00188       STDAIR_LOG_ERROR ("Within the schedule input file, there is a flight "
00189                         << "for which the airports of segments and "
00190                         << "those of the legs do not correspond.");
00191       throw SegmentDateNotFoundException ("Within
      the schedule input file, "
00192                                           "there is a flight for which the "
00193                                           "airports of segments and those of "
00194                                           "the legs do not correspond.");
00195     }
00196
00197     // Add the Cabin structure to the Segment Cabin structure.
00198     assert (itSegment != _segmentList.end());
00199     SegmentStruct& lSegment = *itSegment;
00200
00201     // Retrieve the Segment cabin structure given the cabin code
00202     SegmentCabinStructList_T::iterator itCabin = lSegment._cabinList.
     begin();
00203     for ( ; itCabin != lSegment._cabinList.end(); ++itCabin) {
00204       const SegmentCabinStruct& lCabin = *itCabin;
00205
00206       const stdair::CabinCode_T& lCabinCode = lCabin._cabinCode;
00207       if (iCabin._cabinCode == lCabinCode) {
00208         break;
00209       }
00210     }
00211
00212     // If the segmentCabin key (cabin code) given in the schedule input file
00213     // does not correspond to the stored cabin codes, throw an exception
00214     // so that the user knows the schedule input file is corrupted.
00215     if (itCabin == lSegment._cabinList.end()) {
00216       STDAIR_LOG_ERROR ("Within the schedule input file, there is a flight "
00217                         << "for which the cabin code does not exist.");
00218       throw SegmentDateNotFoundException ("Within
      the schedule input file, "
00219                                           "there is a flight for which the "
00220                                           "cabin code does not exist.");
00221     }
00222
00223     // Add the Cabin structure to the Segment Cabin structure.
00224     assert (itCabin != lSegment._cabinList.end());
00225     SegmentCabinStruct& lCabin = *itCabin;
00226     lCabin._fareFamilies.push_back (iFareFamily);
00227   }
00228
00229   // ////////////////////////////////////////////////////////////////////
00230   void FlightDateStruct::
00231   addFareFamily (const SegmentCabinStruct&
     iCabin,
00232                  const FareFamilyStruct& iFareFamily) {
00233     // Iterate on all the Segment structures (as they get the same cabin
00234     // definitions)
00235
00236     for (SegmentStructList_T::iterator itSegment = _segmentList.
     begin();
00237          itSegment != _segmentList.end(); ++itSegment) {
00238       SegmentStruct& lSegment = *itSegment;
00239
00240       // Retrieve the Segment cabin structure given the cabin code
00241       SegmentCabinStructList_T::iterator itCabin = lSegment._cabinList
     .begin();
00242       for ( ; itCabin != lSegment._cabinList.end(); ++itCabin) {
00243         const SegmentCabinStruct& lCabin = *itCabin;
00244
00245         const stdair::CabinCode_T& lCabinCode = lCabin._cabinCode;
00246         if (iCabin._cabinCode == lCabinCode) {
00247           break;
00248         }
00249       }
00250
00251       // If the segmentCabin key (cabin code) given in the schedule input file
00252       // does not correspond to the stored cabin codes, throw an exception
00253       // so that the user knows the schedule input file is corrupted.
00254       if (itCabin == lSegment._cabinList.end()) {
00255         STDAIR_LOG_ERROR ("Within the schedule input file, there is a flight"
00256                           << " for which the cabin code does not exist.");
00257         throw SegmentDateNotFoundException ("Within
      the schedule input file, "
00258                                             "there is a flight for which the "
```

```
00259                                           "cabin code does not exist.");
00260       }
00261
00262       // Add the Cabin structure to the Segment Cabin structure.
00263       assert (itCabin != lSegment._cabinList.end());
00264       SegmentCabinStruct& lCabin = *itCabin;
00265       lCabin._fareFamilies.push_back (iFareFamily);
00266     }
00267   }
00268
00269 }
```

23.71 airinv/bom/FlightDateStruct.hpp File Reference

```
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/DoWStruct.hpp>
#include <airinv/basic/FlightTypeCode.hpp>
#include <airinv/basic/FlightVisibilityCode.hpp>
#include <airinv/bom/LegStruct.hpp>
#include <airinv/bom/LegCabinStruct.hpp>
#include <airinv/bom/BucketStruct.hpp>
#include <airinv/bom/SegmentStruct.hpp>
#include <airinv/bom/SegmentCabinStruct.hpp>
#include <airinv/bom/FareFamilyStruct.hpp>
#include <airinv/bom/AirportList.hpp>
```

Classes

- struct AIRINV::FlightDateStruct

Namespaces

- namespace AIRINV

23.72 FlightDateStruct.hpp

```
00001 #ifndef __AIRINV_BOM_FLIGHTDATESTRUCT_HPP
00002 #define __AIRINV_BOM_FLIGHTDATESTRUCT_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/stdair_inventory_types.hpp>
00011 #include <stdair/basic/StructAbstract.hpp>
00012 #include <stdair/bom/DoWStruct.hpp>
00013 // AirInv
00014 #include <airinv/basic/FlightTypeCode.hpp>
00015 #include <airinv/basic/FlightVisibilityCode.hpp
      >
00016 #include <airinv/bom/LegStruct.hpp>
00017 #include <airinv/bom/LegCabinStruct.hpp>
00018 #include <airinv/bom/BucketStruct.hpp>
00019 #include <airinv/bom/SegmentStruct.hpp>
00020 #include <airinv/bom/SegmentCabinStruct.hpp>
00021 #include <airinv/bom/FareFamilyStruct.hpp>
00022 #include <airinv/bom/AirportList.hpp>
00023
00024 namespace AIRINV {
00025
00027   struct FlightDateStruct : public stdair::StructAbstract {
00028
00030     stdair::Date_T getDate() const;
```

```
00031
00033     stdair::Duration_T getTime() const;
00034
00036     const std::string describe() const;
00037
00040     void addAirport (const stdair::AirportCode_T&);
00041
00043     void buildSegments ();
00044
00051     void addSegmentCabin (const SegmentStruct&,
00052                           const SegmentCabinStruct&);
00053
00059     void addSegmentCabin (const SegmentCabinStruct
      &);
00060
00067     void addFareFamily (const SegmentStruct&, const
      SegmentCabinStruct&,
00068                         const FareFamilyStruct&);
00069
00075     void addFareFamily (const SegmentCabinStruct
      &, const FareFamilyStruct&);
00076
00078     FlightDateStruct ();
00079
00080     // Attributes
00081     stdair::AirlineCode_T _airlineCode;
00082     stdair::FlightNumber_T _flightNumber;
00083     stdair::Date_T _flightDate;
00084     FlightTypeCode _flightTypeCode;
00085     FlightVisibilityCode _flightVisibilityCode
      ;
00086     LegStructList_T _legList;
00087     SegmentStructList_T _segmentList;
00088
00090     unsigned int _itYear;
00091     unsigned int _itMonth;
00092     unsigned int _itDay;
00093     int _dateOffSet;
00094
00096     long _itHours;
00097     long _itMinutes;
00098     long _itSeconds;
00099
00102     AirportList_T _airportList;
00103     AirportOrderedList_T _airportOrderedList
      ;
00104
00107     bool _legAlreadyDefined;
00108     LegStruct _itLeg;
00109     LegCabinStruct _itLegCabin;
00110     BucketStruct _itBucket;
00111
00113     bool _areSegmentDefinitionsSpecific;
00114     SegmentStruct _itSegment;
00115     SegmentCabinStruct _itSegmentCabin;
00116     BookingClassStruct _itBookingClass;
00117   };
00118
00119 }
00120 #endif // __AIRINV_BOM_FLIGHTDATESTRUCT_HPP
```

23.73 airinv/bom/FlightPeriodStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_Period_BOM.hpp>
#include <stdair/service/Logger.hpp>
#include <airinv/AIRINV_Types.hpp>
#include <airinv/bom/FlightPeriodStruct.hpp>
```

Namespaces

- namespace AIRINV

23.74 FlightPeriodStruct.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 // AIRINV
00011 #include <airinv/AIRINV_Types.hpp>
00012 #include <airinv/bom/FlightPeriodStruct.hpp>
00013
00014 namespace AIRINV {
00015
00016   // //////////////////////////////////////////////////////////////////////
00017   FlightPeriodStruct::FlightPeriodStruct
      ()
00018     : _dateRange (stdair::BOOST_DEFAULT_DATE_PERIOD),
00019       _dow (stdair::DEFAULT_DOW_STRING),
00020       _legAlreadyDefined (false), _itSeconds (0) {
00021   }
00022
00023   // //////////////////////////////////////////////////////////////////////
00024   stdair::Date_T FlightPeriodStruct::getDate() const
       {
00025     return stdair::Date_T (_itYear, _itMonth, _itDay);
00026   }
00027
00028   // //////////////////////////////////////////////////////////////////////
00029   stdair::Duration_T FlightPeriodStruct::getTime()
       const {
00030     return boost::posix_time::hours (_itHours)
00031       + boost::posix_time::minutes (_itMinutes)
00032       + boost::posix_time::seconds (_itSeconds);
00033   }
00034
00035   // //////////////////////////////////////////////////////////////////////
00036   const std::string FlightPeriodStruct::describe()
       const {
00037     std::ostringstream ostr;
00038     ostr << _airlineCode << _flightNumber << ", " <<
      _dateRange
00039          << " - " << _dow << std::endl;
00040
00041     for (LegStructList_T::const_iterator itLeg = _legList.begin();
00042          itLeg != _legList.end(); ++itLeg) {
00043       const LegStruct& lLeg = *itLeg;
00044       ostr << lLeg.describe();
00045     }
00046
00047     for (SegmentStructList_T::const_iterator itSegment = _segmentList
      .begin();
00048          itSegment != _segmentList.end(); ++itSegment) {
00049       const SegmentStruct& lSegment = *itSegment;
00050       ostr << lSegment.describe();
00051     }
00052
00053     //ostr << "[Debug] - Staging Leg: ";
00054     //ostr << _itLeg.describe();
00055     //ostr << "[Debug] - Staging Cabin: ";
00056     //ostr << _itCabin.describe();
00057
00058     return ostr.str();
00059   }
00060
00061   // //////////////////////////////////////////////////////////////////////
00062   void FlightPeriodStruct::addAirport (const
      stdair::AirportCode_T& iAirport) {
00063     AirportList_T::const_iterator itAirport = _airportList.find (
      iAirport);
00064     if (itAirport == _airportList.end()) {
00065       // Add the airport code to the airport set
00066       const bool insertSuccessful = _airportList.insert (iAirport).
      second;
00067
00068       if (insertSuccessful == false) {
00069         // TODO: throw an exception
00070       }
00071
00072       // Add the airport code to the airport vector
00073       _airportOrderedList.push_back (iAirport);
00074     }
00075   }
00076
```

```
00077   // ////////////////////////////////////////////////////////////////////
00078   void FlightPeriodStruct::buildSegments () {
00079     // The list of airports encompasses all the airports on which
00080     // the flight takes off or lands. Moreover, that list is
00081     // time-ordered: the first airport is the initial departure of
00082     // the flight, and the last airport is the eventual point of
00083     // rest of the flight.
00084     // Be l the size of the ordered list of airports.
00085     // We want to generate all the segment combinations from the legs
00086     // and, hence, from all the possible (time-ordered) airport pairs.
00087     // Thus, we both iterator on i=0...l-1 and j=i+1...l
00088     assert (_airportOrderedList.size() >= 2);
00089
00090     _segmentList.clear();
00091     for (AirportOrderedList_T::const_iterator itAirport_i =
00092            _airportOrderedList.begin();
00093          itAirport_i != _airportOrderedList.end()-1; ++
      itAirport_i) {
00094       for (AirportOrderedList_T::const_iterator itAirport_j = itAirport_i + 1;
00095            itAirport_j != _airportOrderedList.end(); ++
      itAirport_j) {
00096         SegmentStruct lSegmentStruct;
00097         lSegmentStruct._boardingPoint = *itAirport_i;
00098         lSegmentStruct._offPoint = *itAirport_j;
00099
00100         _segmentList.push_back (lSegmentStruct);
00101       }
00102     }
00103
00104     // Clear the lists of airports, so that it is ready for the next flight
00105     _airportList.clear();
00106     _airportOrderedList.clear();
00107   }
00108
00109   // ////////////////////////////////////////////////////////////////////
00110   void FlightPeriodStruct::
00111   addSegmentCabin (const SegmentStruct& iSegment,
00112                    const SegmentCabinStruct& iCabin) {
00113     // Retrieve the Segment structure corresponding to the (boarding, off)
       point
00114     // pair.
00115     SegmentStructList_T::iterator itSegment = _segmentList.begin();
00116     for ( ; itSegment != _segmentList.end(); ++itSegment) {
00117       const SegmentStruct& lSegment = *itSegment;
00118
00119       const stdair::AirportCode_T& lBoardingPoint = iSegment._boardingPoint
      ;
00120       const stdair::AirportCode_T& lOffPoint = iSegment._offPoint;
00121       if (lSegment._boardingPoint == lBoardingPoint
00122           && lSegment._offPoint == lOffPoint) {
00123         break;
00124       }
00125     }
00126
00127     // If the segment key (airport pair) given in the schedule input file
00128     // does not correspond to the leg (boarding, off) points, throw an
       exception
00129     // so that the user knows the schedule input file is corrupted.
00130     if (itSegment == _segmentList.end()) {
00131       STDAIR_LOG_ERROR ("Within the schedule input file, there is a "
00132                         << "flight for which the airports of segments "
00133                         << "and those of the legs do not correspond.");
00134       throw SegmentDateNotFoundException ("Within
       the schedule input file, "
00135                                           "there is a flight for which the "
00136                                           "airports of segments and those of "
00137                                           "the legs do not correspond.");
00138     }
00139
00140     // Add the Cabin structure to the Segment Cabin structure.
00141     assert (itSegment != _segmentList.end());
00142     SegmentStruct& lSegment = *itSegment;
00143     lSegment._cabinList.push_back (iCabin);
00144   }
00145
00146   // ////////////////////////////////////////////////////////////////////
00147   void FlightPeriodStruct::
00148   addSegmentCabin (const SegmentCabinStruct&
       iCabin) {
00149     // Iterate on all the Segment structures (as they get the same cabin
00150     // definitions)
00151     for (SegmentStructList_T::iterator itSegment = _segmentList.
      begin();
00152          itSegment != _segmentList.end(); ++itSegment) {
00153       SegmentStruct& lSegment = *itSegment;
00154
00155       lSegment._cabinList.push_back (iCabin);
```

```
00156     }
00157   }
00158
00159   // ////////////////////////////////////////////////////////////////////
00160   void FlightPeriodStruct::
00161   addFareFamily (const SegmentStruct& iSegment,
00162                  const SegmentCabinStruct& iCabin,
00163                  const FareFamilyStruct& iFareFamily) {
00164     // Retrieve the Segment structure corresponding to the (boarding, off)
       point
00165     // pair.
00166     SegmentStructList_T::iterator itSegment = _segmentList.begin();
00167     for ( ; itSegment != _segmentList.end(); ++itSegment) {
00168       const SegmentStruct& lSegment = *itSegment;
00169
00170       const stdair::AirportCode_T& lBoardingPoint = iSegment._boardingPoint
      ;
00171       const stdair::AirportCode_T& lOffPoint = iSegment._offPoint;
00172       if (lSegment._boardingPoint == lBoardingPoint
00173           && lSegment._offPoint == lOffPoint) {
00174         break;
00175       }
00176     }
00177
00178     // If the segment key (airport pair) given in the schedule input file
00179     // does not correspond to the leg (boarding, off) points, throw an
       exception
00180     // so that the user knows the schedule input file is corrupted.
00181     if (itSegment == _segmentList.end()) {
00182       STDAIR_LOG_ERROR ("Within the schedule input file, there is a flight "
00183                         << "for which the airports of segments and "
00184                         << "those of the legs do not correspond.");
00185       throw SegmentDateNotFoundException ("Within
       the schedule input file, "
00186                                           "there is a flight for which the "
00187                                           "airports of segments and those of "
00188                                           "the legs do not correspond.");
00189     }
00190
00191     // Add the Cabin structure to the Segment Cabin structure.
00192     assert (itSegment != _segmentList.end());
00193     SegmentStruct& lSegment = *itSegment;
00194
00195     // Retrieve the Segment cabin structure given the cabin code
00196     SegmentCabinStructList_T::iterator itCabin = lSegment._cabinList.
      begin();
00197     for ( ; itCabin != lSegment._cabinList.end(); ++itCabin) {
00198       const SegmentCabinStruct& lCabin = *itCabin;
00199
00200       const stdair::CabinCode_T& lCabinCode = lCabin._cabinCode;
00201       if (iCabin._cabinCode == lCabinCode) {
00202         break;
00203       }
00204     }
00205
00206     // If the segmentCabin key (cabin code) given in the schedule input file
00207     // does not correspond to the stored cabin codes, throw an exception
00208     // so that the user knows the schedule input file is corrupted.
00209     if (itCabin == lSegment._cabinList.end()) {
00210       STDAIR_LOG_ERROR ("Within the schedule input file, there is a flight "
00211                         << "for which the cabin code does not exist.");
00212       throw SegmentDateNotFoundException ("Within
       the schedule input file, "
00213                                           "there is a flight for which the "
00214                                           "cabin code does not exist.");
00215     }
00216
00217     // Add the Cabin structure to the Segment Cabin structure.
00218     assert (itCabin != lSegment._cabinList.end());
00219     SegmentCabinStruct& lCabin = *itCabin;
00220     lCabin._fareFamilies.push_back(iFareFamily);
00221   }
00222
00223   // ////////////////////////////////////////////////////////////////////
00224   void FlightPeriodStruct::
00225   addFareFamily (const SegmentCabinStruct&
      iCabin,
00226                  const FareFamilyStruct& iFareFamily) {
00227     // Iterate on all the Segment structures (as they get the same cabin
00228     // definitions)
00229
00230     for (SegmentStructList_T::iterator itSegment = _segmentList.
      begin();
00231          itSegment != _segmentList.end(); ++itSegment) {
00232       SegmentStruct& lSegment = *itSegment;
00233
00234       // Retrieve the Segment cabin structure given the cabin code
```

```
00235       SegmentCabinStructList_T::iterator itCabin = lSegment._cabinList
      .begin();
00236       for ( ; itCabin != lSegment._cabinList.end(); ++itCabin) {
00237         const SegmentCabinStruct& lCabin = *itCabin;
00238
00239         const stdair::CabinCode_T& lCabinCode = lCabin._cabinCode;
00240         if (iCabin._cabinCode == lCabinCode) {
00241           break;
00242         }
00243       }
00244
00245       // If the segmentCabin key (cabin code) given in the schedule input file
00246       // does not correspond to the stored cabin codes, throw an exception
00247       // so that the user knows the schedule input file is corrupted.
00248       if (itCabin == lSegment._cabinList.end()) {
00249         STDAIR_LOG_ERROR ("Within the schedule input file, there is a flight"
00250                           << " for which the cabin code does not exist.");
00251         throw SegmentDateNotFoundException ("Within
      the schedule input file, "
00252                                             "there is a flight for which the "
00253                                             "cabin code does not exist.");
00254       }
00255
00256       // Add the Cabin structure to the Segment Cabin structure.
00257       assert (itCabin != lSegment._cabinList.end());
00258       SegmentCabinStruct& lCabin = *itCabin;
00259       lCabin._fareFamilies.push_back(iFareFamily);
00260     }
00261   }
00262
00263 }
```

23.75 airinv/bom/FlightPeriodStruct.hpp File Reference

```
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/DoWStruct.hpp>
#include <airinv/bom/LegCabinStruct.hpp>
#include <airinv/bom/LegStruct.hpp>
#include <airinv/bom/SegmentStruct.hpp>
#include <airinv/bom/SegmentCabinStruct.hpp>
#include <airinv/bom/FareFamilyStruct.hpp>
#include <airinv/bom/AirportList.hpp>
```

Classes

- struct AIRINV::FlightPeriodStruct

Namespaces

- namespace AIRINV

23.76 FlightPeriodStruct.hpp

```
00001 #ifndef __AIRINV_BOM_FLIGHTPERIODSTRUCT_HPP
00002 #define __AIRINV_BOM_FLIGHTPERIODSTRUCT_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/stdair_inventory_types.hpp>
00011 #include <stdair/basic/StructAbstract.hpp>
00012 #include <stdair/bom/DoWStruct.hpp>
00013 // AirInv
00014 #include <airinv/bom/LegCabinStruct.hpp>
```

```
00015 #include <airinv/bom/LegStruct.hpp>
00016 #include <airinv/bom/SegmentStruct.hpp>
00017 #include <airinv/bom/SegmentCabinStruct.hpp>
00018 #include <airinv/bom/FareFamilyStruct.hpp>
00019 #include <airinv/bom/AirportList.hpp>
00020
00021 namespace AIRINV {
00022
00024   struct FlightPeriodStruct : public stdair::StructAbstract {
00025
00027     stdair::Date_T getDate() const;
00028
00030     stdair::Duration_T getTime() const;
00031
00033     const std::string describe() const;
00034
00037     void addAirport (const stdair::AirportCode_T&);
00038
00040     void buildSegments ();
00041
00048     void addSegmentCabin (const SegmentStruct&,
00049                           const SegmentCabinStruct&);
00050
00056     void addSegmentCabin (const SegmentCabinStruct
      &);
00057
00064     void addFareFamily (const SegmentStruct&,
00065                         const SegmentCabinStruct&,
00066                         const FareFamilyStruct&);
00067
00073     void addFareFamily (const SegmentCabinStruct
      &,
00074                         const FareFamilyStruct&);
00075
00077     FlightPeriodStruct ();
00078
00079     // Attributes
00080     stdair::AirlineCode_T _airlineCode;
00081     stdair::FlightNumber_T _flightNumber;
00082     stdair::DatePeriod_T _dateRange;
00083     stdair::DoWStruct _dow;
00084     LegStructList_T _legList;
00085     SegmentStructList_T _segmentList;
00086
00089     bool _legAlreadyDefined;
00090     LegStruct _itLeg;
00091     LegCabinStruct _itLegCabin;
00092
00094     stdair::Date_T _dateRangeStart;
00095     stdair::Date_T _dateRangeEnd;
00096     unsigned int _itYear;
00097     unsigned int _itMonth;
00098     unsigned int _itDay;
00099     int _dateOffset;
00100
00102     long _itHours;
00103     long _itMinutes;
00104     long _itSeconds;
00105
00108     AirportList_T _airportList;
00109     AirportOrderedList_T _airportOrderedList
      ;
00110
00112     bool _areSegmentDefinitionsSpecific;
00113     SegmentStruct _itSegment;
00114     SegmentCabinStruct _itSegmentCabin;
00115   };
00116
00117 }
00118 #endif // __AIRINV_BOM_FLIGHTPERIODSTRUCT_HPP
```

23.77 airinv/bom/FRAT5Struct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/service/Logger.hpp>
#include <airinv/bom/FRAT5Struct.hpp>
```

Namespaces

- namespace AIRINV

23.78 FRAT5Struct.cpp

```
00001 // //////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/service/Logger.hpp>
00009 // AIRINV
00010 #include <airinv/bom/FRAT5Struct.hpp>
00011
00012 namespace AIRINV {
00013
00014   // //////////////////////////////////////////////////////////////////////
00015   FRAT5Struct::FRAT5Struct() {
00016   }
00017
00018   // //////////////////////////////////////////////////////////////////////
00019   FRAT5Struct::~FRAT5Struct() {
00020   }
00021
00022   // //////////////////////////////////////////////////////////////////////
00023   const std::string FRAT5Struct::describe() const {
00024     std::ostringstream oStr;
00025     oStr << _key << "; ";
00026     for (stdair::FRAT5Curve_T::const_reverse_iterator itFRAT5 = _curve.
      rbegin();
00027          itFRAT5 != _curve.rend(); ++itFRAT5) {
00028       const stdair::DTD_T& lDTD = itFRAT5->first;
00029       const double& lFRAT5 = itFRAT5->second;
00030       oStr << lDTD << ":" << lFRAT5 << ";";
00031     }
00032
00033     return oStr.str();
00034   }
00035
00036 }
```

23.79 airinv/bom/FRAT5Struct.hpp File Reference

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

Classes

- struct AIRINV::FRAT5Struct

Namespaces

- namespace AIRINV

23.80 FRAT5Struct.hpp

```
00001 #ifndef __AIRINV_BOM_FRAT5STRUCT_HPP
00002 #define __AIRINV_BOM_FRAT5STRUCT_HPP
00003
00004 // //////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/stdair_rm_types.hpp>
```

```
00011
00012 namespace AIRINV {
00013
00015   struct FRAT5Struct : public stdair::StructAbstract {
00016
00017   public:
00018     // ////////////////// Display Support Methods ////////////////
00020     const std::string describe() const;
00021
00022
00023   public:
00024     // /////////////// Constructors and destructors ///////////////
00026     FRAT5Struct();
00028     ~FRAT5Struct();
00029   private:
00031     FRAT5Struct (const FRAT5Struct&);
00032
00033
00034   public:
00035     // ////////////// Attributes ////////////////////
00037     std::string _key;
00038
00040     stdair::FRAT5Curve_T _curve;
00041
00042   public:
00043     // ///////////////// Staging ////////////////////
00045     stdair::DTD_T _dtd;
00046   };
00047
00048 }
00049 #endif // __AIRINV_BOM_FRAT5STRUCT_HPP
```

23.81 airinv/bom/InventoryHelper.cpp File Reference

```
#include <cassert>
#include <stdair/bom/BomRetriever.hpp>
#include <stdair/bom/BomManager.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/BookingClass.hpp>
#include <stdair/bom/SegmentSnapshotTable.hpp>
#include <stdair/bom/TravelSolutionStruct.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <airinv/bom/InventoryHelper.hpp>
#include <airinv/bom/FlightDateHelper.hpp>
#include <airinv/bom/SegmentSnapshotTableHelper.hpp>
#include <airinv/bom/SegmentCabinHelper.hpp>
```

Namespaces

- namespace AIRINV

23.82 InventoryHelper.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // StdAir
00007 #include <stdair/bom/BomRetriever.hpp>
00008 #include <stdair/bom/BomManager.hpp>
00009 #include <stdair/bom/Inventory.hpp>
00010 #include <stdair/bom/FlightDate.hpp>
```

```
00011 #include <stdair/bom/SegmentDate.hpp>
00012 #include <stdair/bom/SegmentCabin.hpp>
00013 #include <stdair/bom/FareFamily.hpp>
00014 #include <stdair/bom/BookingClass.hpp>
00015 #include <stdair/bom/SegmentSnapshotTable.hpp>
00016 #include <stdair/bom/TravelSolutionStruct.hpp>
00017 #include <stdair/service/Logger.hpp>
00018 #include <stdair/bom/LegCabin.hpp>
00019 // AirInv
00020 #include <airinv/bom/InventoryHelper.hpp>
00021 #include <airinv/bom/FlightDateHelper.hpp>
00022 #include <airinv/bom/SegmentSnapshotTableHelper.hpp
      >
00023 #include <airinv/bom/SegmentCabinHelper.hpp>
00024
00025 namespace AIRINV {
00026
00027   // ////////////////////////////////////////////////////////////////////
00028   void InventoryHelper::fillFromRouting (const
      stdair::Inventory& iInventory) {
00029     const stdair::FlightDateList_T& lFlightDateList =
00030       stdair::BomManager::getList<stdair::FlightDate> (iInventory);
00031
00032     // Browse the list of flight-dates and update each flight-date.
00033     for (stdair::FlightDateList_T::const_iterator itFlightDate =
00034            lFlightDateList.begin();
00035          itFlightDate != lFlightDateList.end(); ++itFlightDate) {
00036       const stdair::FlightDate* lCurrentFlightDate_ptr = *itFlightDate;
00037       assert (lCurrentFlightDate_ptr != NULL);
00038       FlightDateHelper::fillFromRouting (*
      lCurrentFlightDate_ptr);
00039     }
00040   }
00041
00042   // ////////////////////////////////////////////////////////////////////
00043   void InventoryHelper::
00044   calculateAvailability (const stdair::Inventory&
      iInventory,
00045                          const std::string& iFullSegmentDateKey,
00046                          stdair::TravelSolutionStruct& ioTravelSolution) {
00047
00048     // Create the map of class/availability for the given segment date.
00049     stdair::ClassAvailabilityMap_T lClassAvailabilityMap;
00050
00051     // Create the map of class/object ID for the given segment date.
00052     stdair::ClassObjectIDMap_T lClassObjectIDMap;
00053
00054     // DEBUG
00055     STDAIR_LOG_DEBUG (iFullSegmentDateKey);
00056     //
00057     stdair::SegmentDate* lSegmentDate_ptr =
00058       stdair::BomRetriever::retrieveSegmentDateFromLongKey(iInventory,
00059                                                            iFullSegmentDateKey)
      ;
00060     assert (lSegmentDate_ptr != NULL);
00061
00062     // Browse the segment-cabins and fill the map with the availability of
00063     // each booking class.
00064     const stdair::SegmentCabinList_T& lSegmentCabinList =
00065       stdair::BomManager::getList<stdair::SegmentCabin> (*lSegmentDate_ptr);
00066     for (stdair::SegmentCabinList_T::const_iterator itCabin =
00067            lSegmentCabinList.begin();
00068          itCabin != lSegmentCabinList.end(); ++itCabin) {
00069       stdair::SegmentCabin* lSegmentCabin_ptr = *itCabin;
00070       assert (lSegmentCabin_ptr != NULL);
00071
00072
00073       // Compute the availability using the AU and the cumulative
00074       // booking counter.
00075       // SegmentCabinHelper::updateAvailabilities (*lSegmentCabin_ptr);
00076       const stdair::BookingClassList_T& lBCList =
00077         stdair::BomManager::getList<stdair::BookingClass> (*lSegmentCabin_ptr);
00078       for (stdair::BookingClassList_T::const_reverse_iterator itBC =
00079            lBCList.rbegin(); itBC != lBCList.rend(); ++itBC) {
00080         stdair::BookingClass* lBC_ptr = *itBC;
00081         assert (lBC_ptr != NULL);
00082
00083         const stdair::Availability_T lAvl = lBC_ptr->getSegmentAvailability();
00084
00085         const stdair::ClassCode_T& lClassCode = lBC_ptr->getClassCode();

00086         bool insertSuccessful = lClassAvailabilityMap.
00087           insert (stdair::ClassAvailabilityMap_T::value_type (lClassCode,
00088                                                               lAvl)).second;
00089         assert (insertSuccessful == true);
00090
00091         stdair::BookingClassID_T lBCID (*lBC_ptr);
```

```
00092         insertSuccessful = lClassObjectIDMap.
00093           insert (stdair::ClassObjectIDMap_T::value_type (lClassCode,
00094                                                           lBCID)).second;
00095         assert (insertSuccessful == true);
00096       }
00097     }
00098
00099     //
00100     ioTravelSolution.addClassAvailabilityMap (lClassAvailabilityMap);
00101     ioTravelSolution.addClassObjectIDMap (lClassObjectIDMap);
00102   }
00103
00104
00105   // ////////////////////////////////////////////////////////////////////
00106   void InventoryHelper::
00107   getYieldAndBidPrice (const stdair::Inventory& iInventory
      ,
00108                        const std::string& iFullSegmentDateKey,
00109                        stdair::TravelSolutionStruct& ioTravelSolution) {
00110
00111     // Create the map of class/availability for the given segment date.
00112     // stdair::ClassAvailabilityMap_T lClassAvailabilityMap;
00113
00114     stdair::ClassYieldMap_T lClassYieldMap;
00115
00116     stdair::ClassBpvMap_T lClassBpvMap;
00117
00118     // DEBUG
00119     STDAIR_LOG_DEBUG (iFullSegmentDateKey);
00120     //
00121     stdair::SegmentDate* lSegmentDate_ptr =
00122       stdair::BomRetriever::retrieveSegmentDateFromLongKey (iInventory,
00123                                                             iFullSegmentDateKey
      );
00124     assert (lSegmentDate_ptr != NULL);
00125
00126     // Browse the segment-cabins and fill the maps with the bid price vector
       reference
00127     // and yield of each booking class.
00128     const stdair::SegmentCabinList_T& lSegmentCabinList =
00129       stdair::BomManager::getList<stdair::SegmentCabin> (*lSegmentDate_ptr);
00130     for (stdair::SegmentCabinList_T::const_iterator itCabin =
00131            lSegmentCabinList.begin();
00132          itCabin != lSegmentCabinList.end(); ++itCabin) {
00133       stdair::SegmentCabin* lSegmentCabin_ptr = *itCabin;
00134       assert (lSegmentCabin_ptr != NULL);
00135
00136       stdair::BidPriceVector_T lBPV;
00137
00138
00139       //stdair::BidPriceVector_T lBPV;
00140       stdair::LegCabinList_T lLegCabinList =
00141         stdair::BomManager::getList<stdair::LegCabin> (*lSegmentCabin_ptr);
00142       assert (!lLegCabinList.empty());
00143       if (lLegCabinList.size() > 1) {
00144         // Compute the sum of bid prices and return a vector containing that
       value.
00145         stdair::BidPrice_T lBidPriceValue = 0;
00146         for (stdair::LegCabinList_T::const_iterator itLC = lLegCabinList.begin(
      );
00147              itLC != lLegCabinList.end(); ++itLC) {
00148           const stdair::LegCabin* lLegCabin_ptr = *itLC;
00149           const stdair::BidPriceVector_T& lLegCabinBPV = lLegCabin_ptr->
      getBidPriceVector();
00150           if (!lLegCabinBPV.empty()) {
00151             lBidPriceValue += lLegCabinBPV.back();
00152           } else {
00153             // If the remaining capacity is zero (empty bid price vector) on
       one of the legs,
00154             // then the remaining capacity of the segment is also zero (return
       an empty bid price).
00155             lBidPriceValue = std::numeric_limits<stdair::BidPrice_T>::max();
00156             break;
00157           }
00158         }
00159         if (lBidPriceValue < std::numeric_limits<stdair::BidPrice_T>::max()) {
00160           lBPV.push_back(lBidPriceValue);
00161         }
00162
00163       } else {
00164         const stdair::LegCabin* lLegCabin_ptr = lLegCabinList.front();
00165         lBPV = lLegCabin_ptr->getBidPriceVector();
00166       }
00167
00168
00169       // const stdair::CabinCapacity_T& lCabinCapacity =
       lSegmentCabin_ptr->getCapacity();
```

```
00170       // const stdair::CommittedSpace_T& lCommittedSpace =
      lSegmentCabin_ptr->getCommittedSpace();
00171       // assert (lCabinCapacity - lCommittedSpace > 0);
00172       // lBPV.resize(lCabinCapacity - lCommittedSpace);
00173
00174       const stdair::Availability_T& lAvailabilityPool =
00175         lSegmentCabin_ptr->getAvailabilityPool();
00176       //assert (lAvailabilityPool > 0);
00177
00178       if (lAvailabilityPool < lBPV.size()) {
00179         lBPV.resize(lAvailabilityPool);
00180       }
00181
00182
00183       //
00184       ioTravelSolution.addBidPriceVector (lBPV);
00185
00186       const stdair::BidPriceVectorHolder_T& lBidPriceVectorHolder =
00187         ioTravelSolution.getBidPriceVectorHolder();
00188       const stdair::BidPriceVectorHolder_T::const_reverse_iterator itBPV =
00189         lBidPriceVectorHolder.rbegin();
00190       const stdair::BidPriceVector_T& lBpvRef = *itBPV;
00191
00192       const stdair::FareFamilyList_T& lFFList =
00193         stdair::BomManager::getList<stdair::FareFamily> (*lSegmentCabin_ptr);
00194       for (stdair::FareFamilyList_T::const_iterator itFF = lFFList.begin();
00195            itFF != lFFList.end(); ++itFF) {
00196         const stdair::FareFamily* lFareFamily_ptr = *itFF;
00197         assert (lFareFamily_ptr != NULL);
00198
00199         const stdair::BookingClassList_T& lBCList =
00200           stdair::BomManager::getList<stdair::BookingClass> (*lFareFamily_ptr);
00201         for (stdair::BookingClassList_T::const_iterator itBC = lBCList.begin();
00202              itBC != lBCList.end(); ++itBC) {
00203           const stdair::BookingClass* lBC_ptr = *itBC;
00204           assert (lBC_ptr != NULL);
00205
00206           const stdair::ClassCode_T& lClassCode = lBC_ptr->getClassCode();
00207
00208           const stdair::YieldValue_T lYld = lBC_ptr->getYield() ;
00209           const bool insertYieldMapSuccessful = lClassYieldMap.
00210             insert (stdair::ClassYieldMap_T::value_type (lClassCode,
00211                                                          lYld)).second;
00212           assert (insertYieldMapSuccessful == true);
00213
00214           const bool insertBpvMapSuccessful = lClassBpvMap.
00215             insert (stdair::ClassBpvMap_T::value_type (lClassCode,
00216                                                        &lBpvRef)).second;
00217           assert (insertBpvMapSuccessful == true);
00218
00219           // DEBUG
00220           // STDAIR_LOG_DEBUG ("Class: " << lClassCode
00221           //                   << ", " << "Yield: " << lYld << ", "
00222           //                   << "Bid price: " << lBpvRef.back() << ", "
00223           //                   << "Remaining capacity: "
00224           //                   << lCabinCapacity - lCommittedSpace);
00225
00226           //
00227           stdair::BidPrice_T lBpvVal = std::numeric_limits<double>::max();
00228           if (lBpvRef.empty() == false) {
00229             lBpvVal = lBpvRef.back();
00230           }
00231
00232           //lBpvVal = boost::lexical_cast<std::string> (lBpvRef.back());
00233           STDAIR_LOG_DEBUG ("Class: " << lClassCode
00234                             << ", " << "Yield: " << lYld << ", "
00235                             << "Bid price: " << lBpvVal << ", "
00236                             << "Remaining capacity: " << lAvailabilityPool
00237                             << " Segment date: " << iFullSegmentDateKey);
00238         }
00239       }
00240     }
00241
00242     //
00243     ioTravelSolution.addClassYieldMap (lClassYieldMap);
00244     ioTravelSolution.addClassBpvMap (lClassBpvMap);
00245   }
00246
00247   // ////////////////////////////////////////////////////////////////////
00248   bool InventoryHelper::sell (stdair::Inventory&
      ioInventory,
00249                               const std::string& iFullSegmentDateKey,
00250                               const stdair::ClassCode_T& iClassCode,
00251                               const stdair::PartySize_T& iPartySize) {
00252     bool hasSaleBeenSuccessful = false;
00253
00254     // DEBUG
```

```
00255     STDAIR_LOG_DEBUG ("Full key: '" << iFullSegmentDateKey
00256                       << "', " << iClassCode);
00257
00258     //
00259     stdair::BookingClass* lBookingClass_ptr =
00260       stdair::BomRetriever::retrieveBookingClassFromLongKey(ioInventory,
00261                                                             iFullSegmentDateKey
      ,
00262                                                             iClassCode);
00263
00264     // DEBUG
00265     const std::string hasFoundBookingClassStr =
00266       (lBookingClass_ptr != NULL)?"Yes":"No";
00267     STDAIR_LOG_DEBUG ("Found booking class? " << hasFoundBookingClassStr);
00268
00269     if (lBookingClass_ptr != NULL) {
00270
00271       // Register the sale in the class.
00272       hasSaleBeenSuccessful =
00273         InventoryHelper::sell (*lBookingClass_ptr,
      iPartySize);
00274
00275       return hasSaleBeenSuccessful;
00276     }
00277
00278     return hasSaleBeenSuccessful;
00279   }
00280
00281   // ////////////////////////////////////////////////////////////////////
00282   bool InventoryHelper::sell (const
      stdair::BookingClassID_T& iClassID,
00283                               const stdair::PartySize_T& iPartySize) {
00284
00285     //
00286     stdair::BookingClass& lBookingClass = iClassID.getObject();
00287
00288     // Register the sale in the class.
00289     const bool hasSaleBeenSuccessful =
00290       InventoryHelper::sell (lBookingClass, iPartySize);
00291
00292     return hasSaleBeenSuccessful;
00293   }
00294
00295   // ////////////////////////////////////////////////////////////////////
00296   bool InventoryHelper::sell (stdair::BookingClass&
      iBookingClass,
00297                               const stdair::PartySize_T& iPartySize) {
00298
00299     // Register the sale in the class.
00300     iBookingClass.sell (iPartySize);
00301
00302     //
00303     stdair::FareFamily& lFareFamily =
00304       stdair::BomManager::getParent<stdair::FareFamily> (iBookingClass);
00305
00306     //
00307     stdair::SegmentCabin& lSegmentCabin =
00308       stdair::BomManager::getParent<stdair::SegmentCabin> (lFareFamily);
00309
00310     //
00311     stdair::SegmentDate& lSegmentDate =
00312       stdair::BomManager::getParent<stdair::SegmentDate,
00313                                     stdair::SegmentCabin> (lSegmentCabin);
00314
00315     //
00316     stdair::FlightDate& lFlightDate =
00317       stdair::BomManager::getParent<stdair::FlightDate,
00318                                     stdair::SegmentDate> (lSegmentDate);
00319
00320     // Update the commited space of the segment-cabins and the leg-cabins.
00321     SegmentCabinHelper::updateFromReservation
       (lFlightDate, lSegmentCabin,
00322                                               iPartySize);
00323
00324     return true;
00325   }
00326
00327   // ////////////////////////////////////////////////////////////////////
00328   bool InventoryHelper::cancel (stdair::Inventory&
      ioInventory,
00329                                 const std::string& iFullSegmentDateKey,
00330                                 const stdair::ClassCode_T& iClassCode,
00331                                 const stdair::PartySize_T& iPartySize) {
00332     bool hasCancellationBeenSuccessful = false;
00333
00334     // DEBUG
00335     STDAIR_LOG_DEBUG ("Full key: '" << iFullSegmentDateKey
```

```
00336                       << "', " << iClassCode);
00337
00338     //
00339     stdair::BookingClass* lBookingClass_ptr =
00340       stdair::BomRetriever::retrieveBookingClassFromLongKey(ioInventory,
00341                                                             iFullSegmentDateKey
    ,
00342                                                             iClassCode);
00343
00344     // DEBUG
00345     const std::string hasFoundBookingClassStr =
00346       (lBookingClass_ptr != NULL)?"Yes":"No";
00347     STDAIR_LOG_DEBUG ("Found booking class? " << hasFoundBookingClassStr);
00348
00349     if (lBookingClass_ptr != NULL) {
00350
00351       // Register the cancellation in the class.
00352       hasCancellationBeenSuccessful =
00353         cancel (*lBookingClass_ptr, iPartySize);
00354
00355       return hasCancellationBeenSuccessful;
00356     }
00357
00358     return hasCancellationBeenSuccessful;
00359   }
00360
00361   // ////////////////////////////////////////////////////////////////////
00362   bool InventoryHelper::cancel (const
    stdair::BookingClassID_T& iClassID,
00363                                 const stdair::PartySize_T& iPartySize) {
00364
00365     stdair::BookingClass& lBookingClass = iClassID.getObject();
00366
00367     // Register the cancellation in the class.
00368     const bool hasCancellationBeenSuccessful =
00369       cancel (lBookingClass, iPartySize);
00370
00371     return hasCancellationBeenSuccessful;
00372   }
00373
00374   // ////////////////////////////////////////////////////////////////////
00375   bool InventoryHelper::cancel (stdair::BookingClass&
    iBookingClass,
00376                                 const stdair::PartySize_T& iPartySize) {
00377
00378     // Register the cancellation in the class.
00379     iBookingClass.cancel (iPartySize);
00380
00381     //
00382     stdair::FareFamily& lFareFamily =
00383       stdair::BomManager::getParent<stdair::FareFamily> (iBookingClass);
00384
00385     //
00386     stdair::SegmentCabin& lSegmentCabin =
00387       stdair::BomManager::getParent<stdair::SegmentCabin> (lFareFamily);
00388
00389     //
00390     stdair::SegmentDate& lSegmentDate =
00391       stdair::BomManager::getParent<stdair::SegmentDate,
00392                                     stdair::SegmentCabin> (lSegmentCabin);
00393
00394     //
00395     stdair::FlightDate& lFlightDate =
00396       stdair::BomManager::getParent<stdair::FlightDate,
00397                                     stdair::SegmentDate> (lSegmentDate);
00398
00399     // Update the commited space of the segment-cabins and the leg-cabins.
00400     SegmentCabinHelper::updateFromReservation
     (lFlightDate, lSegmentCabin,
00401                                               -iPartySize);
00402
00403     return true;
00404   }
00405
00406   // ////////////////////////////////////////////////////////////////////
00407   void InventoryHelper::takeSnapshots(const
    stdair::Inventory& iInventory,
00408                                       const stdair::DateTime_T& iSnapshotTime)
    {
00409     // Browse the guillotine block list and take the snapshots for
00410     // each guillotine.
00411     const stdair::SegmentSnapshotTableList_T& lSegmentSnapshotTableList =
00412       stdair::BomManager::getList<stdair::SegmentSnapshotTable> (iInventory);
00413     for (stdair::SegmentSnapshotTableList_T::const_iterator itGB =
00414            lSegmentSnapshotTableList.begin();
00415          itGB != lSegmentSnapshotTableList.end(); ++itGB) {
00416       stdair::SegmentSnapshotTable* lSegmentSnapshotTable_ptr = *itGB;
```

```
00417
00418       SegmentSnapshotTableHelper::takeSnapshots
    (*lSegmentSnapshotTable_ptr,
00419                                                 iSnapshotTime);
00420     }
00421   }
00422 }
```

23.83 airinv/bom/InventoryHelper.hpp File Reference

```
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/BomIDTypes.hpp>
```

Classes

- class AIRINV::InventoryHelper

Namespaces

- namespace stdair

  Forward declarations.
- namespace AIRINV

23.84 InventoryHelper.hpp

```
00001 #ifndef __AIRINV_BOM_INVENTORYHELPER_HPP
00002 #define __AIRINV_BOM_INVENTORYHELPER_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/stdair_basic_types.hpp>
00011 #include <stdair/bom/BomIDTypes.hpp>
00012
00013 // Forward declarations
00014 namespace stdair {
00015   struct TravelSolutionStruct;
00016   class Inventory;
00017 }
00018
00019 namespace AIRINV {
00020
00023   class InventoryHelper {
00024   public:
00025     // ////////// Business Methods /////////
00028     static void fillFromRouting (const stdair::Inventory&);
00029
00031     static void calculateAvailability (const
    stdair::Inventory&,
00032                                        const std::string&,
00033                                        stdair::TravelSolutionStruct&);
00034
00036     static void getYieldAndBidPrice (const stdair::Inventory
    &,
00037                                      const std::string&,
00038                                      stdair::TravelSolutionStruct&);
00039
00041     static bool sell (stdair::Inventory&, const std::string&
    iSegmentDateKey,
00042                       const stdair::ClassCode_T&, const stdair::PartySize_T&);
00043
00045     static bool sell (const stdair::BookingClassID_T&,
00046                       const stdair::PartySize_T&);
00047
00049     static bool cancel (stdair::Inventory&, const std::string&
    iSegmentDateKey,
00050                         const stdair::ClassCode_T&, const stdair::PartySize_T&)
```

```
    ;
00051
00053     static bool cancel (const stdair::BookingClassID_T&,
00054                         const stdair::PartySize_T&);
00055
00057     static void takeSnapshots (const stdair::Inventory&,
00058                                const stdair::DateTime_T&);
00059   private:
00060
00062     static bool sell (stdair::BookingClass&, const stdair::PartySize_T&);
00063
00065     static bool cancel (stdair::BookingClass&, const stdair::PartySize_T&
    );
00066   };
00067
00068 }
00069 #endif // __AIRINV_BOM_INVENTORYHELPER_HPP
```

23.85 airinv/bom/LegCabinHelper.cpp File Reference

```
#include <cassert>
#include <stdair/bom/LegCabin.hpp>
#include <airinv/bom/LegCabinHelper.hpp>
```

Namespaces

- namespace AIRINV

23.86 LegCabinHelper.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // STDAIR
00007 #include <stdair/bom/LegCabin.hpp>
00008 // AIRINV
00009 #include <airinv/bom/LegCabinHelper.hpp>
00010
00011 namespace AIRINV {
00012
00013 }
```

23.87 airinv/bom/LegCabinHelper.hpp File Reference

Classes

- class AIRINV::LegCabinHelper

Namespaces

- namespace stdair

  Forward declarations.
- namespace AIRINV

23.88 LegCabinHelper.hpp

```
00001 #ifndef __AIRINV_BOM_LEGCABINHELPER_HPP
00002 #define __AIRINV_BOM_LEGCABINHELPER_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
```

```
00007
00008 // Forward declarations
00009 namespace stdair {
00010   class LegCabin;
00011 }
00012
00013 namespace AIRINV {
00016   class LegCabinHelper {
00017
00018   };
00019
00020 }
00021 #endif // __AIRINV_BOM_LEGCABINHELPER_HPP
```

23.89 airinv/bom/LegCabinStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/bom/LegCabin.hpp>
#include <airinv/bom/LegCabinStruct.hpp>
```

Namespaces

- namespace AIRINV

23.90 LegCabinStruct.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/bom/LegCabin.hpp>
00009 // AirInv
00010 #include <airinv/bom/LegCabinStruct.hpp>
00011
00012 namespace AIRINV {
00013
00014   // ////////////////////////////////////////////////////////////////////
00015   const std::string LegCabinStruct::describe() const {
00016     std::ostringstream ostr;
00017     ostr << "         " << _cabinCode << ", " << _saleableCapacity
00018          << ", " << _adjustment  << ", " << _dcsRegrade
00019          << ", " << _au << ", " << _avPool
00020          << ", " << _upr << ", " << _nbOfBookings << ", " <<
      _nav
00021          << ", " << _gav << ", " << _acp << ", " << _etb
00022          << ", " << _staffNbOfBookings << ", " <<
      _wlNbOfBookings
00023          << ", " << _groupNbOfBookings
00024          << std::endl;
00025
00026     for (BucketStructList_T::const_iterator itBucket = _bucketList.
      begin();
00027          itBucket != _bucketList.end(); ++itBucket) {
00028       const BucketStruct& lBucket = *itBucket;
00029       ostr << lBucket.describe();
00030     }
00031     if (_bucketList.empty() == false) {
00032       ostr << std::endl;
00033     }
00034     return ostr.str();
00035   }
00036
00037   // ////////////////////////////////////////////////////////////////////
00038   void LegCabinStruct::fill (stdair::LegCabin& ioLegCabin)
       const {
00039     // Set the Capacity
00040     ioLegCabin.setCapacities (_saleableCapacity);
00041   }
00042
00043 }
```

23.91 airinv/bom/LegCabinStruct.hpp File Reference

```
#include <string>
#include <vector>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <airinv/bom/BucketStruct.hpp>
```

Classes

- struct AIRINV::LegCabinStruct

Namespaces

- namespace stdair

  Forward declarations.
- namespace AIRINV

Typedefs

- typedef std::vector
  < LegCabinStruct > AIRINV::LegCabinStructList_T

23.92 LegCabinStruct.hpp

```
00001 #ifndef __AIRINV_BOM_LEGCABINSTRUCT_HPP
00002 #define __AIRINV_BOM_LEGCABINSTRUCT_HPP
00003
00004 // //////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/basic/StructAbstract.hpp>
00013 // AirInv
00014 #include <airinv/bom/BucketStruct.hpp>
00015
00016 // Forward declarations
00017 namespace stdair {
00018   class LegCabin;
00019 }
00020
00021 namespace AIRINV {
00022
00024   struct LegCabinStruct : public stdair::StructAbstract {
00025     // Attributes
00026     stdair::CabinCode_T _cabinCode;
00027     stdair::CabinCapacity_T _saleableCapacity;
00028     stdair::CapacityAdjustment_T _adjustment;
00029     stdair::CapacityAdjustment_T _dcsRegrade;
00030     stdair::AuthorizationLevel_T _au;
00031     stdair::Availability_T _avPool;
00032     stdair::UPR_T _upr;
00033     stdair::NbOfBookings_T _nbOfBookings;
00034     stdair::Availability_T _nav;
00035     stdair::Availability_T _gav;
00036     stdair::OverbookingRate_T _acp;
00037     stdair::NbOfBookings_T _etb;
00038     stdair::NbOfBookings_T _staffNbOfBookings;
00039     stdair::NbOfBookings_T _wlNbOfBookings;
00040     stdair::NbOfBookings_T _groupNbOfBookings;
00041     BucketStructList_T _bucketList;
00042
00045     void fill (stdair::LegCabin&) const;
00046
00048     const std::string describe() const;
```

```
00049   };
00050
00052   typedef std::vector<LegCabinStruct> LegCabinStructList_T;
00053
00054 }
00055 #endif // __AIRINV_BOM_LEGCABINSTRUCT_HPP
```

23.93 airinv/bom/LegStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/LegDate.hpp>
#include <airinv/bom/LegStruct.hpp>
```

Namespaces

- namespace AIRINV

23.94 LegStruct.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/LegDate.hpp>
00010 // AIRINV
00011 #include <airinv/bom/LegStruct.hpp>
00012
00013 namespace AIRINV {
00014
00015   // ////////////////////////////////////////////////////////////////////
00016   LegStruct::LegStruct ()
00017     : _boardingDate (stdair::DEFAULT_DATE), _offDate (stdair::DEFAULT_DATE) {
00018   }
00019
00020   // ////////////////////////////////////////////////////////////////////
00021   const std::string LegStruct::describe() const {
00022     std::ostringstream ostr;
00023     ostr << "    " << _boardingPoint << " / " << _boardingDate
       << " "
00024          << boost::posix_time::to_simple_string(_boardingTime)
00025          << " -- " << _offPoint << " / " << _offDate << " "
00026          << boost::posix_time::to_simple_string(_offTime)
00027          << " --> "
00028          << boost::posix_time::to_simple_string(_elapsed)
00029          << std::endl;
00030     for (LegCabinStructList_T::const_iterator itCabin = _cabinList.
      begin();
00031          itCabin != _cabinList.end(); itCabin++) {
00032       const LegCabinStruct& lCabin = *itCabin;
00033       ostr << lCabin.describe();
00034     }
00035     ostr << std::endl;
00036
00037     return ostr.str();
00038   }
00039
00040   // ////////////////////////////////////////////////////////////////////
00041   void LegStruct::fill (const stdair::Date_T& iRefDate,
00042                         stdair::LegDate& ioLegDate) const {
00043     // Set the Off Point
00044     ioLegDate.setOffPoint (_offPoint);
00045     // Set the Boarding Date
00046     ioLegDate.setBoardingDate (iRefDate + _boardingDateOffset
      );
00047     // Set the Boarding Time
00048     ioLegDate.setBoardingTime (_boardingTime);
00049     // Set the Off Date
00050     ioLegDate.setOffDate (iRefDate + _offDateOffset);
```

```
00051     // Set the Off Time
00052     ioLegDate.setOffTime (_offTime);
00053     // Set the Elapsed Time
00054     ioLegDate.setElapsedTime (_elapsed);
00055     // Set the operating airline code
00056     ioLegDate.setOperatingAirlineCode (_airlineCode);
00057     // Set the operating flight number
00058     ioLegDate.setOperatingFlightNumber (_flightNumber);
00059   }
00060
00061   // ////////////////////////////////////////////////////////////////////
00062   void LegStruct::fill (stdair::LegDate& ioLegDate) const {
00063     // Set the Off Point
00064     ioLegDate.setOffPoint (_offPoint);
00065     // Set the Boarding Date
00066     ioLegDate.setBoardingDate (_offDate);
00067     // Set the Boarding Time
00068     ioLegDate.setBoardingTime (_boardingTime);
00069     // Set the Off Date
00070     ioLegDate.setOffDate (_offDate);
00071     // Set the Off Time
00072     ioLegDate.setOffTime (_offTime);
00073     // Set the Elapsed Time
00074     ioLegDate.setElapsedTime (_elapsed);
00075     // Set the operating airline code
00076     ioLegDate.setOperatingAirlineCode (_airlineCode);
00077     // Set the operating flight number
00078     ioLegDate.setOperatingFlightNumber (_flightNumber);
00079   }
00080
00081 }
```

23.95 airinv/bom/LegStruct.hpp File Reference

```
#include <string>
#include <vector>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <airinv/bom/LegCabinStruct.hpp>
```

Classes

- struct AIRINV::LegStruct

Namespaces

- namespace stdair

  Forward declarations.
- namespace AIRINV

Typedefs

- typedef std::vector< LegStruct > AIRINV::LegStructList_T

23.96 LegStruct.hpp

```
00001 #ifndef __AIRINV_BOM_LEGSTRUCT_HPP
00002 #define __AIRINV_BOM_LEGSTRUCT_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010 // STDAIR
00011 #include <stdair/stdair_inventory_types.hpp>
```

```
00012 #include <stdair/basic/StructAbstract.hpp>
00013 // AIRINV
00014 #include <airinv/bom/LegCabinStruct.hpp>
00015
00016 // Forward declarations
00017 namespace stdair {
00018   class LegDate;
00019 }
00020
00021 namespace AIRINV {
00022
00024   struct LegStruct : public stdair::StructAbstract {
00025     // Attributes
00026     stdair::AirlineCode_T _airlineCode;
00027     stdair::FlightNumber_T _flightNumber;
00028     stdair::AirportCode_T _boardingPoint;
00029     stdair::DateOffset_T _boardingDateOffset;
00030     stdair::Date_T _boardingDate;
00031     stdair::Duration_T _boardingTime;
00032     stdair::AirportCode_T _offPoint;
00033     stdair::DateOffset_T _offDateOffset;
00034     stdair::Date_T _offDate;
00035     stdair::Duration_T _offTime;
00036     stdair::Duration_T _elapsed;
00037     LegCabinStructList_T _cabinList;
00038
00044     void fill (const stdair::Date_T& iRefDate, stdair::LegDate&) const;
00045
00047     void fill (stdair::LegDate&) const;
00048
00050     const std::string describe() const;
00051
00053     LegStruct();
00054   };
00055
00057   typedef std::vector<LegStruct> LegStructList_T;
00058
00059 }
00060 #endif // __AIRINV_BOM_LEGSTRUCT_HPP
```

23.97 airinv/bom/SegmentCabinHelper.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <limits>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/float_utils.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/SimpleNestingStructure.hpp>
#include <stdair/bom/NestingNode.hpp>
#include <stdair/bom/Policy.hpp>
#include <stdair/factory/FacBomManager.hpp>
#include <airinv/bom/SegmentCabinHelper.hpp>
#include <airinv/bom/FlightDateHelper.hpp>
```

Namespaces

- namespace AIRINV

23.98 SegmentCabinHelper.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
```

```
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 #include <limits>
00008 // StdAir
00009 #include <stdair/basic/BasConst_Inventory.hpp>
00010 #include <stdair/basic/float_utils.hpp>
00011 #include <stdair/bom/BomManager.hpp>
00012 #include <stdair/bom/FlightDate.hpp>
00013 #include <stdair/bom/LegCabin.hpp>
00014 #include <stdair/bom/SegmentCabin.hpp>
00015 #include <stdair/bom/FareFamily.hpp>
00016 #include <stdair/bom/BookingClass.hpp>
00017 #include <stdair/bom/SimpleNestingStructure.hpp>
00018 #include <stdair/bom/NestingNode.hpp>
00019 #include <stdair/bom/Policy.hpp>
00020 #include <stdair/factory/FacBomManager.hpp>
00021 // AirInv
00022 #include <airinv/bom/SegmentCabinHelper.hpp>
00023 #include <airinv/bom/FlightDateHelper.hpp>
00024
00025 namespace AIRINV {
00026
00027   // ////////////////////////////////////////////////////////////////////
00028   void SegmentCabinHelper::initialiseAU (
      stdair::SegmentCabin& iSegmentCabin) {
00029
00030     // Initialise the capacity and availability pool.
00031     const stdair::LegCabinList_T& lLCList =
00032       stdair::BomManager::getList<stdair::LegCabin> (iSegmentCabin);
00033
00034     stdair::CabinCapacity_T lCapacity =
00035       std::numeric_limits<stdair::CabinCapacity_T>::max();
00036     for (stdair::LegCabinList_T::const_iterator itLC = lLCList.begin();
00037          itLC != lLCList.end(); ++itLC) {
00038
00039       const stdair::LegCabin* lLC_ptr = *itLC;
00040       assert (lLC_ptr != NULL);
00041
00042       const stdair::CabinCapacity_T& lCabinCap = lLC_ptr->getOfferedCapacity();
00043       if (lCapacity > lCabinCap) {
00044         lCapacity = lCabinCap;
00045       }
00046     }
00047     iSegmentCabin.setCapacity (lCapacity);
00048     iSegmentCabin.setAvailabilityPool (lCapacity);
00049
00050     // Browse the list of booking classes and set the AU of each booking
00051     // class to the availability pool of the cabin.
00052     const stdair::BookingClassList_T& lBCList =
00053       stdair::BomManager::getList<stdair::BookingClass> (iSegmentCabin);
00054     for (stdair::BookingClassList_T::const_iterator itBC = lBCList.begin();
00055          itBC != lBCList.end(); ++itBC) {
00056       stdair::BookingClass* lBC_ptr = *itBC;
00057       assert (lBC_ptr != NULL);
00058       lBC_ptr->setAuthorizationLevel (lCapacity);
00059     }
00060   }
00061
00062   // ////////////////////////////////////////////////////////////////////
00063   void SegmentCabinHelper::
00064   updateFromReservation (const stdair::FlightDate&
      iFlightDate,
00065                          stdair::SegmentCabin& ioSegmentCabin,
00066                          const stdair::PartySize_T& iNbOfBookings){
00067     // Update the commited space of the segment-cabin.
00068     ioSegmentCabin.updateFromReservation (iNbOfBookings);
00069
00070     // Update the availability of the flight-date.
00071     FlightDateHelper::updateAvailability (
      iFlightDate, ioSegmentCabin,
00072                                           iNbOfBookings);
00073   }
00074
00075   // ////////////////////////////////////////////////////////////////////
00076   void SegmentCabinHelper::
00077   buildPseudoBidPriceVector (stdair::SegmentCabin&
      ioSegmentCabin) {
00078     // Retrieve the segment-cabin capacity.
00079     const stdair::Availability_T& lAvlPool =
00080       ioSegmentCabin.getAvailabilityPool();
00081     unsigned int lAvlPoolInt;
00082     if (lAvlPool < 0) {
00083       lAvlPoolInt = 0;
00084     } else {
00085       assert (lAvlPool >= 0);
```

```
00086       lAvlPoolInt = static_cast<unsigned int> (lAvlPool);
00087     }
00088     stdair::BidPriceVector_T lPseudoBidPriceVector (lAvlPoolInt, 0.0);
00089
00090     // Browse the leg-cabin list.
00091     const stdair::LegCabinList_T& lLCList =
00092       stdair::BomManager::getList<stdair::LegCabin> (ioSegmentCabin);
00093     for (stdair::LegCabinList_T::const_iterator itLC = lLCList.begin();
00094          itLC != lLCList.end(); ++itLC) {
00095       const stdair::LegCabin* lLC_ptr = *itLC;
00096       assert (lLC_ptr != NULL);
00097
00098       const stdair::BidPriceVector_T& lBPV = lLC_ptr->getBidPriceVector();
00099       stdair::BidPriceVector_T::const_reverse_iterator itBP = lBPV.rbegin();
00100       for (stdair::BidPriceVector_T::reverse_iterator itPBP =
00101              lPseudoBidPriceVector.rbegin();
00102            itPBP != lPseudoBidPriceVector.rend(); ++itPBP, ++itBP) {
00103         assert (itBP != lBPV.rend());
00104         stdair::BidPrice_T& lCurrentPBP = *itPBP;
00105         const stdair::BidPrice_T& lCurrentBP = *itBP;
00106         lCurrentPBP += lCurrentBP;
00107       }
00108     }
00109
00110     ioSegmentCabin.setBidPriceVector (lPseudoBidPriceVector);
00111
00112     // // DEBUG
00113     // std::ostringstream oStr;
00114     // oStr << "Pseudo BPV: ";
00115     // for (stdair::BidPriceVector_T::const_iterator itBP =
00116     //       lPseudoBidPriceVector.begin(); itBP !=
 lPseudoBidPriceVector.end();
00117     //      ++itBP) {
00118     //   const stdair::BidPrice_T& lCurrentBP = *itBP;
00119     //   oStr << lCurrentBP << " ";
00120     // }
00121     // oStr << std::endl;
00122     // // STDAIR_LOG_DEBUG (oStr.str());
00123     // std::cout << oStr.str() << std::endl;
00124   }
00125
00126   // ////////////////////////////////////////////////////////////////////
00127   void SegmentCabinHelper::
00128   updateBookingControlsUsingPseudoBidPriceVector
       (const stdair::SegmentCabin& iSegmentCabin) {
00129     // Retrieve the pseudo bid price vector.
00130     const stdair::BidPriceVector_T& lPseudoBPV =
00131       iSegmentCabin.getBidPriceVector();
00132     const stdair::Availability_T& lAvlPool= iSegmentCabin.getAvailabilityPool()
     ;
00133
00134     // Update the cumulative booking limit for all booking classes.
00135     // Browse the nesting structure
00136     const stdair::SimpleNestingStructure& lYieldBasedNestingStructure =
00137       stdair::BomManager::getObject<stdair::SimpleNestingStructure>(
     iSegmentCabin, stdair::YIELD_BASED_NESTING_STRUCTURE_CODE);
00138     const stdair::NestingNodeList_T& lNestingNodeList =
00139       stdair::BomManager::getList<stdair::NestingNode>(
     lYieldBasedNestingStructure);
00140     for (stdair::NestingNodeList_T::const_iterator itNS =
00141            lNestingNodeList.begin();
00142          itNS != lNestingNodeList.end(); ++itNS) {
00143       stdair::NestingNode* lNestingNode_ptr = *itNS;
00144       assert (lNestingNode_ptr != NULL);
00145       const stdair::Yield_T lNodeYield =
00146         lNestingNode_ptr->getYield();
00147       if (lNodeYield < 0) {
00148         continue;
00149       }
00150       const stdair::BookingClassList_T& lBCList =
00151         stdair::BomManager::getList<stdair::BookingClass> (*lNestingNode_ptr);
00152       stdair::BookingClassList_T::const_iterator itBC = lBCList.begin();
00153       assert(itBC != lBCList.end());
00154       // Browse the booking class list of the current node
00155       const stdair::Yield_T& lYield = lNestingNode_ptr->getYield();
00156       const FloatingPoint<double> lYieldFlotingPoint (lYield);
00157       stdair::BookingLimit_T lCumuBL = lAvlPool;
00158       for (stdair::BidPriceVector_T::const_reverse_iterator itBP =
00159            lPseudoBPV.rbegin(); itBP != lPseudoBPV.rend(); ++itBP) {
00160         const stdair::BidPrice_T& lBP = *itBP;
00161         const FloatingPoint<double> lBPFlotingPoint (lBP);
00162         const bool isAlmostEqual =
00163           lYieldFlotingPoint.AlmostEquals(lBPFlotingPoint);
00164
00165         if ((lYield < lBP) && (isAlmostEqual == false)) {
00166           lCumuBL = itBP - lPseudoBPV.rbegin();
00167           break;
```

```
00168         }
00169       }
00170       for (; itBC != lBCList.end(); ++itBC) {
00171         stdair::BookingClass* lBC_ptr = *itBC;
00172         assert (lBC_ptr != NULL);
00173         lBC_ptr->setCumulatedBookingLimit (lCumuBL);
00174         // DEBUG
00175         // STDAIR_LOG_DEBUG("Updating the BL for class: "
00176         //                  << lBC_ptr->describeKey()
00177         //                  << ", with yield " << lNodeYield
00178         //                  << " and BL: " << lCumuBL);
00179       }
00180     }
00181     // Update the authorization levels from the booking limits
00182     updateAUs (iSegmentCabin);
00183   }
00184
00185   // ////////////////////////////////////////////////////////////////////
00186   void SegmentCabinHelper::updateAUs(const
      stdair::SegmentCabin& iSegmentCabin){
00187     // Browse the nesting structure and compute the AU from the
00188     // cumulative booking counter and the cumulative booking limit.
00189     stdair::NbOfBookings_T lCumulativeBookingCounter = 0.0;
00190     // Browse the nesting structure
00191     const stdair::SimpleNestingStructure& lYieldBasedNestingStructure =
00192       stdair::BomManager::getObject<stdair::SimpleNestingStructure>(
      iSegmentCabin, stdair::YIELD_BASED_NESTING_STRUCTURE_CODE);
00193     const stdair::NestingNodeList_T& lNestingNodeList =
00194       stdair::BomManager::getList<stdair::NestingNode>(
      lYieldBasedNestingStructure);
00195     for (stdair::NestingNodeList_T::const_reverse_iterator itNS =
00196            lNestingNodeList.rbegin();
00197          itNS != lNestingNodeList.rend(); ++itNS) {
00198       stdair::NestingNode* lNestingNode_ptr = *itNS;
00199       assert (lNestingNode_ptr != NULL);
00200       const stdair::Yield_T lNodeYield =
00201         lNestingNode_ptr->getYield();
00202       if (lNodeYield < 0) {
00203         continue;
00204       }
00205       const stdair::BookingClassList_T& lBCList =
00206         stdair::BomManager::getList<stdair::BookingClass> (*lNestingNode_ptr);
00207       stdair::BookingClassList_T::const_iterator itBC = lBCList.begin();
00208       assert(itBC != lBCList.end());
00209       const stdair::BookingLimit_T& lCumuBookingLimit =
00210         (*itBC)->getCumulatedBookingLimit();
00211       // Browse the booking class list of the current node to update the
00212       // cumulative booking counter
00213       for (; itBC != lBCList.end(); ++itBC) {
00214         stdair::BookingClass* lBC_ptr = *itBC;
00215         assert (lBC_ptr != NULL);
00216         assert(lCumuBookingLimit == lBC_ptr->getCumulatedBookingLimit());
00217         const stdair::NbOfBookings_T& lNbOfBookings = lBC_ptr->getNbOfBookings(
      );
00218         lCumulativeBookingCounter += lNbOfBookings;
00219       }
00220       stdair::AuthorizationLevel_T lAU =
00221           lCumulativeBookingCounter + lCumuBookingLimit;
00222       // Browse the booking class list of the current node to set
00223       // the authorization level of all booking classes of the node
00224       for (itBC = lBCList.begin(); itBC != lBCList.end(); ++itBC) {
00225         stdair::BookingClass* lBC_ptr = *itBC;
00226         assert (lBC_ptr != NULL);
00227         lBC_ptr->setAuthorizationLevel (lAU);
00228         // DEBUG
00229         // STDAIR_LOG_DEBUG ("Updating the AU for class: "
00230         //                   << lBC_ptr->describeKey()
00231         //                   << ", with BL: " << lCumuBookingLimit
00232         //                   << ", CumuBkg: " << lCumulativeBookingCounter
00233         //                   << ", AU: " << lAU);
00234       }
00235     }
00236   }
00237
00238   // ////////////////////////////////////////////////////////////////////
00239   void SegmentCabinHelper::
00240   updateAvailabilities (const stdair::SegmentCabin&
      iSegmentCabin) {
00241     // Browse the nesting structure and compute the avl from the
00242     // cumulative booking counter and the AU.
00243     stdair::NbOfBookings_T lCumulativeBookingCounter = 0.0;
00244     const stdair::SimpleNestingStructure& lYieldBasedNestingStructure =
00245       stdair::BomManager::getObject<stdair::SimpleNestingStructure>(
      iSegmentCabin, stdair::YIELD_BASED_NESTING_STRUCTURE_CODE);
00246     const stdair::NestingNodeList_T& lNestingNodeList =
00247       stdair::BomManager::getList<stdair::NestingNode>(
      lYieldBasedNestingStructure);
```

```
00248     for (stdair::NestingNodeList_T::const_reverse_iterator itNS =
00249            lNestingNodeList.rbegin();
00250          itNS != lNestingNodeList.rend(); ++itNS) {
00251       stdair::NestingNode* lNestingNode_ptr = *itNS;
00252       assert (lNestingNode_ptr != NULL);
00253       const stdair::Yield_T& lNodeYield = lNestingNode_ptr->getYield();
00254       if (lNodeYield < 0) {
00255         continue;
00256       }
00257       const stdair::BookingClassList_T& lBCList =
00258         stdair::BomManager::getList<stdair::BookingClass> (*lNestingNode_ptr);
00259       stdair::BookingClassList_T::const_iterator itBC = lBCList.begin();
00260       assert(itBC != lBCList.end());
00261       stdair::BookingClass* lFirstBC_ptr = *itBC;
00262       assert (lFirstBC_ptr != NULL);
00263       const stdair::AuthorizationLevel_T& lNodeAU =
00264         lFirstBC_ptr->getAuthorizationLevel();
00265       // Browse the booking class list of the current node to update the
00266       // cumulative booking counter
00267       for (; itBC != lBCList.end(); ++itBC) {
00268         stdair::BookingClass* lBC_ptr = *itBC;
00269         assert (lBC_ptr != NULL);
00270         assert(lNodeAU == lBC_ptr->getAuthorizationLevel());
00271         const stdair::NbOfBookings_T& lNbOfBookings = lBC_ptr->getNbOfBookings(
      );
00272         lCumulativeBookingCounter += lNbOfBookings;
00273       }
00274       const stdair::Availability_T lNodeAvl = lNodeAU -
      lCumulativeBookingCounter;
00275       // Browse the booking class list of the current node to set
00276       // the availability of all booking classes of the node
00277       for (itBC = lBCList.begin(); itBC != lBCList.end(); ++itBC) {
00278         stdair::BookingClass* lBC_ptr = *itBC;
00279         assert (lBC_ptr != NULL);
00280         lBC_ptr->setSegmentAvailability (lNodeAvl);
00281       }
00282     }
00283
00284     // Cascading
00285     stdair::NestingNodeList_T::const_iterator itCurrentNode =
00286       lNestingNodeList.begin();
00287     assert (itCurrentNode != lNestingNodeList.end());
00288     stdair::NestingNodeList_T::const_iterator itNextNode = itCurrentNode;
00289     ++itNextNode;
00290     for (; itNextNode != lNestingNodeList.end(); ++itCurrentNode, ++itNextNode)
       {
00291       assert(itCurrentNode != lNestingNodeList.end());
00292       stdair::NestingNode* lCurrentNode_ptr = *itCurrentNode;
00293       assert (lCurrentNode_ptr != NULL);
00294       const stdair::Yield_T& lCurrentNodeYield = lCurrentNode_ptr->getYield();
00295       if (lCurrentNodeYield < 0) {
00296         break;
00297       }
00298       const stdair::BookingClassList_T& lCurrentBCList =
00299         stdair::BomManager::getList<stdair::BookingClass> (*lCurrentNode_ptr);
00300       stdair::BookingClassList_T::const_iterator itCurrentBC =
00301         lCurrentBCList.begin();
00302       stdair::BookingClass* lCurrentBC_ptr = *(itCurrentBC);
00303       assert (lCurrentBC_ptr != NULL);
00304       assert(itNextNode != lNestingNodeList.end());
00305       stdair::NestingNode* lNextNode_ptr = *itNextNode;
00306       assert (lNextNode_ptr != NULL);
00307       const stdair::Yield_T& lNextNodeYield = lNextNode_ptr->getYield();
00308       if (lNextNodeYield < 0) {
00309         break;
00310       }
00311       const stdair::BookingClassList_T& lNextBCList =
00312         stdair::BomManager::getList<stdair::BookingClass> (*lNextNode_ptr);
00313       stdair::BookingClassList_T::const_iterator itNextBC =
00314         lNextBCList.begin();
00315       stdair::BookingClass* lNextBC_ptr = *(itNextBC);
00316       assert (lNextBC_ptr != NULL);
00317       const stdair::Availability_T& lCurrentAvl =
00318         lCurrentBC_ptr->getSegmentAvailability();
00319       const stdair::Availability_T& lNextAvl =
00320         lNextBC_ptr->getSegmentAvailability();
00321       if (lCurrentAvl < lNextAvl) {
00322         for (; itNextBC != lNextBCList.end(); ++itNextBC) {
00323           lNextBC_ptr = *itNextBC;
00324           assert (lNextBC_ptr != NULL);
00325           lNextBC_ptr->setSegmentAvailability (lCurrentAvl);
00326         }
00327       }
00328     }
00329   }
00330
00331   // ////////////////////////////////////////////////////////////////////
```

```
00332   void SegmentCabinHelper::
00333   initYieldBasedNestingStructure (
      stdair::SegmentCabin& ioSegmentCabin) {
00334     // Create the nesting structure.
00335     stdair::NestingStructureKey lKey (
      stdair::YIELD_BASED_NESTING_STRUCTURE_CODE);
00336     stdair::SimpleNestingStructure& lNestingStructure =
00337       stdair::FacBom<stdair::SimpleNestingStructure>::instance().create(lKey);
00338     stdair::FacBomManager::addToListAndMap (ioSegmentCabin, lNestingStructure);
00339     stdair::FacBomManager::linkWithParent (ioSegmentCabin, lNestingStructure);
00340
00341     // Build a multimap of booking classes with their yields as keys.
00342     std::multimap<const stdair::Yield_T, stdair::BookingClass*> lClassMap;
00343     const stdair::BookingClassList_T& lBCList =
00344       stdair::BomManager::getList<stdair::BookingClass> (ioSegmentCabin);
00345     for (stdair::BookingClassList_T::const_iterator itBC = lBCList.begin();
00346          itBC != lBCList.end(); ++itBC) {
00347       stdair::BookingClass* lBC_ptr = *itBC;
00348       assert (lBC_ptr != NULL);
00349       const stdair::Yield_T& lYield = lBC_ptr->getYield();
00350       lClassMap.insert(std::multimap<const stdair::Yield_T,
       stdair::BookingClass*>::value_type(lYield, lBC_ptr));
00351     }
00352
00353     stdair::Yield_T lLastYield = -1.0;
00354     stdair::NestingNode* lCurrentNode_ptr = NULL;
00355     for (std::multimap<const stdair::Yield_T,
      stdair::BookingClass*>::reverse_iterator itBC = lClassMap.rbegin();
00356          itBC != lClassMap.rend(); ++itBC) {
00357       const stdair::Yield_T& lCurrentYield = itBC->first;
00358       stdair::BookingClass* lBC_ptr = itBC->second;
00359
00360       // Compare the current yield and the last one.
00361       // TODO: use float utils
00362       //if (lCurrentYield.AlmostEquals (lLastYield) == false) {
00363       if (lCurrentYield != lLastYield) {
00364         // Create a nesting node
00365         stdair::NestingNodeCode_T lNodeCode (lBC_ptr->describeKey());
00366         stdair::NestingNodeKey lNodeKey (lNodeCode);
00367         stdair::NestingNode& lNestingNode =
00368           stdair::FacBom<stdair::NestingNode>::instance().create (lNodeKey);
00369         stdair::FacBomManager::addToList (lNestingStructure, lNestingNode);
00370         stdair::FacBomManager::linkWithParent (lNestingStructure, lNestingNode)
     ;
00371         lCurrentNode_ptr = &lNestingNode;
00372         lCurrentNode_ptr->setYield(lCurrentYield);
00373         // Add the booking class to the node.
00374         stdair::FacBomManager::addToList (lNestingNode, *lBC_ptr);
00375         lLastYield = lCurrentYield;
00376       } else {
00377         // Add the booking class to the current node.
00378         stdair::FacBomManager::addToList (*lCurrentNode_ptr, *lBC_ptr);
00379       }
00380     }
00381   }
00382
00383   // ////////////////////////////////////////////////////////////////////
00384   void SegmentCabinHelper::
00385   initListOfUsablePolicies (stdair::SegmentCabin&
      ioSegmentCabin) {
00386     const stdair::FareFamilyList_T& lFareFamilyList =
00387       stdair::BomManager::getList<stdair::FareFamily> (ioSegmentCabin);
00388     stdair::FareFamilyList_T::const_iterator itFF = lFareFamilyList.begin();
00389
00390     unsigned int lPolicyCounter = 0;
00391     std::ostringstream oStr;
00392     oStr << lPolicyCounter;
00393     stdair::PolicyKey lKey (oStr.str());
00394     stdair::Policy& lPolicy =
00395       stdair::FacBom<stdair::Policy>::instance().create(lKey);
00396     stdair::FacBomManager::addToList (ioSegmentCabin, lPolicy);
00397     stdair::FacBomManager::linkWithParent (ioSegmentCabin, lPolicy);
00398     createPolicies (ioSegmentCabin, lFareFamilyList, itFF, lPolicy,
00399                     lPolicyCounter,
00400                     std::numeric_limits<stdair::Yield_T>::max());
00401   }
00402
00403   // ////////////////////////////////////////////////////////////////////
00404   void SegmentCabinHelper::
00405   createPolicies (stdair::SegmentCabin& ioSegmentCabin,
00406                   const stdair::FareFamilyList_T& iFareFamilyList,
00407                   const stdair::FareFamilyList_T::const_iterator& itFF,
00408                   stdair::Policy& ioCurrentPolicy,
00409                   unsigned int& ioPolicyCounter,
00410                   const stdair::Yield_T& iPreviousYield) {
00411     if (itFF != iFareFamilyList.end()) {
00412       // We add a booking class of the next Fare Family if it is cheapest than
```

```
00413       // the previous booking class in the policy.
00414       // Assumption: the fare family list is sorted according to their fares:
00415       // Fare_1 > Fare_2 > ... > Fare_n
00416       const stdair::FareFamily* lFF_ptr = *itFF;
00417       //Retrieve the booking class list of the current fare family
00418       const stdair::BookingClassList_T& lBookingClassList =
00419         stdair::BomManager::getList<stdair::BookingClass> (*lFF_ptr);
00420       stdair::BookingClassList_T::const_iterator itBC =
00421         lBookingClassList.begin();
00422       stdair::FareFamilyList_T::const_iterator lItFF = itFF;
00423       lItFF++;
00424
00425       // Browse the booking class list
00426       for (; itBC != lBookingClassList.end(); ++itBC) {
00427         stdair::BookingClass* lBC_ptr = *itBC;
00428         assert(lBC_ptr != NULL);
00429         const stdair::Yield_T& lCurrentYield = lBC_ptr->getYield();
00430         if (lCurrentYield >= iPreviousYield) {
00431           continue;
00432         }
00433         assert(lCurrentYield < iPreviousYield);
00434         // Add the current booking class to the list, update the current policy
00435         // and call the same method for the next fare family
00436         ++ioPolicyCounter;
00437         std::ostringstream oStr;
00438         oStr << ioPolicyCounter;
00439         stdair::PolicyKey lKey (oStr.str());
00440         stdair::Policy& lNewPolicy =
00441           stdair::FacBom<stdair::Policy>::instance().create(lKey);
00442         stdair::FacBomManager::addToList (ioSegmentCabin, lNewPolicy);
00443         stdair::FacBomManager::linkWithParent (ioSegmentCabin, lNewPolicy);
00444
00445         // Copy the list of booking classes of the current policy to the new
     one
00446         bool hasAListOfBC =
00447           stdair::BomManager::hasList<stdair::BookingClass> (ioCurrentPolicy);
00448         if (hasAListOfBC == true) {
00449           const stdair::BookingClassList_T& lToBeCopiedBCList =
00450             stdair::BomManager::getList<stdair::BookingClass> (ioCurrentPolicy)
    ;
00451           for (stdair::BookingClassList_T::const_iterator itBCToBeCopied =
00452                  lToBeCopiedBCList.begin();
00453                itBCToBeCopied != lToBeCopiedBCList.end(); ++itBCToBeCopied) {
00454             stdair::BookingClass* lBCToBeCopied_ptr = *itBCToBeCopied;
00455             assert (lBCToBeCopied_ptr != NULL);
00456             stdair::FacBomManager::addToList (lNewPolicy, *lBCToBeCopied_ptr);
00457           }
00458         }
00459         stdair::FacBomManager::addToList(lNewPolicy, *lBC_ptr);
00460
00461         createPolicies (ioSegmentCabin, iFareFamilyList, lItFF, lNewPolicy,
00462                         ioPolicyCounter, lCurrentYield);
00463       }
00464     }
00465
00466   }
00467
00468 }
```

23.99 airinv/bom/SegmentCabinHelper.hpp File Reference

```
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/FareFamilyTypes.hpp>
```

Classes

- class AIRINV::SegmentCabinHelper

  Class representing the actual business functions for an airline segment-cabin.

Namespaces

- namespace stdair

  Forward declarations.
- namespace AIRINV

23.100 SegmentCabinHelper.hpp

```
00001 #ifndef __AIRINV_BOM_SEGMENTCABINHELPER_HPP
00002 #define __AIRINV_BOM_SEGMENTCABINHELPER_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009 #include <stdair/bom/FareFamilyTypes.hpp>
00010
00011 // Forward declarations
00012 namespace stdair {
00013   class FlightDate;
00014   class SegmentCabin;
00015   class FareFamily;
00016   class Policy;
00017 }
00018
00019 namespace AIRINV {
00020
00025   class SegmentCabinHelper {
00026   public:
00027     // ////////// Business Methods /////////
00031     static void updateFromReservation (const
      stdair::FlightDate&,
00032                                        stdair::SegmentCabin&,
00033                                        const stdair::PartySize_T&);
00034
00038     static void buildPseudoBidPriceVector (
      stdair::SegmentCabin&);
00039
00043     static void updateBookingControlsUsingPseudoBidPriceVector
       (const stdair::SegmentCabin&);
00044
00047     static void updateAUs (const stdair::SegmentCabin&);
00048
00051     static void updateAvailabilities (const
      stdair::SegmentCabin&);
00052
00056     static void initialiseAU (stdair::SegmentCabin&);
00057
00061     static void initYieldBasedNestingStructure (
      stdair::SegmentCabin&);
00062
00066     static void initListOfUsablePolicies (
      stdair::SegmentCabin&);
00067
00068   private:
00072     static void createPolicies (stdair::SegmentCabin&,
00073                                 const stdair::FareFamilyList_T&,
00074                                 const stdair::FareFamilyList_T::const_iterator&
      ,
00075                                 stdair::Policy&, unsigned int&,
00076                                 const stdair::Yield_T&);
00077   };
00078
00079 }
00080 #endif // __AIRINV_BOM_SEGMENTCABINHELPER_HPP
```

23.101 airinv/bom/SegmentCabinStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/bom/SegmentCabin.hpp>
#include <airinv/bom/SegmentCabinStruct.hpp>
```

Namespaces

- namespace AIRINV

23.102 SegmentCabinStruct.cpp

```
00001 // //////////////////////////////////////////////////////////////////
```

```
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/bom/SegmentCabin.hpp>
00009 // AirInv
00010 #include <airinv/bom/SegmentCabinStruct.hpp>
00011
00012 namespace AIRINV {
00013
00014   // ////////////////////////////////////////////////////////////////////
00015   const std::string SegmentCabinStruct::describe()
       const {
00016     std::ostringstream ostr;
00017
00018     ostr << "        " << _cabinCode << ", " << _nbOfBookings
       << std::endl;
00019
00020     for (FareFamilyStructList_T::const_iterator itFF = _fareFamilies
      .begin();
00021          itFF != _fareFamilies.end(); ++itFF) {
00022       const FareFamilyStruct& lFF = *itFF;
00023       ostr << lFF.describe();
00024     }
00025     if (_fareFamilies.empty() == false) {
00026       ostr << std::endl;
00027     }
00028
00029     return ostr.str();
00030   }
00031
00032   // ////////////////////////////////////////////////////////////////////
00033   void SegmentCabinStruct::fill (stdair::SegmentCabin&
      ioSegmentCabin) const {
00034     // Set the total number of bookings
00035     // ioSegmentCabin.setNbOfBookings (_nbOfBookings);
00036   }
00037
00038 }
```

23.103 airinv/bom/SegmentCabinStruct.hpp File Reference

```
#include <string>
#include <vector>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <airinv/bom/FareFamilyStruct.hpp>
```

Classes

- struct AIRINV::SegmentCabinStruct

  Utility Structure for the parsing of SegmentCabin details.

Namespaces

- namespace stdair

  Forward declarations.
- namespace AIRINV

Typedefs

- typedef std::vector
  < SegmentCabinStruct > AIRINV::SegmentCabinStructList_T

23.104 SegmentCabinStruct.hpp

```
00001 #ifndef __AIRINV_BOM_SEGMENTCABINSTRUCT_HPP
00002 #define __AIRINV_BOM_SEGMENTCABINSTRUCT_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/basic/StructAbstract.hpp>
00013 // AirInv
00014 #include <airinv/bom/FareFamilyStruct.hpp>
00015
00016 // Forward declarations
00017 namespace stdair {
00018   class SegmentCabin;
00019 }
00020
00021 namespace AIRINV {
00022
00026   struct SegmentCabinStruct : public stdair::StructAbstract {
00027     // Attributes
00028     stdair::CabinCode_T _cabinCode;
00029     stdair::NbOfBookings_T _nbOfBookings;
00030     FareFamilyStruct _itFareFamily;
00031     FareFamilyStructList_T _fareFamilies;
00032
00037     void fill (stdair::SegmentCabin&) const;
00038
00042     const std::string describe() const;
00043   };
00044
00048   typedef std::vector<SegmentCabinStruct> SegmentCabinStructList_T
    ;
00049
00050 }
00051 #endif // __AIRINV_BOM_SEGMENTCABINSTRUCT_HPP
```

23.105 airinv/bom/SegmentDateHelper.cpp File Reference

```
#include <cassert>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/LegDate.hpp>
#include <airinv/bom/SegmentDateHelper.hpp>
#include <airinv/bom/SegmentCabinHelper.hpp>
```

Namespaces

- namespace AIRINV

23.106 SegmentDateHelper.cpp

```
00001 // //////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // STDAIR
00007 #include <stdair/basic/BasConst_General.hpp>
00008 #include <stdair/bom/BomManager.hpp>
00009 #include <stdair/bom/SegmentDate.hpp>
00010 #include <stdair/bom/SegmentCabin.hpp>
00011 #include <stdair/bom/LegDate.hpp>
00012 // AIRINV
00013 #include <airinv/bom/SegmentDateHelper.hpp>
```

```
00014 #include <airinv/bom/SegmentCabinHelper.hpp>
00015
00016 namespace AIRINV {
00017   // ////////////////////////////////////////////////////////////////////
00018   void SegmentDateHelper::fillFromRouting (
      stdair::SegmentDate& ioSegmentDate) {
00019     /*
00020      * If the segment is just marketed by this carrier,
00021      * retrieve the operating segment and call the fillFromRouting
00022      * method on it.
00023      */
00024     stdair::SegmentDate* lOperatingSegmentDate_ptr =
00025       ioSegmentDate.getOperatingSegmentDate ();
00026     if (lOperatingSegmentDate_ptr != NULL) {
00027       return;
00028     }
00029     // Retrieve the first and the last legs of the routing.
00030     // Note that in the majority of the cases, as flights are mono-legs,
00031     // the first and last legs are thus the same.
00032     const stdair::LegDateList_T& lLegDateList =
00033       stdair::BomManager::getList<stdair::LegDate> (ioSegmentDate);
00034     stdair::LegDateList_T::const_iterator itFirstLeg = lLegDateList.begin();
00035     const stdair::LegDate* lFirstLeg_ptr = *itFirstLeg;
00036     assert (lFirstLeg_ptr != NULL);
00037     stdair::LegDateList_T::const_reverse_iterator itLastLeg =
00038       lLegDateList.rbegin();
00039     const stdair::LegDate* lLastLeg_ptr = *itLastLeg;
00040     assert (lLastLeg_ptr != NULL);
00041
00042     // Set the Boarding Date
00043     const stdair::Date_T& lBoardingDate = lFirstLeg_ptr->getBoardingDate();
00044     ioSegmentDate.setBoardingDate (lBoardingDate);
00045     // Set the Boarding Time
00046     const stdair::Duration_T& lBoardingTime = lFirstLeg_ptr->getBoardingTime();
00047     ioSegmentDate.setBoardingTime (lBoardingTime);
00048     // Set the Off Date
00049     const stdair::Date_T& lOffDate = lLastLeg_ptr->getOffDate();
00050     ioSegmentDate.setOffDate (lOffDate);
00051     // Set the Off Time
00052     const stdair::Duration_T& lOffTime = lLastLeg_ptr->getOffTime();
00053     ioSegmentDate.setOffTime (lOffTime);
00054     // Set the Elapsed Time for the whole path
00055     updateElapsedTimeFromRouting (ioSegmentDate);
00056
00057     // Initialise the AU for all classes.
00058     const stdair::SegmentCabinList_T& lSegmentCabinList =
00059       stdair::BomManager::getList<stdair::SegmentCabin> (ioSegmentDate);
00060     for (stdair::SegmentCabinList_T::const_iterator itSC =
00061            lSegmentCabinList.begin(); itSC != lSegmentCabinList.end(); ++itSC)
      {
00062       stdair::SegmentCabin* lSC_ptr = *itSC;
00063       assert (lSC_ptr != NULL);
00064
00065       // Initialise the AU for children booking classes.
00066       SegmentCabinHelper::initialiseAU (*
      lSC_ptr);
00067     }
00068   }
00069
00070   // ////////////////////////////////////////////////////////////////////
00071   void SegmentDateHelper::
00072   updateElapsedTimeFromRouting (
      stdair::SegmentDate& ioSegmentDate) {
00073
00074     const stdair::LegDateList_T& lLegDateList =
00075       stdair::BomManager::getList<stdair::LegDate> (ioSegmentDate);
00076
00077     stdair::LegDateList_T::const_iterator itLegDate = lLegDateList.begin();
00078     const stdair::LegDate* lCurrentLegDate_ptr = *itLegDate;
00079     assert (lCurrentLegDate_ptr != NULL);
00080
00081     // Retrieve the elapsed time of the first leg
00082     stdair::Duration_T lElapsedTime = lCurrentLegDate_ptr->getElapsedTime();
00083
00084     // Go to the next leg, if existing. If not existing, the following
00085     // loop will not be entered (as it means: currentLeg ==
       _legDateList.end()).
00086     ++itLegDate;
00087
00088     for (const stdair::LegDate* lPreviousLegDate_ptr = lCurrentLegDate_ptr;
00089          itLegDate != lLegDateList.end();
00090          ++itLegDate, lPreviousLegDate_ptr = lCurrentLegDate_ptr) {
00091       lCurrentLegDate_ptr = *itLegDate;
00092
00093       // As the boarding point of the current leg is the same as the off point
00094       // of the previous leg (by construction), there is no time difference.
00095       assert (lCurrentLegDate_ptr->getBoardingPoint()
```

```
00096               == lPreviousLegDate_ptr->getOffPoint());
00097       const stdair::Duration_T& lStopOverTime =
00098         lCurrentLegDate_ptr->getBoardingTime() - lPreviousLegDate_ptr->
      getOffTime();
00099       lElapsedTime += lStopOverTime;
00100
00101       // Add the elapsed time of the current leg
00102       const stdair::Duration_T& currentElapsedTime =
00103         lCurrentLegDate_ptr->getElapsedTime();
00104       lElapsedTime += currentElapsedTime;
00105     }
00106
00107     // Store the result
00108     ioSegmentDate.setElapsedTime (lElapsedTime);
00109     // From the elapsed time, update the distance
00110     updateDistanceFromElapsedTime (ioSegmentDate);
00111   }
00112
00113   // ////////////////////////////////////////////////////////////////////
00114   void SegmentDateHelper::
00115   updateDistanceFromElapsedTime (
      stdair::SegmentDate& ioSegmentDate) {
00116     const stdair::Duration_T& lElapsedTime = ioSegmentDate.getElapsedTime();
00117     const double lElapseInHours=static_cast<const double>(lElapsedTime.hours())
      ;
00118     const long int lDistance =
00119       static_cast<const long int>(stdair::DEFAULT_FLIGHT_SPEED*lElapseInHours);
00120     ioSegmentDate.setDistance (lDistance);
00121   }
00122
00123 }
```

23.107 airinv/bom/SegmentDateHelper.hpp File Reference

Classes

- class AIRINV::SegmentDateHelper

Namespaces

- namespace stdair

  Forward declarations.
- namespace AIRINV

23.108 SegmentDateHelper.hpp

```
00001 #ifndef __AIRINV_BOM_SEGMENTDATEHELPER_HPP
00002 #define __AIRINV_BOM_SEGMENTDATEHELPER_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007
00008 // Forward declarations
00009 namespace stdair {
00010   class SegmentDate;
00011 }
00012
00013 namespace AIRINV {
00016   class SegmentDateHelper {
00017   public:
00018     // ////////// Business Methods /////////
00021     static void fillFromRouting (stdair::SegmentDate&);
00022
00032     static void updateElapsedTimeFromRouting (
      stdair::SegmentDate&);
00033
00035     static void updateDistanceFromElapsedTime (
      stdair::SegmentDate&);
00036   };
00037
00038 }
00039 #endif // __AIRINV_BOM_SEGMENTDATEHELPER_HPP
```

23.109 airinv/bom/SegmentSnapshotTableHelper.cpp File Reference

```
#include <cassert>
#include <cmath>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomRetriever.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/SegmentSnapshotTable.hpp>
#include <stdair/service/Logger.hpp>
#include <airinv/basic/BasConst_Curves.hpp>
#include <airinv/bom/SegmentSnapshotTableHelper.hpp>
#include <airinv/bom/FlightDateHelper.hpp>
#include <airinv/bom/SegmentCabinHelper.hpp>
```

Namespaces

- namespace AIRINV

23.110 SegmentSnapshotTableHelper.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <cmath>
00007 // StdAir
00008 #include <stdair/basic/BasConst_Inventory.hpp>
00009 #include <stdair/bom/BomRetriever.hpp>
00010 #include <stdair/bom/BomManager.hpp>
00011 #include <stdair/bom/SegmentDate.hpp>
00012 #include <stdair/bom/SegmentCabin.hpp>
00013 #include <stdair/bom/FareFamily.hpp>
00014 #include <stdair/bom/BookingClass.hpp>
00015 #include <stdair/bom/SegmentSnapshotTable.hpp>
00016 #include <stdair/service/Logger.hpp>
00017 // AirInv
00018 #include <airinv/basic/BasConst_Curves.hpp>
00019 #include <airinv/bom/SegmentSnapshotTableHelper.hpp
     >
00020 #include <airinv/bom/FlightDateHelper.hpp>
00021 #include <airinv/bom/SegmentCabinHelper.hpp>
00022
00023 namespace AIRINV {
00024
00025   // ////////////////////////////////////////////////////////////////////
00026   void SegmentSnapshotTableHelper::
00027   takeSnapshots (stdair::SegmentSnapshotTable&
     ioSegmentSnapshotTable,
00028                  const stdair::DateTime_T& iSnapshotTime) {
00029     // Retrieve the segment-cabin index and take the snapshots for
00030     // each segment-cabin.
00031     const stdair::SegmentCabinIndexMap_T& lSegmentCabinIndexMap =
00032       ioSegmentSnapshotTable.getSegmentCabinIndexMap();
00033     for (stdair::SegmentCabinIndexMap_T::const_iterator itSCIdx =
00034            lSegmentCabinIndexMap.begin();
00035          itSCIdx != lSegmentCabinIndexMap.end(); ++itSCIdx) {
00036       const stdair::SegmentCabin* lSC_ptr = itSCIdx->first;
00037       assert (lSC_ptr != NULL);
00038       const stdair::SegmentDataID_T& lSCIdx = itSCIdx->second;
00039
00040       const stdair::Date_T& lSnapshotDate = iSnapshotTime.date();
00041
00042       // Compare the date of the snapshot time and the departure date of
00043       // the segment-cabin in order to verify the necescity of taking
     snapshots.
00044       const stdair::SegmentDate& lSegmentDate =
00045         stdair::BomManager::getParent<stdair::SegmentDate> (*lSC_ptr);
```

```
00046       const stdair::Date_T& lDepartureDate = lSegmentDate.getBoardingDate();
00047       const stdair::DateOffset_T lDateOffset = lDepartureDate - lSnapshotDate;
00048       const stdair::DTD_T lDTD = lDateOffset.days() + 1;
00049
00050       if (lDTD >= 0 && lDTD <= stdair::DEFAULT_MAX_DTD) {
00051         SegmentCabinHelper::updateAvailabilities
     (*lSC_ptr);
00052         takeSnapshots (ioSegmentSnapshotTable, lDTD, *lSC_ptr,
    lSCIdx);
00053         registerProductAndPriceOrientedBookings (ioSegmentSnapshotTable,
00054                                                  lDTD, *lSC_ptr, lSCIdx);
00055       }
00056     }
00057   }
00058
00059   // ////////////////////////////////////////////////////////////////////
00060   void SegmentSnapshotTableHelper::
00061   takeSnapshots (stdair::SegmentSnapshotTable&
    ioSegmentSnapshotTable,
00062                  const stdair::DTD_T& iDTD,
00063                  const stdair::SegmentCabin& iSegmentCabin,
00064                  const stdair::SegmentDataID_T iSegmentCabinIdx) {
00065
00066     // Extract the views for the corresponding DTD and segment-cabin.
00067     stdair::SegmentCabinDTDSnapshotView_T lBookingView = ioSegmentSnapshotTable
    .
00068       getSegmentCabinDTDBookingSnapshotView (iSegmentCabinIdx,
00069                                              iSegmentCabinIdx, iDTD);
00070     stdair::SegmentCabinDTDSnapshotView_T lCancellationView =
    ioSegmentSnapshotTable.
00071       getSegmentCabinDTDCancellationSnapshotView (iSegmentCabinIdx,
00072                                                   iSegmentCabinIdx, iDTD);
00073     stdair::SegmentCabinDTDSnapshotView_T lAvailabilityView =
    ioSegmentSnapshotTable.
00074       getSegmentCabinDTDAvailabilitySnapshotView (iSegmentCabinIdx,
00075                                                   iSegmentCabinIdx, iDTD);
00076
00077     // Retrieve the block index of the segment-cabin.
00078     std::ostringstream lSCMapKey;
00079     lSCMapKey << stdair::DEFAULT_SEGMENT_CABIN_VALUE_TYPE
00080               << iSegmentCabin.describeKey();
00081     const stdair::ClassIndex_T& lCabinIdx =
00082       ioSegmentSnapshotTable.getClassIndex (lSCMapKey.str());
00083     lAvailabilityView[lCabinIdx] = iSegmentCabin.getAvailabilityPool();
00084     lBookingView[lCabinIdx] = iSegmentCabin.getCommittedSpace();
00085
00086
00087     // Browse the booking class list
00088     const stdair::BookingClassList_T& lBCList =
00089       stdair::BomManager::getList<stdair::BookingClass> (iSegmentCabin);
00090     for (stdair::BookingClassList_T::const_iterator itBC = lBCList.begin();
00091          itBC != lBCList.end(); ++itBC) {
00092       const stdair::BookingClass* lBookingClass_ptr = *itBC;
00093       assert (lBookingClass_ptr != NULL);
00094
00095       // Retrieve the block index of the booking class.
00096       const stdair::ClassIndex_T& lIdx =
00097         ioSegmentSnapshotTable.getClassIndex (lBookingClass_ptr->describeKey())
    ;
00098
00099       // DEBUG
00100       // STDAIR_LOG_DEBUG ("Taking snapshot for "
00101       //                   << iSegmentCabin.describeKey() << ", "
00102       //                   << lBookingClass_ptr->describeKey()
00103       //                   << ", DTD: " << iDTD << ", nb of bookings: "
00104       //                   << lBookingClass_ptr->getNbOfBookings());
00105
00106       // Write the snapshot.
00107       lBookingView[lIdx]=lBookingClass_ptr->getNbOfBookings();
00108       lCancellationView[lIdx] =
00109         lBookingClass_ptr->getNbOfCancellations();
00110       lAvailabilityView[lIdx] =
00111         lBookingClass_ptr->getSegmentAvailability();
00112     }
00113   }
00114
00115   // ////////////////////////////////////////////////////////////////////
00116   void SegmentSnapshotTableHelper::registerProductAndPriceOrientedBookings
00117    (stdair::SegmentSnapshotTable& ioSegmentSnapshotTable, const stdair::DTD_T&
    iDTD,
00118     const stdair::SegmentCabin& iSegmentCabin,
00119     const stdair::SegmentDataID_T iSegmentCabinIdx) {
00120
00121     // Extract the views for the corresponding DTD and segment-cabin.
00122     stdair::SegmentCabinDTDRangeSnapshotView_T lRangeBookingView =
00123       ioSegmentSnapshotTable.getSegmentCabinDTDRangeBookingSnapshotView (
    iSegmentCabinIdx, iSegmentCabinIdx, iDTD, iDTD + 1);
```

```
00124     stdair::SegmentCabinDTDRangeSnapshotView_T lRangeCancellationView =
00125       ioSegmentSnapshotTable.getSegmentCabinDTDRangeCancellationSnapshotView (
    iSegmentCabinIdx, iSegmentCabinIdx, iDTD, iDTD + 1);
00126     stdair::SegmentCabinDTDSnapshotView_T lProductOrientedGrossBookingView =
00127       ioSegmentSnapshotTable.
    getSegmentCabinDTDProductOrientedGrossBookingSnapshotView (iSegmentCabinIdx, iSegmentCabinIdx, iDTD);
00128     stdair::SegmentCabinDTDSnapshotView_T lPriceOrientedGrossBookingView =
00129       ioSegmentSnapshotTable.
    getSegmentCabinDTDPriceOrientedGrossBookingSnapshotView (iSegmentCabinIdx, iSegmentCabinIdx, iDTD);
00130     stdair::SegmentCabinDTDSnapshotView_T lProductOrientedNetBookingView =
00131       ioSegmentSnapshotTable.
    getSegmentCabinDTDProductOrientedNetBookingSnapshotView (iSegmentCabinIdx, iSegmentCabinIdx, iDTD);
00132     stdair::SegmentCabinDTDSnapshotView_T lPriceOrientedNetBookingView =
00133       ioSegmentSnapshotTable.
    getSegmentCabinDTDPriceOrientedNetBookingSnapshotView (iSegmentCabinIdx, iSegmentCabinIdx, iDTD);
00134
00135     // Retrieve the lowest yield and the lowest class and treat the number of
     gross
00136     // bookings of this class the price oriented bookings.
00137     const stdair::BookingClassList_T& lBCList =
00138       stdair::BomManager::getList<stdair::BookingClass> (iSegmentCabin);
00139     stdair::BookingClassList_T::const_iterator iterBC = lBCList.begin();
00140     assert (iterBC != lBCList.end());
00141     stdair::BookingClass* lLowestClass_ptr = *iterBC;
00142     assert (lLowestClass_ptr  != NULL);
00143     stdair::Yield_T lLowestYield = lLowestClass_ptr->getYield();
00144     for (; iterBC != lBCList.end(); iterBC++) {
00145       const stdair::BookingClass* lBookingClass_ptr = *iterBC;
00146       assert (lBookingClass_ptr != NULL);
00147       const stdair::Yield_T& lYield = lBookingClass_ptr->getYield();
00148       if (lYield < lLowestYield) {
00149         lLowestYield = lYield;
00150         lLowestClass_ptr = *iterBC;
00151       }
00152     }
00153     assert (lLowestClass_ptr != NULL);
00154
00155     // Retrieve the block index of the booking class.
00156     const stdair::ClassIndex_T& lClassIdx =
00157       ioSegmentSnapshotTable.getClassIndex (lLowestClass_ptr->describeKey());
00158
00159     // Compute the number of gross bookings for this class.
00160     const stdair::NbOfBookings_T lNbOfNetBkgs =
00161       lRangeBookingView[lClassIdx][0] - lRangeBookingView[lClassIdx][1];
00162     const stdair::NbOfCancellations_T lNbOfCx =
00163       lRangeCancellationView[lClassIdx][0]-lRangeCancellationView[lClassIdx][1]
    ;
00164     const stdair::NbOfBookings_T lNbOfGrossBkgs = lNbOfNetBkgs + lNbOfCx;
00165
00166     // Write these numbern of bookings to the price-oriented value.
00167     lPriceOrientedGrossBookingView[lClassIdx] = lNbOfGrossBkgs;
00168     lPriceOrientedNetBookingView[lClassIdx] = lNbOfNetBkgs;
00169
00170     // Boolean for "no lower class available" verification.
00171     bool noLowerClassAvl = true;
00172     if (lLowestClass_ptr->getSegmentAvailability() >= 1.0) {
00173       noLowerClassAvl = false;
00174     }
00175
00176     // Browse the booking class list
00177     stdair::BookingClassList_T::const_reverse_iterator itBC = lBCList.rbegin();
00178     for (; itBC != lBCList.rend(); itBC++) {
00179       const stdair::BookingClass* lBookingClass_ptr = *itBC;
00180       assert (lBookingClass_ptr != NULL);
00181
00182       // Retrieve the yield of the this class.
00183       const stdair::Yield_T& lYield = lBookingClass_ptr->getYield();
00184       assert (lYield >= lLowestYield);
00185       if (lYield > lLowestYield) {
00186
00187         // Retrieve the block index of the booking class.
00188         const stdair::ClassIndex_T& lIdx =
00189           ioSegmentSnapshotTable.getClassIndex (lBookingClass_ptr->describeKey(
    ));
00190
00191         // Compute the number of gross bookings for this class.
00192         const stdair::NbOfBookings_T lNetBkgs =
00193           lRangeBookingView[lIdx][0] - lRangeBookingView[lIdx][1];
00194         const stdair::NbOfCancellations_T lCx =
00195           lRangeCancellationView[lIdx][0] - lRangeCancellationView[lIdx][1];
00196         const stdair::NbOfBookings_T lGrossBkgs = lNetBkgs + lCx;
00197
00198         // If there is a lower class available, these gross bookings
00199         // will be considered product-oriented. Otherwise, they will be
00200         // considered price-oriented
00201         if (noLowerClassAvl == false) {
00202           lProductOrientedGrossBookingView[lIdx] = lGrossBkgs;
```

```
00203           lProductOrientedNetBookingView[lIdx] = lNetBkgs;
00204         } else {
00205           lPriceOrientedGrossBookingView[lIdx] = lGrossBkgs;
00206           lPriceOrientedNetBookingView[lIdx] = lNetBkgs;
00207
00208           if (lBookingClass_ptr->getSegmentAvailability() >= 1.0) {
00209             noLowerClassAvl = false;
00210           }
00211         }
00212       }
00213     }
00214   }
00215
00216 }
```

23.111 airinv/bom/SegmentSnapshotTableHelper.hpp File Reference

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

Classes

- class AIRINV::SegmentSnapshotTableHelper

Namespaces

- namespace stdair

  Forward declarations.
- namespace AIRINV

23.112 SegmentSnapshotTableHelper.hpp

```
00001 #ifndef __AIRINV_BOM_SEGMENTSNAPSHOTTABLEHELPER_HPP
00002 #define __AIRINV_BOM_SEGMENTSNAPSHOTTABLEHELPER_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/stdair_basic_types.hpp>
00011
00012 // Forward declarations
00013 namespace stdair {
00014   class SegmentSnapshotTable;
00015   class SegmentCabin;
00016 }
00017
00018 namespace AIRINV {
00019
00022   class SegmentSnapshotTableHelper {
00023   public:
00024     // ********** Business Methods *********
00026     static void takeSnapshots (stdair::SegmentSnapshotTable&,
00027                                const stdair::DateTime_T&);
00028   private:
00029     // ********** Helpers for business methods. **********
00031     static void takeSnapshots (stdair::SegmentSnapshotTable&,
00032                                const stdair::DTD_T&,
00033                                const stdair::SegmentCabin&,
00034                                const stdair::SegmentDataID_T);
00035
00037     static void registerProductAndPriceOrientedBookings
00038     (stdair::SegmentSnapshotTable&, const stdair::DTD_T&,
00039      const stdair::SegmentCabin&, const stdair::SegmentDataID_T);
00040   };
00041
00042 }
00043 #endif // __AIRINV_BOM_SEGMENTSNAPSHOTTABLEHELPER_HPP
```

23.113 airinv/bom/SegmentStruct.cpp File Reference

```
#include <cassert>
#include <stdair/bom/SegmentDate.hpp>
#include <airinv/bom/SegmentStruct.hpp>
```

Namespaces

- namespace AIRINV

23.114 SegmentStruct.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // STDAIR
00007 #include <stdair/bom/SegmentDate.hpp>
00008 // AIRINV
00009 #include <airinv/bom/SegmentStruct.hpp>
00010
00011 namespace AIRINV {
00012
00013   // ////////////////////////////////////////////////////////////////////
00014   const std::string SegmentStruct::describe() const {
00015     std::ostringstream ostr;
00016
00017     ostr << "    " << _boardingPoint << " / "
00018          << boost::posix_time::to_simple_string(_boardingTime)
00019          << " -- " << _offPoint << " / "
00020          << boost::posix_time::to_simple_string(_offTime)
00021          << " --> "
00022          << boost::posix_time::to_simple_string(_elapsed)
00023          << std::endl;
00024
00025     for (SegmentCabinStructList_T::const_iterator itCabin =
00026            _cabinList.begin(); itCabin != _cabinList.end();
       itCabin++) {
00027       const SegmentCabinStruct& lCabin = *itCabin;
00028       ostr << lCabin.describe();
00029     }
00030     ostr << std::endl;
00031
00032     return ostr.str();
00033   }
00034
00035   // ////////////////////////////////////////////////////////////////////
00036   void SegmentStruct::fill (stdair::SegmentDate&
     ioSegmentDate) const {
00037     // Set the Boarding Date
00038     ioSegmentDate.setBoardingDate (_offDate);
00039     // Set the Boarding Time
00040     ioSegmentDate.setBoardingTime (_boardingTime);
00041     // Set the Off Date
00042     ioSegmentDate.setOffDate (_offDate);
00043     // Set the Off Time
00044     ioSegmentDate.setOffTime (_offTime);
00045     // Set the Elapsed Time
00046     ioSegmentDate.setElapsedTime (_elapsed);
00047   }
00048
00049 }
```

23.115 airinv/bom/SegmentStruct.hpp File Reference

```
#include <string>
#include <vector>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <airinv/bom/SegmentCabinStruct.hpp>
```

Classes

- struct AIRINV::SegmentStruct

Namespaces

- namespace stdair

  Forward declarations.
- namespace AIRINV

Typedefs

- typedef std::vector
  < SegmentStruct > AIRINV::SegmentStructList_T

23.116 SegmentStruct.hpp

```
00001 #ifndef __AIRINV_BOM_SEGMENTSTRUCT_HPP
00002 #define __AIRINV_BOM_SEGMENTSTRUCT_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010 // STDAIR
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/basic/StructAbstract.hpp>
00013 // AIRINV
00014 #include <airinv/bom/SegmentCabinStruct.hpp>
00015
00016 // Forward declarations
00017 namespace stdair {
00018   class SegmentDate;
00019 }
00020
00021 namespace AIRINV {
00023   struct SegmentStruct : public stdair::StructAbstract {
00024     // Attributes
00025     stdair::AirportCode_T _boardingPoint;
00026     stdair::AirportCode_T _offPoint;
00027     stdair::Date_T _boardingDate;
00028     stdair::Duration_T _boardingTime;
00029     stdair::Date_T _offDate;
00030     stdair::Duration_T _offTime;
00031     stdair::Duration_T _elapsed;
00032     SegmentCabinStructList_T _cabinList;
00033
00036     void fill (stdair::SegmentDate&) const;
00037
00039     const std::string describe() const;
00040   };
00041
00043   typedef std::vector<SegmentStruct> SegmentStructList_T;
00044
00045 }
00046 #endif // __AIRINV_BOM_SEGMENTSTRUCT_HPP
```

23.117 airinv/command/FFDisutilityParser.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/service/Logger.hpp>
#include <airinv/command/FFDisutilityParserHelper.hpp>
#include <airinv/command/FFDisutilityParser.hpp>
#include <airinv/command/InventoryManager.hpp>
```

Namespaces

- namespace AIRINV

23.118 FFDisutilityParser.cpp

```
00001 // //////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasFileMgr.hpp>
00009 #include <stdair/bom/BomRoot.hpp>
00010 #include <stdair/service/Logger.hpp>
00011 // Airinv
00012 #include <airinv/command/FFDisutilityParserHelper.hpp
      >
00013 #include <airinv/command/FFDisutilityParser.hpp
      >
00014 #include <airinv/command/InventoryManager.hpp
      >
00015
00016 namespace AIRINV {
00017
00018   // //////////////////////////////////////////////////////////////////
00019   void FFDisutilityParser::
00020   parse (const stdair::FFDisutilityFilePath& iFFDisutilityFilename,
00021          stdair::BomRoot& ioBomRoot) {
00022
00023     const stdair::Filename_T lFilename = iFFDisutilityFilename.name();
00024
00025     // Check that the file path given as input corresponds to an actual file
00026     bool doesExistAndIsReadable =
00027       stdair::BasFileMgr::doesExistAndIsReadable (lFilename);
00028     if (doesExistAndIsReadable == false) {
00029       std::ostringstream oMessage;
00030       oMessage << "The FF disutility input file, '" << lFilename
00031                << "', can not be retrieved on the file-system";
00032       STDAIR_LOG_ERROR (oMessage.str());
00033       throw FFDisutilityInputFileNotFoundException
       (oMessage.str());
00034     }
00035
00036     // Initialise the FFDisutility file parser.
00037     FFDisutilityFileParser lFFDisutilityParser (ioBomRoot
      , lFilename);
00038
00039     // Parse the CSV-formatted FFDisutility input file, and generate the
00040     // corresponding FFDisutility curves.
00041     lFFDisutilityParser.generateFFDisutilityCurves ()
      ;
00042   }
00043 }
```

23.119 airinv/command/FFDisutilityParser.hpp File Reference

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

Classes

- class AIRINV::FFDisutilityParser

  Class wrapping the parser entry point.

Namespaces

- namespace stdair

  Forward declarations.
- namespace AIRINV

23.120 FFDisutilityParser.hpp

```
00001 #ifndef __AIRINV_CMD_FFDISUTILITYPARSER_HPP
00002 #define __AIRINV_CMD_FFDISUTILITYPARSER_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009 #include <stdair/stdair_file.hpp>
00010 #include <stdair/command/CmdAbstract.hpp>
00011
00013 namespace stdair {
00014   class BomRoot;
00015 }
00016
00017 namespace AIRINV {
00018
00022   class FFDisutilityParser : public stdair::CmdAbstract {
00023   public:
00032     static void parse (const stdair::FFDisutilityFilePath&
      iFFDisutilityInputFilename,
00033                        stdair::BomRoot&);
00034   };
00035 }
00036 #endif // __AIRINV_CMD_FFDISUTILITYPARSER_HPP
```

23.121 airinv/command/FFDisutilityParserHelper.cpp File Reference

```
#include <cassert>
#include <ostream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/stdair_types.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/service/Logger.hpp>
#include <airinv/command/FFDisutilityParserHelper.hpp>
```

Namespaces

- namespace AIRINV
- namespace AIRINV::FFDisutilityParserHelper

Functions

- repeat_p_t AIRINV::FFDisutilityParserHelper::key_p (chset_t("0-9A-Z").derived(), 1, 10)

23.122 FFDisutilityParserHelper.cpp

```
00001 // //////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <ostream>
00007 // StdAir
00008 #include <stdair/stdair_exceptions.hpp>
00009 #include <stdair/stdair_types.hpp>
```

```
00010 #include <stdair/bom/BomRoot.hpp>
00011 #include <stdair/service/Logger.hpp>
00012 // #define BOOST_SPIRIT_DEBUG
00013 #include <airinv/command/FFDisutilityParserHelper.hpp
      >
00014
00015 //
00016 namespace bsc = boost::spirit::classic;
00017
00018 namespace AIRINV {
00019
00020   namespace FFDisutilityParserHelper {
00021
00022     // ////////////////////////////////////////////////////////////////////
00023     //  Semantic actions
00024     // ////////////////////////////////////////////////////////////////////
00025
00026     ParserSemanticAction::
00027     ParserSemanticAction (FFDisutilityStruct
      & ioFFDisutility)
00028       : _ffDisutility (ioFFDisutility) {
00029     }
00030
00031     // ////////////////////////////////////////////////////////////////////
00032     storeCurveKey::
00033     storeCurveKey (FFDisutilityStruct&
      ioFFDisutility)
00034       : ParserSemanticAction (ioFFDisutility) {
00035     }
00036
00037     // ////////////////////////////////////////////////////////////////////
00038     void storeCurveKey::operator() (iterator_t
       iStr,
00039                                     iterator_t iStrEnd) const {
00040       const std::string lKey (iStr, iStrEnd);
00041       _ffDisutility._key = lKey;
00042     }
00043
00044     // ////////////////////////////////////////////////////////////////////
00045     storeDTD::storeDTD (FFDisutilityStruct&
       ioFFDisutility)
00046       : ParserSemanticAction (ioFFDisutility) {
00047     }
00048
00049     // ////////////////////////////////////////////////////////////////////
00050     void storeDTD::operator() (int iDTD) const {
00051       _ffDisutility._dtd = iDTD;
00052       //STDAIR_LOG_DEBUG ("DTD: " << iDTD);
00053     }
00054
00055     // ////////////////////////////////////////////////////////////////////
00056     storeFFDisutilityValue::storeFFDisutilityValue
       (FFDisutilityStruct& ioFFDisutility)
00057       : ParserSemanticAction (ioFFDisutility) {
00058     }
00059
00060     // ////////////////////////////////////////////////////////////////////
00061     void storeFFDisutilityValue::operator()
      (double iReal) const {
00062       const bool hasInsertBeenSuccessfull =
00063         _ffDisutility._curve.
00064         insert (stdair::FFDisutilityCurve_T::
00065                 value_type (_ffDisutility._dtd, iReal)).second
      ;
00066       if (hasInsertBeenSuccessfull == false) {
00067         std::ostringstream oStr;
00068         oStr << "The same DTD ('" << _ffDisutility._dtd
00069              << "') has probably been given twice";
00070         STDAIR_LOG_ERROR (oStr.str());
00071         throw stdair::KeyDuplicationException (oStr.str());
00072       }
00073
00074       //STDAIR_LOG_DEBUG ("PosProbMass: " << iReal);
00075     }
00076
00077     // ////////////////////////////////////////////////////////////////////
00078     doEndCurve::
00079     doEndCurve (stdair::BomRoot& ioBomRoot,
00080                 FFDisutilityStruct& ioFFDisutility)
00081       : ParserSemanticAction (ioFFDisutility),
00082         _bomRoot (ioBomRoot) {
00083     }
00084
00085     // ////////////////////////////////////////////////////////////////////
00086     // void doEndCurve::operator() (char iChar) const {
00087     void doEndCurve::operator() (iterator_t
      iStr,
```

```
00088                                 iterator_t iStrEnd) const {
00089       // DEBUG: Display the result
00090       STDAIR_LOG_DEBUG ("FFDisutility: " << _ffDisutility.describe
      ());
00091
00092       // Add the curve to the BomRoot.
00093       _bomRoot.addFFDisutilityCurve (_ffDisutility._key
      , _ffDisutility._curve);
00094
00095       // As that's the end of a curve, the values must be cleared.
00096       _ffDisutility._curve.clear();
00097     }
00098
00099
00100     // ////////////////////////////////////////////////////////////////////
00101     //
00102     //  Utility Parsers
00103     //
00104     // ////////////////////////////////////////////////////////////////////
00106     repeat_p_t key_p (chset_t("0-9A-Z").derived(), 1, 10)
      ;
00107
00108     // ////////////////////////////////////////////////////////////////////
00109     //  (Boost Spirit) Grammar Definition
00110     // ////////////////////////////////////////////////////////////////////
00111
00112     // ////////////////////////////////////////////////////////////////////
00113     FFDisutilityParser::
00114     FFDisutilityParser (stdair::BomRoot& ioBomRoot,
00115                         FFDisutilityStruct& ioFFDisutility)
00116       : _bomRoot (ioBomRoot),
00117         _ffDisutility (ioFFDisutility) {
00118     }
00119
00120     // ////////////////////////////////////////////////////////////////////
00121     template<typename ScannerT>
00122     FFDisutilityParser::definition<ScannerT>::
00123     definition (FFDisutilityParser const& self)
       {
00124
00125       curve_list = *( not_to_be_parsed | curve )
00126         ;
00127
00128       not_to_be_parsed =
00129         bsc::lexeme_d[ bsc::comment_p("//") | bsc::comment_p("/*", "*/")
00130                        | bsc::space_p ]
00131         ;
00132
00133       curve = key >> ';' >> map
00134                   >> curve_end[doEndCurve(self._bomRoot, self.
      _ffDisutility)]
00135         ;
00136
00137       curve_end = bsc::ch_p(';')
00138         ;
00139
00140       key =
00141         bsc::lexeme_d[(key_p)[storeCurveKey(self.
      _ffDisutility)]]
00142         ;
00143
00144       map =
00145         value_pair >> *( ';' >> value_pair)
00146         ;
00147
00148       value_pair = bsc::uint_p[storeDTD(self._ffDisutility)]
00149         >> ":" >> bsc::ureal_p[storeFFDisutilityValue(
      self._ffDisutility)]
00150         ;
00151
00152       // BOOST_SPIRIT_DEBUG_NODE (FFDisutilityParser);
00153       BOOST_SPIRIT_DEBUG_NODE (curve_list);
00154       BOOST_SPIRIT_DEBUG_NODE (not_to_be_parsed);
00155       BOOST_SPIRIT_DEBUG_NODE (key);
00156       BOOST_SPIRIT_DEBUG_NODE (map);
00157       BOOST_SPIRIT_DEBUG_NODE (value_pair);
00158     }
00159
00160     // ////////////////////////////////////////////////////////////////////
00161     template<typename ScannerT>
00162     bsc::rule<ScannerT> const&
00163     FFDisutilityParser::definition<ScannerT>::start
      () const {
00164       return curve_list;
00165     }
00166   }
00167
```

```
00168
00170   //
00171   //  Entry class for the file parser
00172   //
00174
00175   // ////////////////////////////////////////////////////////////////////
00176   FFDisutilityFileParser::
00177   FFDisutilityFileParser (stdair::BomRoot& ioBomRoot,
00178                           const stdair::Filename_T& iFilename)
00179     : _filename (iFilename), _bomRoot (ioBomRoot) {
00180     init();
00181   }
00182
00183   // ////////////////////////////////////////////////////////////////////
00184   void FFDisutilityFileParser::init() {
00185     // Open the file
00186     _startIterator = iterator_t (_filename);
00187
00188     // Check the filename exists and can be open
00189     if (!_startIterator) {
00190       std::ostringstream oMessage;
00191       oMessage << "The file " << _filename << " can not be open." << std::endl;
00192       STDAIR_LOG_ERROR (oMessage.str());
00193       throw FFDisutilityInputFileNotFoundException
       (oMessage.str());
00194     }
00195
00196     // Create an EOF iterator
00197     _endIterator = _startIterator.make_end();
00198   }
00199
00200   // ////////////////////////////////////////////////////////////////////
00201   bool FFDisutilityFileParser::generateFFDisutilityCurves
       () {
00202     bool oResult = false;
00203
00204     STDAIR_LOG_DEBUG ("Parsing FFDisutility input file: " << _filename);
00205
00206     // Initialise the parser (grammar) with the helper/staging structure.
00207     FFDisutilityParserHelper::FFDisutilityParser
       lFFDisutilityParser (_bomRoot, _ffDisutility);
00208
00209     // Launch the parsing of the file and, thanks to the doEndCurve
00210     // call-back structure, the building of the whole BomRoot BOM
00211     // (i.e., including Inventory, FlightDate, LegDate, SegmentDate, etc.)
00212     bsc::parse_info<iterator_t> info = bsc::parse (_startIterator, _endIterator
,
00213                                                    lFFDisutilityParser,
00214                                                    bsc::space_p - bsc::eol_p);
00215
00216     // Retrieves whether or not the parsing was successful
00217     oResult = info.hit;
00218
00219     const bool isFull = info.full;
00220
00221     const std::string hasBeenFullyReadStr = (isFull  == true)?"":"not ";
00222     if (oResult == true && isFull == true) {
00223       STDAIR_LOG_DEBUG ("Parsing of FFDisutility input file: " << _filename
00224                         << " succeeded: read " << info.length
00225                         << " characters. The input file has "
00226                         << hasBeenFullyReadStr
00227                         << "been fully read. Stop point: " << info.stop);
00228
00229     } else {
00230       STDAIR_LOG_ERROR ("Parsing of FFDisutility input file: " << _filename
00231                         << " failed: read " << info.length
00232                         << " characters. The input file has "
00233                         << hasBeenFullyReadStr
00234                         << "been fully read. Stop point: " << info.stop);
00235       throw FFDisutilityFileParsingFailedException
       ("Parsing of FFDisutility input file: " + _filename + " failed.");
00236     }
00237
00238     return oResult;
00239   }
00240
00241 }
```

23.123 airinv/command/FFDisutilityParserHelper.hpp File Reference

```
#include <string>
```

```
#include <stdair/command/CmdAbstract.hpp>
#include <airinv/AIRINV_Types.hpp>
#include <airinv/basic/BasParserTypes.hpp>
#include <airinv/bom/FFDisutilityStruct.hpp>
```

Classes

- struct AIRINV::FFDisutilityParserHelper::ParserSemanticAction
- struct AIRINV::FFDisutilityParserHelper::storeCurveKey
- struct AIRINV::FFDisutilityParserHelper::storeDTD
- struct AIRINV::FFDisutilityParserHelper::storeFFDisutilityValue
- struct AIRINV::FFDisutilityParserHelper::doEndCurve
- struct AIRINV::FFDisutilityParserHelper::FFDisutilityParser
- struct AIRINV::FFDisutilityParserHelper::FFDisutilityParser::definition< ScannerT >
- class AIRINV::FFDisutilityFileParser

Namespaces

- namespace stdair

  Forward declarations.
- namespace AIRINV
- namespace AIRINV::FFDisutilityParserHelper

23.124 FFDisutilityParserHelper.hpp

```
00001 #ifndef __AIRINV_CMD_FFDISUTILITYPARSERHELPER_HPP
00002 #define __AIRINV_CMD_FFDISUTILITYPARSERHELPER_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/command/CmdAbstract.hpp>
00011 // Airinv
00012 #include <airinv/AIRINV_Types.hpp>
00013 #include <airinv/basic/BasParserTypes.hpp>
00014 #include <airinv/bom/FFDisutilityStruct.hpp>
00015
00016 // Forward declarations
00017 namespace stdair {
00018   class BomRoot;
00019 }
00020
00021 namespace AIRINV {
00022
00023   namespace FFDisutilityParserHelper {
00024
00025     // //////////////////////////////////////////////////////////////////
00026     //  Semantic actions
00027     // //////////////////////////////////////////////////////////////////
00029     struct ParserSemanticAction {
00031       ParserSemanticAction (FFDisutilityStruct
      &);
00033       FFDisutilityStruct& _ffDisutility;
00034     };
00035
00037     struct storeCurveKey : public ParserSemanticAction
      {
00039       storeCurveKey (FFDisutilityStruct&);
00041       void operator() (iterator_t iStr, iterator_t
      iStrEnd) const;
00042     };
00043
00045     struct storeDTD : public ParserSemanticAction {
00047       storeDTD (FFDisutilityStruct&);
00049       void operator() (int iDTD) const;
00050     };
```

```
00051
00053     struct storeFFDisutilityValue : public
      ParserSemanticAction {
00055       storeFFDisutilityValue (FFDisutilityStruct
      &);
00057       void operator() (double iReal) const;
00058     };
00059
00061     struct doEndCurve : public ParserSemanticAction
       {
00063       doEndCurve (stdair::BomRoot&, FFDisutilityStruct
      &);
00065       void operator() (iterator_t iStr, iterator_t
       iStrEnd) const;
00067       stdair::BomRoot& _bomRoot;
00068     };
00069
00071     //
00072     //  (Boost Spirit) Grammar Definition
00073     //
00075
00089     struct FFDisutilityParser :
00090       public boost::spirit::classic::grammar<FFDisutilityParser> {
00091
00092       FFDisutilityParser (stdair::BomRoot&,
      FFDisutilityStruct&);
00093
00094       template <typename ScannerT>
00095       struct definition {
00096         definition (FFDisutilityParser const& self)
      ;
00097
00098         // Instantiation of rules
00099         boost::spirit::classic::rule<ScannerT> curve_list,
00100           not_to_be_parsed, curve, key, map,
      value_pair, curve_end;
00101
00103         boost::spirit::classic::rule<ScannerT> const& start() const;
00104       };
00105
00106       // Parser Context
00107       stdair::BomRoot& _bomRoot;
00108       FFDisutilityStruct& _ffDisutility;
00109     };
00110   }
00111
00116
00117   //
00118   //  Entry class for the file parser
00119   //
00121
00126   class FFDisutilityFileParser : public
      stdair::CmdAbstract {
00127   public:
00129     FFDisutilityFileParser (stdair::BomRoot& ioBomRoot,
00130                             const stdair::Filename_T& iFilename);
00131
00133     bool generateFFDisutilityCurves ();
00134
00135   private:
00137     void init();
00138
00139   private:
00140     // Attributes
00142     stdair::Filename_T _filename;
00143
00145     iterator_t _startIterator;
00146
00148     iterator_t _endIterator;
00149
00151     stdair::BomRoot& _bomRoot;
00152
00154     FFDisutilityStruct _ffDisutility;
00155   };
00156
00157 }
00158 #endif // __AIRINV_CMD_FFDISUTILITYPARSERHELPER_HPP
```

23.125 airinv/command/FRAT5Parser.cpp File Reference

```
#include <cassert>
```

```
#include <sstream>
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/service/Logger.hpp>
#include <airinv/command/FRAT5ParserHelper.hpp>
#include <airinv/command/FRAT5Parser.hpp>
#include <airinv/command/InventoryManager.hpp>
```

Namespaces

- namespace AIRINV

23.126 FRAT5Parser.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasFileMgr.hpp>
00009 #include <stdair/bom/BomRoot.hpp>
00010 #include <stdair/service/Logger.hpp>
00011 // Airinv
00012 #include <airinv/command/FRAT5ParserHelper.hpp
      >
00013 #include <airinv/command/FRAT5Parser.hpp>
00014 #include <airinv/command/InventoryManager.hpp
      >
00015
00016 namespace AIRINV {
00017
00018   // ////////////////////////////////////////////////////////////////////
00019   void FRAT5Parser::
00020   parse (const stdair::FRAT5FilePath& iFRAT5Filename,
00021          stdair::BomRoot& ioBomRoot) {
00022
00023     const stdair::Filename_T lFilename = iFRAT5Filename.name();
00024
00025     // Check that the file path given as input corresponds to an actual file
00026     bool doesExistAndIsReadable =
00027       stdair::BasFileMgr::doesExistAndIsReadable (lFilename);
00028     if (doesExistAndIsReadable == false) {
00029       std::ostringstream oMessage;
00030       oMessage << "The FRAT5 input file, '" << lFilename
00031                << "', can not be retrieved on the file-system";
00032       STDAIR_LOG_ERROR (oMessage.str());
00033       throw FRAT5InputFileNotFoundException (
      oMessage.str());
00034     }
00035
00036     // Initialise the FRAT5 file parser.
00037     FRAT5FileParser lFRAT5Parser (ioBomRoot, lFilename);
00038
00039     // Parse the CSV-formatted FRAT5 input file, and generate the
00040     // corresponding FRAT5 curves.
00041     lFRAT5Parser.generateFRAT5Curves ();
00042   }
00043 }
```

23.127 airinv/command/FRAT5Parser.hpp File Reference

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

Classes

- class AIRINV::FRAT5Parser

  Class wrapping the parser entry point.

Namespaces

- namespace stdair

  Forward declarations.
- namespace AIRINV

23.128 FRAT5Parser.hpp

```
00001 #ifndef __AIRINV_CMD_FRAT5PARSER_HPP
00002 #define __AIRINV_CMD_FRAT5PARSER_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009 #include <stdair/stdair_file.hpp>
00010 #include <stdair/command/CmdAbstract.hpp>
00011
00013 namespace stdair {
00014   class BomRoot;
00015 }
00016
00017 namespace AIRINV {
00018
00022   class FRAT5Parser : public stdair::CmdAbstract {
00023   public:
00032     static void parse (const stdair::FRAT5FilePath& iFRAT5InputFilename,
00033                        stdair::BomRoot&);
00034   };
00035 }
00036 #endif // __AIRINV_CMD_FRAT5PARSER_HPP
```

23.129 airinv/command/FRAT5ParserHelper.cpp File Reference

```
#include <cassert>
#include <ostream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/stdair_types.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/service/Logger.hpp>
#include <airinv/command/FRAT5ParserHelper.hpp>
```

Namespaces

- namespace AIRINV
- namespace AIRINV::FRAT5ParserHelper

Functions

- repeat_p_t AIRINV::FRAT5ParserHelper::key_p (chset_t("0-9A-Z").derived(), 1, 10)

23.130 FRAT5ParserHelper.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <ostream>
00007 // StdAir
00008 #include <stdair/stdair_exceptions.hpp>
00009 #include <stdair/stdair_types.hpp>
00010 #include <stdair/bom/BomRoot.hpp>
00011 #include <stdair/service/Logger.hpp>
00012 // #define BOOST_SPIRIT_DEBUG
00013 #include <airinv/command/FRAT5ParserHelper.hpp
      >
00014
00015 //
00016 namespace bsc = boost::spirit::classic;
00017
00018 namespace AIRINV {
00019
00020   namespace FRAT5ParserHelper {
00021
00022     // ////////////////////////////////////////////////////////////////////
00023     //  Semantic actions
00024     // ////////////////////////////////////////////////////////////////////
00025
00026     ParserSemanticAction::
00027     ParserSemanticAction (FRAT5Struct&
      ioFRAT5)
00028       : _frat5 (ioFRAT5) {
00029     }
00030
00031     // ////////////////////////////////////////////////////////////////////
00032     storeCurveKey::
00033     storeCurveKey (FRAT5Struct& ioFRAT5)
00034       : ParserSemanticAction (ioFRAT5) {
00035     }
00036
00037     // ////////////////////////////////////////////////////////////////////
00038     void storeCurveKey::operator() (iterator_t
       iStr,
00039                                     iterator_t iStrEnd) const {
00040       const std::string lKey (iStr, iStrEnd);
00041       _frat5._key = lKey;
00042       //STDAIR_LOG_DEBUG ("Key: " << lKey);
00043     }
00044
00045     // ////////////////////////////////////////////////////////////////////
00046     storeDTD::storeDTD (FRAT5Struct& ioFRAT5)
00047       : ParserSemanticAction (ioFRAT5) {
00048     }
00049
00050     // ////////////////////////////////////////////////////////////////////
00051     void storeDTD::operator() (int iDTD) const {
00052       _frat5._dtd = iDTD;
00053       //STDAIR_LOG_DEBUG ("DTD: " << iDTD);
00054     }
00055
00056     // ////////////////////////////////////////////////////////////////////
00057     storeFRAT5Value::storeFRAT5Value (
      FRAT5Struct& ioFRAT5)
00058       : ParserSemanticAction (ioFRAT5) {
00059     }
00060
00061     // ////////////////////////////////////////////////////////////////////
00062     void storeFRAT5Value::operator() (double iReal)
       const {
00063       const bool hasInsertBeenSuccessfull =
00064         _frat5._curve.
00065         insert (stdair::FRAT5Curve_T::
00066                 value_type (_frat5._dtd, iReal)).second;
00067       if (hasInsertBeenSuccessfull == false) {
00068         std::ostringstream oStr;
00069         oStr << "The same DTD ('" << _frat5._dtd
00070              << "') has probably been given twice";
00071         STDAIR_LOG_ERROR (oStr.str());
00072         throw stdair::KeyDuplicationException (oStr.str());
00073       }
00074
00075       //STDAIR_LOG_DEBUG ("Value: " << iReal);
00076     }
00077
00078     // ////////////////////////////////////////////////////////////////////
00079     doEndCurve::
00080     doEndCurve (stdair::BomRoot& ioBomRoot,
```

```
00081                 FRAT5Struct& ioFRAT5)
00082       : ParserSemanticAction (ioFRAT5),
00083         _bomRoot (ioBomRoot) {
00084     }
00085
00086     // ////////////////////////////////////////////////////////////////////
00087     // void doEndCurve::operator() (char iChar) const {
00088     void doEndCurve::operator() (iterator_t
    iStr,
00089                                  iterator_t iStrEnd) const {
00090       // DEBUG: Display the result
00091       STDAIR_LOG_DEBUG ("FRAT5: " << _frat5.describe());
00092
00093       // Add the curve to the BomRoot.
00094       _bomRoot.addFRAT5Curve (_frat5._key, _frat5.
    _curve);
00095
00096       // As that's the end of a curve, the values must be cleared.
00097       _frat5._curve.clear();
00098     }
00099
00100
00101     // ////////////////////////////////////////////////////////////////////
00102     //
00103     //  Utility Parsers
00104     //
00105     // ////////////////////////////////////////////////////////////////////
00107     repeat_p_t key_p (chset_t("0-9A-Z").derived(), 1, 10)
    ;
00108
00109     // ////////////////////////////////////////////////////////////////////
00110     //  (Boost Spirit) Grammar Definition
00111     // ////////////////////////////////////////////////////////////////////
00112
00113     // ////////////////////////////////////////////////////////////////////
00114     FRAT5Parser::
00115     FRAT5Parser (stdair::BomRoot& ioBomRoot,
00116                  FRAT5Struct& ioFRAT5)
00117       : _bomRoot (ioBomRoot),
00118         _frat5 (ioFRAT5) {
00119     }
00120
00121     // ////////////////////////////////////////////////////////////////////
00122     template<typename ScannerT>
00123     FRAT5Parser::definition<ScannerT>::
00124     definition (FRAT5Parser const& self) {
00125
00126       curve_list = *( not_to_be_parsed | curve )
00127         ;
00128
00129       not_to_be_parsed =
00130         bsc::lexeme_d[ bsc::comment_p("//") | bsc::comment_p("/*", "*/")
00131                        | bsc::space_p ]
00132         ;
00133
00134       curve = key >> ';' >> map
00135                   >> curve_end[doEndCurve(self._bomRoot, self._frat5)
    ]
00136         ;
00137
00138       curve_end = bsc::ch_p(';')
00139         ;
00140
00141       key =
00142         bsc::lexeme_d[(key_p)[storeCurveKey(self._frat5)]]
00143         ;
00144
00145       map =
00146         value_pair >> *( ';' >> value_pair)
00147         ;
00148
00149       value_pair = bsc::uint_p[storeDTD(self._frat5)]
00150         >> ":" >> bsc::ureal_p[storeFRAT5Value(self._frat5)]
00151         ;
00152
00153       // BOOST_SPIRIT_DEBUG_NODE (FRAT5Parser);
00154       BOOST_SPIRIT_DEBUG_NODE (curve_list);
00155       BOOST_SPIRIT_DEBUG_NODE (not_to_be_parsed);
00156       BOOST_SPIRIT_DEBUG_NODE (key);
00157       BOOST_SPIRIT_DEBUG_NODE (map);
00158       BOOST_SPIRIT_DEBUG_NODE (value_pair);
00159     }
00160
00161     // ////////////////////////////////////////////////////////////////////
00162     template<typename ScannerT>
00163     bsc::rule<ScannerT> const&
00164     FRAT5Parser::definition<ScannerT>::start
```

```
    () const {
00165       return curve_list;
00166     }
00167   }
00168
00169
00171   //
00172   //  Entry class for the file parser
00173   //
00175
00176   // ////////////////////////////////////////////////////////////////////
00177   FRAT5FileParser::
00178   FRAT5FileParser (stdair::BomRoot& ioBomRoot,
00179                            const stdair::Filename_T& iFilename)
00180     : _filename (iFilename), _bomRoot (ioBomRoot) {
00181     init();
00182   }
00183
00184   // ////////////////////////////////////////////////////////////////////
00185   void FRAT5FileParser::init() {
00186     // Open the file
00187     _startIterator = iterator_t (_filename);
00188
00189     // Check the filename exists and can be open
00190     if (!_startIterator) {
00191       std::ostringstream oMessage;
00192       oMessage << "The file " << _filename << " can not be open." << std::endl;
00193       STDAIR_LOG_ERROR (oMessage.str());
00194       throw FRAT5InputFileNotFoundException (
    oMessage.str());
00195     }
00196
00197     // Create an EOF iterator
00198     _endIterator = _startIterator.make_end();
00199   }
00200
00201   // ////////////////////////////////////////////////////////////////////
00202   bool FRAT5FileParser::generateFRAT5Curves
     () {
00203     bool oResult = false;
00204
00205     STDAIR_LOG_DEBUG ("Parsing FRAT5 input file: " << _filename);
00206
00207     // Initialise the parser (grammar) with the helper/staging structure.
00208     FRAT5ParserHelper::FRAT5Parser lFRAT5Parser (
    _bomRoot, _frat5);
00209
00210     // Launch the parsing of the file and, thanks to the doEndCurve
00211     // call-back structure, the building of the whole BomRoot BOM
00212     // (i.e., including Inventory, FlightDate, LegDate, SegmentDate, etc.)
00213     bsc::parse_info<iterator_t> info = bsc::parse (_startIterator, _endIterator
    ,
00214                                                    lFRAT5Parser,
00215                                                    bsc::space_p - bsc::eol_p);
00216
00217     // Retrieves whether or not the parsing was successful
00218     oResult = info.hit;
00219
00220     const bool isFull = info.full;
00221
00222     const std::string hasBeenFullyReadStr = (isFull  == true)?"":"not ";
00223     if (oResult == true && isFull == true) {
00224       STDAIR_LOG_DEBUG ("Parsing of FRAT5 input file: " << _filename
00225                         << " succeeded: read " << info.length
00226                         << " characters. The input file has "
00227                         << hasBeenFullyReadStr
00228                         << "been fully read. Stop point: " << info.stop);
00229
00230     } else {
00231       STDAIR_LOG_ERROR ("Parsing of FRAT5 input file: " << _filename
00232                         << " failed: read " << info.length
00233                         << " characters. The input file has "
00234                         << hasBeenFullyReadStr
00235                         << "been fully read. Stop point: " << info.stop);
00236       throw FRAT5FileParsingFailedException ("
    Parsing of FRAT5 input file: "
00237                                              + _filename + " failed.");
00238     }
00239
00240     return oResult;
00241   }
00242
00243 }
```

23.131 airinv/command/FRAT5ParserHelper.hpp File Reference

```
#include <string>
#include <stdair/command/CmdAbstract.hpp>
#include <airinv/AIRINV_Types.hpp>
#include <airinv/basic/BasParserTypes.hpp>
#include <airinv/bom/FRAT5Struct.hpp>
```

Classes

- struct AIRINV::FRAT5ParserHelper::ParserSemanticAction
- struct AIRINV::FRAT5ParserHelper::storeCurveKey
- struct AIRINV::FRAT5ParserHelper::storeDTD
- struct AIRINV::FRAT5ParserHelper::storeFRAT5Value
- struct AIRINV::FRAT5ParserHelper::doEndCurve
- struct AIRINV::FRAT5ParserHelper::FRAT5Parser
- struct AIRINV::FRAT5ParserHelper::FRAT5Parser::definition< ScannerT >
- class AIRINV::FRAT5FileParser

Namespaces

- namespace stdair

  Forward declarations.
- namespace AIRINV
- namespace AIRINV::FRAT5ParserHelper

23.132 FRAT5ParserHelper.hpp

```
00001 #ifndef __AIRINV_CMD_FRAT5PARSERHELPER_HPP
00002 #define __AIRINV_CMD_FRAT5PARSERHELPER_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/command/CmdAbstract.hpp>
00011 // Airinv
00012 #include <airinv/AIRINV_Types.hpp>
00013 #include <airinv/basic/BasParserTypes.hpp>
00014 #include <airinv/bom/FRAT5Struct.hpp>
00015
00016 // Forward declarations
00017 namespace stdair {
00018   class BomRoot;
00019 }
00020
00021 namespace AIRINV {
00022
00023   namespace FRAT5ParserHelper {
00024
00025     // //////////////////////////////////////////////////////////////////
00026     //  Semantic actions
00027     // //////////////////////////////////////////////////////////////////
00029     struct ParserSemanticAction {
00031       ParserSemanticAction (FRAT5Struct&);
00033       FRAT5Struct& _frat5;
00034     };
00035
00037     struct storeCurveKey : public ParserSemanticAction
    {
00039       storeCurveKey (FRAT5Struct&);
00041       void operator() (iterator_t iStr, iterator_t
    iStrEnd) const;
00042     };
00043
```

```
00045     struct storeDTD : public ParserSemanticAction {
00047       storeDTD (FRAT5Struct&);
00049       void operator() (int iDTD) const;
00050     };
00051
00053     struct storeFRAT5Value : public ParserSemanticAction
      {
00055       storeFRAT5Value (FRAT5Struct&);
00057       void operator() (double iReal) const;
00058     };
00059
00061     struct doEndCurve : public ParserSemanticAction
      {
00063       doEndCurve (stdair::BomRoot&, FRAT5Struct&);
00065       void operator() (iterator_t iStr, iterator_t
      iStrEnd) const;
00067       stdair::BomRoot& _bomRoot;
00068     };
00069
00071     //
00072     //  (Boost Spirit) Grammar Definition
00073     //
00075
00089     struct FRAT5Parser :
00090       public boost::spirit::classic::grammar<FRAT5Parser> {
00091
00092       FRAT5Parser (stdair::BomRoot&, FRAT5Struct&);
00093
00094       template <typename ScannerT>
00095       struct definition {
00096         definition (FRAT5Parser const& self);
00097
00098         // Instantiation of rules
00099         boost::spirit::classic::rule<ScannerT> curve_list,
00100           not_to_be_parsed, curve, key, map,
      value_pair, curve_end;
00101
00103         boost::spirit::classic::rule<ScannerT> const& start() const;
00104       };
00105
00106       // Parser Context
00107       stdair::BomRoot& _bomRoot;
00108       FRAT5Struct& _frat5;
00109     };
00110   }
00111
00116
00117   //
00118   //  Entry class for the file parser
00119   //
00121
00126   class FRAT5FileParser : public stdair::CmdAbstract {
00127   public:
00129     FRAT5FileParser (stdair::BomRoot& ioBomRoot,
00130                      const stdair::Filename_T& iFilename);
00131
00133     bool generateFRAT5Curves ();
00134
00135   private:
00137     void init();
00138
00139   private:
00140     // Attributes
00142     stdair::Filename_T _filename;
00143
00145     iterator_t _startIterator;
00146
00148     iterator_t _endIterator;
00149
00151     stdair::BomRoot& _bomRoot;
00152
00154     FRAT5Struct _frat5;
00155   };
00156
00157 }
00158 #endif // __AIRINV_CMD_FRAT5PARSERHELPER_HPP
```

23.133 airinv/command/InventoryBuilder.cpp File Reference

```
#include <cassert>
```

```
#include <boost/date_time/date_iterator.hpp>
#include <stdair/basic/BasConst_BookingClass.hpp>
#include <stdair/basic/BasConst_Yield.hpp>
#include <stdair/basic/BasConst_Inventory.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/SegmentDate.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/Bucket.hpp>
#include <stdair/bom/BomKeyManager.hpp>
#include <stdair/bom/ParsedKey.hpp>
#include <stdair/bom/BomRetriever.hpp>
#include <stdair/command/CmdCloneBomManager.hpp>
#include <stdair/factory/FacBom.hpp>
#include <stdair/factory/FacBomManager.hpp>
#include <stdair/service/Logger.hpp>
#include <airinv/AIRINV_Types.hpp>
#include <airinv/bom/FlightDateStruct.hpp>
#include <airinv/command/InventoryBuilder.hpp>
```

Namespaces

- namespace AIRINV

23.134 InventoryBuilder.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // Boost
00007 #include <boost/date_time/date_iterator.hpp>
00008 // StdAir
00009 #include <stdair/basic/BasConst_BookingClass.hpp>
00010 #include <stdair/basic/BasConst_Yield.hpp>
00011 #include <stdair/basic/BasConst_Inventory.hpp>
00012 #include <stdair/bom/BomManager.hpp>
00013 #include <stdair/bom/BomRoot.hpp>
00014 #include <stdair/bom/Inventory.hpp>
00015 #include <stdair/bom/AirlineFeature.hpp>
00016 #include <stdair/bom/FlightDate.hpp>
00017 #include <stdair/bom/SegmentDate.hpp>
00018 #include <stdair/bom/SegmentCabin.hpp>
00019 #include <stdair/bom/FareFamily.hpp>
00020 #include <stdair/bom/BookingClass.hpp>
00021 #include <stdair/bom/LegDate.hpp>
00022 #include <stdair/bom/LegCabin.hpp>
00023 #include <stdair/bom/Bucket.hpp>
00024 #include <stdair/bom/BomKeyManager.hpp>
00025 #include <stdair/bom/ParsedKey.hpp>
00026 #include <stdair/bom/BomRetriever.hpp>
00027 #include <stdair/command/CmdCloneBomManager.hpp>
00028 #include <stdair/factory/FacBom.hpp>
00029 #include <stdair/factory/FacBomManager.hpp>
00030 #include <stdair/service/Logger.hpp>
00031 // AirInv
00032 #include <airinv/AIRINV_Types.hpp>
00033 #include <airinv/bom/FlightDateStruct.hpp>
00034 #include <airinv/command/InventoryBuilder.hpp
```

```
    >
00035
00036 namespace AIRINV {
00037
00038   // ////////////////////////////////////////////////////////////////////
00039   void InventoryBuilder::
00040   buildInventory (stdair::BomRoot& ioBomRoot,
00041                   const FlightDateStruct& iFlightDateStruct) {
00042     const stdair::AirlineCode_T& lAirlineCode = iFlightDateStruct._airlineCode;
00043
00044     // Instantiate an inventory object (if not exist)
00045     // for the given key (airline code)
00046     stdair::Inventory* lInventory_ptr = stdair::BomManager::
00047       getObjectPtr<stdair::Inventory> (ioBomRoot, lAirlineCode);
00048     if (lInventory_ptr == NULL) {
00049       stdair::InventoryKey lKey (lAirlineCode);
00050       lInventory_ptr =
00051         &stdair::FacBom<stdair::Inventory>::instance().create (lKey);
00052       stdair::FacBomManager::addToListAndMap (ioBomRoot, *lInventory_ptr);
00053       stdair::FacBomManager::linkWithParent (ioBomRoot, *lInventory_ptr);
00054
00055       // Add the airline feature object to the inventory
00056       const stdair::AirlineFeatureKey lAirlineFeatureKey (lAirlineCode);
00057       stdair::AirlineFeature& lAirlineFeature =
00058         stdair::FacBom<stdair::AirlineFeature>::instance().create (
    lAirlineFeatureKey);
00059       stdair::FacBomManager::setAirlineFeature (*lInventory_ptr,
00060                                                 lAirlineFeature);
00061       stdair::FacBomManager::linkWithParent (*lInventory_ptr, lAirlineFeature);
00062       // Link the airline feature object with the top of the BOM tree
00063       stdair::FacBomManager::addToListAndMap (ioBomRoot, lAirlineFeature);
00064     }
00065     assert (lInventory_ptr != NULL);
00066
00067     // Build the flight-date within the inventory.
00068     buildFlightDate (*lInventory_ptr, iFlightDateStruct);
00069   }
00070
00071   // ////////////////////////////////////////////////////////////////////
00072   void InventoryBuilder::
00073   buildFlightDate (stdair::Inventory& ioInventory,
00074                    const FlightDateStruct& iFlightDateStruct) {
00075     // Create the FlightDateKey
00076     const stdair::FlightDateKey lFlightDateKey (iFlightDateStruct._flightNumber
    ,
00077                                                 iFlightDateStruct._flightDate);
00078
00079     // Check that the flight-date object is not already existing. If a
00080     // flight-date object with the same key has already been created,
00081     // then just update it, ifnot, create a flight-date and update it.
00082     stdair::FlightDate* lFlightDate_ptr = stdair::BomManager::
00083       getObjectPtr<stdair::FlightDate> (ioInventory, lFlightDateKey.toString())
    ;
00084     if (lFlightDate_ptr == NULL) {
00085       // Instantiate a fligh-date object for the given key (flight number and
00086       // flight date)
00087       lFlightDate_ptr =
00088         &stdair::FacBom<stdair::FlightDate>::instance().create (lFlightDateKey)
    ;
00089       stdair::FacBomManager::addToListAndMap (ioInventory, *lFlightDate_ptr);
00090       stdair::FacBomManager::linkWithParent (ioInventory, *lFlightDate_ptr);
00091     }
00092     assert (lFlightDate_ptr != NULL);
00093
00094     // Update the BOM flight-date with the attributes of the flight-date
     struct.
00095
00096     // Browse the list of leg-date struct and segment-date struct and
00097     // create the corresponding BOM.
00098     for (LegStructList_T::const_iterator itLegDate =
00099            iFlightDateStruct._legList.begin();
00100          itLegDate != iFlightDateStruct._legList.end(); ++itLegDate) {
00101       const LegStruct& lCurrentLegDateStruct = *itLegDate;
00102       buildLegDate (*lFlightDate_ptr, lCurrentLegDateStruct);
00103     }
00104
00105     for (SegmentStructList_T::const_iterator itSegmentDate =
00106            iFlightDateStruct._segmentList.begin();
00107          itSegmentDate != iFlightDateStruct._segmentList.end();
00108          ++itSegmentDate) {
00109       const SegmentStruct& lCurrentSegmentDateStruct = *itSegmentDate;
00110       buildSegmentDate (*lFlightDate_ptr, lCurrentSegmentDateStruct);
00111     }
00112
00113     buildRoutingLegKey (*lFlightDate_ptr);
00114   }
00115
```

```
00116   // ////////////////////////////////////////////////////////////////////
00117   void InventoryBuilder::
00118   buildLegDate (stdair::FlightDate& ioFlightDate,
00119                 const LegStruct& iLegDateStruct) {
00120     // Check that the leg-date object is not already existing. If a
00121     // leg-date object with the same key has already been created,
00122     // then just update it, ifnot, create a leg-date and update it.
00123     stdair::LegDate* lLegDate_ptr = stdair::BomManager::
00124       getObjectPtr<stdair::LegDate>(ioFlightDate, iLegDateStruct._boardingPoint
      );
00125
00126     if (lLegDate_ptr == NULL) {
00127       // Instantiate a leg-date object for the given key (boarding point);
00128       stdair::LegDateKey lKey (iLegDateStruct._boardingPoint);
00129       lLegDate_ptr = &stdair::FacBom<stdair::LegDate>::instance().create (lKey)
      ;
00130       stdair::FacBomManager::addToListAndMap (ioFlightDate, *lLegDate_ptr);
00131       stdair::FacBomManager::linkWithParent (ioFlightDate, *lLegDate_ptr);
00132     }
00133     assert (lLegDate_ptr != NULL);
00134
00135     // Update the BOM leg-date with the attributes of the leg-date struct.
00136     iLegDateStruct.fill (*lLegDate_ptr);
00137
00138     // Browse the list of leg-cabin structs and create the corresponding BOM.
00139     for (LegCabinStructList_T::const_iterator itLegCabin =
00140            iLegDateStruct._cabinList.begin();
00141          itLegCabin != iLegDateStruct._cabinList.end(); ++itLegCabin) {
00142       const LegCabinStruct& lCurrentLegCabinStruct = *itLegCabin;
00143       buildLegCabin (*lLegDate_ptr, lCurrentLegCabinStruct);
00144     }
00145   }
00146
00147   // ////////////////////////////////////////////////////////////////////
00148   void InventoryBuilder::
00149   buildRoutingLegKey (stdair::FlightDate& ioFlightDate) {
00150
00151     // Browse the list of segment-dates and create direct accesses
00152     // within each segment-date.
00153     const stdair::SegmentDateList_T& lSegmentDateList =
00154       stdair::BomManager::getList<stdair::SegmentDate> (ioFlightDate);
00155     for (stdair::SegmentDateList_T::const_iterator itSegmentDate =
00156            lSegmentDateList.begin();
00157          itSegmentDate != lSegmentDateList.end(); ++itSegmentDate) {
00158
00159       stdair::SegmentDate* lCurrentSegmentDate_ptr = *itSegmentDate;
00160       assert (lCurrentSegmentDate_ptr != NULL);
00161
00162       /*
00163        * If the segment is just marketed by this carrier,
00164        * retrieve the operating segment and call the createDirectAcces
00165        * method on its parent (flight date).
00166        */
00167       const stdair::SegmentDate* lOperatingSegmentDate_ptr =
00168         lCurrentSegmentDate_ptr->getOperatingSegmentDate ();
00169       if (lOperatingSegmentDate_ptr == NULL) {
00170
00171         const stdair::AirportCode_T& lBoardingPoint =
00172           lCurrentSegmentDate_ptr->getBoardingPoint();
00173
00174         stdair::AirportCode_T currentBoardingPoint = lBoardingPoint;
00175         const stdair::AirportCode_T& lOffPoint =
00176           lCurrentSegmentDate_ptr->getOffPoint();
00177
00178         // Add a sanity check so as to ensure that the loop stops. If
00179         // there are more than MAXIMAL_NUMBER_OF_LEGS legs, there is
00180         // an issue somewhere in the code (not in the parser, as the
00181         // segments are derived from the legs thanks to the
00182         // FlightPeriodStruct::buildSegments() method).
00183         unsigned short i = 1;
00184         while (currentBoardingPoint != lOffPoint
00185                && i <= stdair::MAXIMAL_NUMBER_OF_LEGS_IN_FLIGHT) {
00186           // Retrieve the (unique) LegDate getting that Boarding Point
00187           stdair::LegDate& lLegDate = stdair::BomManager::
00188             getObject<stdair::LegDate> (ioFlightDate, currentBoardingPoint);
00189
00190           // Link the SegmentDate and LegDate together
00191           const std::string& lRoutingKeyStr = lLegDate.describeRoutingKey();
00192           lCurrentSegmentDate_ptr->addLegKey(lRoutingKeyStr);
00193
00194           // Prepare the next iteration
00195           currentBoardingPoint = lLegDate.getOffPoint();
00196           ++i;
00197         }
00198         assert (i <= stdair::MAXIMAL_NUMBER_OF_LEGS_IN_FLIGHT);
00199       }
00200     }
```

```
00201   }
00202
00203   // ////////////////////////////////////////////////////////////////////
00204   void InventoryBuilder::
00205   buildLegCabin (stdair::LegDate& ioLegDate,
00206                  const LegCabinStruct& iLegCabinStruct) {
00207     // Check that the leg-cabin object is not already existing. If a
00208     // leg-cabin object with the same key has already been created,
00209     // then just update it, ifnot, create a leg-cabin and update it.
00210     stdair::LegCabin* lLegCabin_ptr = stdair::BomManager::
00211       getObjectPtr<stdair::LegCabin> (ioLegDate, iLegCabinStruct._cabinCode);
00212     if (lLegCabin_ptr == NULL) {
00213       // Instantiate a leg-cabin object for the given key (cabin code);
00214       stdair::LegCabinKey lKey (iLegCabinStruct._cabinCode);
00215       lLegCabin_ptr = &stdair::FacBom<stdair::LegCabin>::instance().create(lKey
      );
00216       stdair::FacBomManager::addToListAndMap (ioLegDate, *lLegCabin_ptr);
00217       stdair::FacBomManager::linkWithParent (ioLegDate, *lLegCabin_ptr);
00218     }
00219     assert (lLegCabin_ptr != NULL);
00220
00221     // TODO: Update the BOM leg-cabin with the attributes of the
00222     // leg-cabin struct.
00223     iLegCabinStruct.fill (*lLegCabin_ptr);
00224
00225     // Browse the list of bucket structs and create the corresponding BOM.
00226     for (BucketStructList_T::const_iterator itBucket =
00227            iLegCabinStruct._bucketList.begin();
00228          itBucket != iLegCabinStruct._bucketList.end(); ++itBucket) {
00229       const BucketStruct& lCurrentBucketStruct = *itBucket;
00230       buildBucket (*lLegCabin_ptr, lCurrentBucketStruct);
00231     }
00232   }
00233
00234   // ////////////////////////////////////////////////////////////////////
00235   void InventoryBuilder::buildBucket (stdair::LegCabin& ioLegCabin,
00236                                       const BucketStruct& iBucketStruct) {
00237     // Create the BucketKey
00238     const stdair::BucketKey lBucketKey (iBucketStruct._seatIndex);
00239
00240     // Check that the bucket object is not already existing. If a
00241     // bucket object with the same key has already been created,
00242     // then just update it, ifnot, create a bucket and update it.
00243     stdair::Bucket* lBucket_ptr = stdair::BomManager::
00244       getObjectPtr<stdair::Bucket> (ioLegCabin, lBucketKey.toString());
00245     if (lBucket_ptr == NULL) {
00246       // Instantiate a bucket object for the given key (seat index);
00247       stdair::BucketKey lKey (iBucketStruct._seatIndex);
00248       lBucket_ptr = &stdair::FacBom<stdair::Bucket>::instance().create (lKey);
00249       stdair::FacBomManager::addToListAndMap (ioLegCabin, *lBucket_ptr);
00250       stdair::FacBomManager::linkWithParent (ioLegCabin, *lBucket_ptr);
00251     }
00252     assert (lBucket_ptr != NULL);
00253
00254     //
00255     iBucketStruct.fill (*lBucket_ptr);
00256   }
00257
00258   // ////////////////////////////////////////////////////////////////////
00259   void InventoryBuilder::
00260   buildSegmentDate (stdair::FlightDate& ioFlightDate,
00261                     const SegmentStruct& iSegmentDateStruct) {
00262     // Check that the segment-date object is not already existing. If a
00263     // segment-date object with the same key has already been created,
00264     // then just update it, ifnot, create a segment-date and update it.
00265     const stdair::SegmentDateKey
00266       lSegmentDateKey (iSegmentDateStruct._boardingPoint,
00267                        iSegmentDateStruct._offPoint);
00268     stdair::SegmentDate* lSegmentDate_ptr = stdair::BomManager::
00269       getObjectPtr<stdair::SegmentDate>(ioFlightDate,lSegmentDateKey.toString()
      );
00270     if (lSegmentDate_ptr == NULL) {
00271       // Instantiate a segment-date object for the given key (boarding
00272       // and off points);
00273       lSegmentDate_ptr =  &stdair::FacBom<stdair::SegmentDate>::
00274         instance().create (lSegmentDateKey);
00275       stdair::FacBomManager::addToListAndMap (ioFlightDate, *lSegmentDate_ptr);
00276       stdair::FacBomManager::linkWithParent (ioFlightDate, *lSegmentDate_ptr);
00277     }
00278     assert (lSegmentDate_ptr != NULL);
00279
00280     // Update the BOM segment-date with the attributes of the
00281     // segment-date struct.
00282     iSegmentDateStruct.fill (*lSegmentDate_ptr);
00283
00284     // Browse the list of segment-cabin struct and create the corresponding
      BOM.
```

```
00285     for (SegmentCabinStructList_T::const_iterator itSegmentCabin =
00286            iSegmentDateStruct._cabinList.begin();
00287          itSegmentCabin != iSegmentDateStruct._cabinList.end();
00288          ++itSegmentCabin) {
00289       const SegmentCabinStruct& lCurrentSegmentCabinStruct = *itSegmentCabin;
00290       buildSegmentCabin (*lSegmentDate_ptr, lCurrentSegmentCabinStruct);
00291     }
00292   }
00293
00294   // ////////////////////////////////////////////////////////////////////
00295   void InventoryBuilder::
00296   buildSegmentCabin (stdair::SegmentDate& ioSegmentDate,
00297                      const SegmentCabinStruct& iSegmentCabinStruct) {
00298     // Check that the segment-cabin object is not already existing. If a
00299     // segment-cabin object with the same key has already been created,
00300     // then just update it, ifnot, create a segment-cabin and update it.
00301     stdair::SegmentCabin* lSegmentCabin_ptr = stdair::BomManager::
00302       getObjectPtr<stdair::SegmentCabin> (ioSegmentDate,
00303                                           iSegmentCabinStruct._cabinCode);
00304     if (lSegmentCabin_ptr == NULL) {
00305       // Instantiate a segment-cabin object for the given key (cabin code);
00306       stdair::SegmentCabinKey lKey (iSegmentCabinStruct._cabinCode);
00307       lSegmentCabin_ptr =
00308         &stdair::FacBom<stdair::SegmentCabin>::instance().create (lKey);
00309
00310       // Link the segment-cabin to the segment-date
00311       stdair::FacBomManager::addToListAndMap (ioSegmentDate, *lSegmentCabin_ptr
    );
00312       stdair::FacBomManager::linkWithParent (ioSegmentDate, *lSegmentCabin_ptr)
    ;
00313     }
00314     assert (lSegmentCabin_ptr != NULL);
00315
00316     // TODO: Update the BOM segment-cabin with the attributes of the
00317     // segment-cabin struct.
00318     iSegmentCabinStruct.fill (*lSegmentCabin_ptr);
00319
00320     // Browse the list of fare family struct and create the corresponding BOM.
00321     for (FareFamilyStructList_T::const_iterator itFareFamily =
00322            iSegmentCabinStruct._fareFamilies.begin();
00323          itFareFamily != iSegmentCabinStruct._fareFamilies.end();
00324          ++itFareFamily) {
00325       const FareFamilyStruct& lCurrentFareFamilyStruct = *itFareFamily;
00326       buildFareFamily (*lSegmentCabin_ptr, lCurrentFareFamilyStruct);
00327     }
00328   }
00329
00330   // ////////////////////////////////////////////////////////////////////
00331   void InventoryBuilder::
00332   buildFareFamily (stdair::SegmentCabin& ioSegmentCabin,
00333                    const FareFamilyStruct& iFareFamilyStruct) {
00334
00335     // Check that the fare family object is not already existing. If a
00336     // fare family object with the same key has already been created,
00337     // then just update it. If not, create a fare family and update it.
00338     stdair::FareFamily* lFareFamily_ptr = stdair::BomManager::
00339       getObjectPtr<stdair::FareFamily> (ioSegmentCabin,
00340                                         iFareFamilyStruct._familyCode);
00341     if (lFareFamily_ptr == NULL) {
00342       // Instantiate a fare family object for the given key (fare family code);
00343       const stdair::FareFamilyKey lFFKey (iFareFamilyStruct._familyCode);
00344       lFareFamily_ptr =
00345         &stdair::FacBom<stdair::FareFamily>::instance().create (lFFKey);
00346
00347       // Link the fare family to the segment-cabin
00348       stdair::FacBomManager::addToListAndMap (ioSegmentCabin, *lFareFamily_ptr)
    ;
00349       stdair::FacBomManager::linkWithParent (ioSegmentCabin, *lFareFamily_ptr);
00350     }
00351     assert (lFareFamily_ptr != NULL);
00352
00353     // TODO: Upcabin the BOM fare family with the attributes of the
00354     // fare family struct.
00355     iFareFamilyStruct.fill (*lFareFamily_ptr);
00356
00357     // Browse the list of booking-class struct and create the corresponding
     BOM.
00358     for (BookingClassStructList_T::const_iterator itBookingClass =
00359            iFareFamilyStruct._classList.begin();
00360          itBookingClass != iFareFamilyStruct._classList.end();
00361          ++itBookingClass) {
00362       const BookingClassStruct& lCurrentBookingClassStruct = *itBookingClass;
00363       buildBookingClass (*lFareFamily_ptr, lCurrentBookingClassStruct);
00364     }
00365   }
00366
00367   // ////////////////////////////////////////////////////////////////////
```

```
00368   void InventoryBuilder::
00369   buildBookingClass (stdair::FareFamily& ioFareFamily,
00370                      const BookingClassStruct& iBookingClassStruct) {
00371
00372     // Check that the booking class object is not already existing. If a
00373     // booking-class object with the same key has already been created,
00374     // then just update it. If not, create a booking-class and update it.
00375     stdair::BookingClass* lBookingClass_ptr = stdair::BomManager::
00376       getObjectPtr<stdair::BookingClass> (ioFareFamily,
00377                                           iBookingClassStruct._classCode);
00378     if (lBookingClass_ptr == NULL) {
00379       // Instantiate a booking class object for the given key (class code);
00380       const stdair::BookingClassKey lClassKey (iBookingClassStruct._classCode);
00381       lBookingClass_ptr =
00382         &stdair::FacBom<stdair::BookingClass>::instance().create (lClassKey);
00383
00384       // Link the booking-class to the fare family
00385       stdair::FacBomManager::addToListAndMap (ioFareFamily, *lBookingClass_ptr)
     ;
00386       stdair::FacBomManager::linkWithParent (ioFareFamily, *lBookingClass_ptr);
00387
00388       // Link the booking-class to the segment-cabin
00389       stdair::SegmentCabin& lSegmentCabin =
00390         stdair::BomManager::getParent<stdair::SegmentCabin> (ioFareFamily);
00391       stdair::FacBomManager::addToListAndMap (lSegmentCabin, *lBookingClass_ptr
     );
00392
00393       // Link the booking-class to the segment-date
00394       stdair::SegmentDate& lSegmentDate =
00395         stdair::BomManager::getParent<stdair::SegmentDate> (lSegmentCabin);
00396       stdair::FacBomManager::addToListAndMap (lSegmentDate, *lBookingClass_ptr)
     ;
00397     }
00398     assert (lBookingClass_ptr != NULL);
00399
00400     // TODO: Upcabin the BOM booking-class with the attributes of the
00401     // booking-class struct.
00402     iBookingClassStruct.fill (*lBookingClass_ptr);
00403   }
00404
00405   // ////////////////////////////////////////////////////////////////////
00406   void InventoryBuilder::buildPartnerInventories (stdair::BomRoot& ioBomRoot) {
00407
00408     // Browse the list of inventories to build partner inventories
00409     // within each inventory.
00410     const stdair::InventoryList_T& lInvList =
00411       stdair::BomManager::getList<stdair::Inventory> (ioBomRoot);
00412     for (stdair::InventoryList_T::const_iterator itInv = lInvList.begin();
00413          itInv != lInvList.end(); ++itInv) {
00414       stdair::Inventory* lCurrentInv_ptr = *itInv;
00415       assert (lCurrentInv_ptr != NULL);
00416
00417       buildPartnerInventories (ioBomRoot, *lCurrentInv_ptr);
00418     }
00419   }
00420
00421   // ////////////////////////////////////////////////////////////////////
00422   void InventoryBuilder::buildPartnerInventories (stdair::BomRoot& ioBomRoot,
00423                                                   stdair::Inventory&
     ioInventory) {
00424
00425     // Browse the list of flight-dates to build partner inventories.
00426     const stdair::FlightDateList_T& lFlightDateList =
00427       stdair::BomManager::getList<stdair::FlightDate> (ioInventory);
00428     for (stdair::FlightDateList_T::const_iterator itFlightDate =
00429            lFlightDateList.begin();
00430          itFlightDate != lFlightDateList.end(); ++itFlightDate) {
00431       stdair::FlightDate* lCurrentFlightDate_ptr = *itFlightDate;
00432       assert (lCurrentFlightDate_ptr != NULL);
00433
00434       buildPartnerInventories (ioBomRoot, ioInventory, *lCurrentFlightDate_ptr)
     ;
00435     }
00436
00437   }
00438
00439   // ////////////////////////////////////////////////////////////////////
00440   void InventoryBuilder::buildPartnerInventories (stdair::BomRoot& ioBomRoot,
00441                                                   stdair::Inventory&
     ioInventory,
00442                                                   stdair::FlightDate&
     iFlightDate) {
00443
00444     // Browse the list of flight-dates to build partner inventories.
00445     const stdair::SegmentDateList_T& lSegmentDateList =
00446       stdair::BomManager::getList<stdair::SegmentDate> (iFlightDate);
00447     for (stdair::SegmentDateList_T::const_iterator itSegmentDate =
```

```
00448           lSegmentDateList.begin();
00449         itSegmentDate != lSegmentDateList.end(); ++itSegmentDate) {
00450
00451       stdair::SegmentDate* lCurrentSegmentDate_ptr = *itSegmentDate;
00452       assert (lCurrentSegmentDate_ptr != NULL);
00453
00454       const stdair::RoutingLegKeyList_T& lRoutingLegKeyList =
00455         lCurrentSegmentDate_ptr->getLegKeyList ();
00456
00457       // Browse the list of routing leg keys.
00458       for (stdair::RoutingLegKeyList_T::const_iterator itRoutingLegKey =
00459            lRoutingLegKeyList.begin();
00460          itRoutingLegKey != lRoutingLegKeyList.end(); ++itRoutingLegKey) {
00461
00462         // Extract the operating airline code
00463         const stdair::ParsedKey& lParsedKey =
00464           stdair::BomKeyManager::extractKeys (*itRoutingLegKey);
00465         const stdair::InventoryKey& lInventoryKey =
00466           lParsedKey.getInventoryKey();
00467         const stdair::AirlineCode_T lOperatingAirlineCode =
00468           lInventoryKey.getAirlineCode();
00469
00470         // Extract the current airline code
00471         const stdair::AirlineCode_T lAirlineCode =
00472           iFlightDate.getAirlineCode();
00473
00474         // If the operating airline is not the current one
00475         if (lOperatingAirlineCode != lAirlineCode) {
00476
00477           // Look for the inventory of the partner within the Bom root
00478           stdair::Inventory* lInventory_ptr =
00479             stdair::BomRetriever::
00480             retrieveInventoryFromKey (ioBomRoot, lOperatingAirlineCode);
00481
00482           // Build the current segment full key
00483           std::ostringstream lRoutingSegment;
00484           lRoutingSegment << iFlightDate.getAirlineCode() << ";"
00485                           << iFlightDate.describeKey() << ";"
00486                           << lCurrentSegmentDate_ptr->getBoardingPoint() << ";"
00487                           << lCurrentSegmentDate_ptr->getOffPoint();
00488
00489           // If such inventory does not exist, throw an exception
00490           if (lInventory_ptr == NULL) {
00491             std::ostringstream oMessage;
00492             oMessage << "The input file does not contain information about "
00493                      << "the '" << *itRoutingLegKey << "' leg date needed by "
00494                      << "the '" << lRoutingSegment.str() << "' segment.";
00495             STDAIR_LOG_ERROR (oMessage.str());
00496             throw MissingPartnerFlightDateWithinScheduleFile (oMessage.str());
00497           }
00498           assert (lInventory_ptr != NULL);
00499
00500           // Build the partner inventory within the current inventory
00501           buildInventory (ioBomRoot, ioInventory, *itRoutingLegKey);
00502
00503           // Build the current inventory as a partner of the partner inventory
00504           buildInventory (ioBomRoot, *lInventory_ptr, lRoutingSegment.str());
00505         }
00506       }
00507     }
00508   }
00509
00510   // ////////////////////////////////////////////////////////////////////
00511   void InventoryBuilder::
00512   buildInventory (stdair::BomRoot& ioBomRoot,
00513                   stdair::Inventory& ioInventory,
00514                   const std::string& iFullKeyStr) {
00515
00516     // Check that the inventory object is not already existing. If an
00517     // inventory object with the same key has already been created,
00518     // then just update it, ifnot, create an inventory and update it.
00519     // for the given key (iFullKeyStr)
00520     stdair::Inventory* lInventory_ptr =
00521       stdair::BomRetriever::retrieveInventoryFromLongKey (ioInventory,
00522                                                           iFullKeyStr);
00523     if (lInventory_ptr == NULL) {
00524       // Instantiate a fligh-date object for the given key (airline code
00525       // within the iFullKeyStr)
00526       stdair::Inventory* lOperatingInventory_ptr =
00527         stdair::BomRetriever::retrieveInventoryFromLongKey (ioBomRoot,
00528                                                             iFullKeyStr);
00529       assert (lOperatingInventory_ptr != NULL);
00530       lInventory_ptr =
00531         &stdair::FacBom<stdair::Inventory>::instance().create (*
      lOperatingInventory_ptr);
00532       stdair::FacBomManager::addToListAndMap (ioInventory, *lInventory_ptr);
00533       stdair::FacBomManager::linkWithParent (ioInventory, *lInventory_ptr);
```

```
00534     }
00535     assert (lInventory_ptr != NULL);
00536
00537     // Build the flight-date within the inventory.
00538     buildFlightDate (ioBomRoot, *lInventory_ptr, iFullKeyStr);
00539   }
00540
00541   // ////////////////////////////////////////////////////////////////////
00542   void InventoryBuilder::
00543   buildFlightDate (stdair::BomRoot& ioBomRoot,
00544                    stdair::Inventory& ioInventory,
00545                    const std::string& iFullKeyStr) {
00546
00547     // Check that the flight-date object is not already existing. If a
00548     // flight-date object with the same key has already been created,
00549     // then just update it, ifnot, create a flight-date and update it.
00550     stdair::FlightDate* lFlightDate_ptr  =
00551       stdair::BomRetriever::retrieveFlightDateFromLongKey (ioInventory,
00552                                                            iFullKeyStr);
00553     if (lFlightDate_ptr == NULL) {
00554       // Instantiate a flighy-date object for the given key (flight number and
00555       // flight date within the iFullKeyStr)
00556       stdair::FlightDate* lOperatingFlightDate_ptr =
00557         stdair::BomRetriever::retrieveFlightDateFromLongKey (ioBomRoot,
00558                                                              iFullKeyStr);
00559       assert (lOperatingFlightDate_ptr != NULL);
00560       stdair::FlightDate& lFlightDate =
00561         cloneFlightDate (*lOperatingFlightDate_ptr);
00562       stdair::FacBomManager::addToListAndMap (ioInventory, lFlightDate);
00563       stdair::FacBomManager::linkWithParent (ioInventory, lFlightDate);
00564     }
00565   }
00566
00567   // ////////////////////////////////////////////////////////////////////
00568   stdair::FlightDate& InventoryBuilder::
00569   cloneFlightDate (const stdair::FlightDate& iFlightDate) {
00570
00574     stdair::FlightDate& lCloneFlightDate =
00575       stdair::FacBom<stdair::FlightDate>::instance().create (iFlightDate);
00576
00577     // Check whether there are LegDate objects
00578     const bool hasLegDateList = stdair::BomManager::hasList<stdair::LegDate> (
      iFlightDate);
00579     if (hasLegDateList == true) {
00580
00581       // Browse the leg-dates
00582       const stdair::LegDateList_T& lLegDateList =
00583         stdair::BomManager::getList<stdair::LegDate> (iFlightDate);
00584       for (stdair::LegDateList_T::const_iterator itLD = lLegDateList.begin();
00585            itLD != lLegDateList.end(); ++itLD) {
00586         const stdair::LegDate* lLD_ptr = *itLD;
00587         assert (lLD_ptr != NULL);
00588
00589         // Clone the current leg-date
00590         stdair::LegDate& lCloneLegDate = cloneLegDate (*lLD_ptr);
00591         stdair::FacBomManager::addToListAndMap (lCloneFlightDate, lCloneLegDate
      );
00592         stdair::FacBomManager::linkWithParent (lCloneFlightDate, lCloneLegDate)
      ;
00593       }
00594     }
00595
00596     // Check whether there are SegmentDate objects
00597     const bool hasSegmentDateList =
00598       stdair::BomManager::hasList<stdair::SegmentDate> (iFlightDate);
00599     if (hasSegmentDateList == true) {
00600
00601       // Browse the segment-dates
00602       const stdair::SegmentDateList_T& lSegmentDateList =
00603         stdair::BomManager::getList<stdair::SegmentDate> (iFlightDate);
00604       for (stdair::SegmentDateList_T::const_iterator itSD = lSegmentDateList.
      begin();
00605            itSD != lSegmentDateList.end(); ++itSD) {
00606         const stdair::SegmentDate* lSD_ptr = *itSD;
00607         assert (lSD_ptr != NULL);
00608
00609         // Clone the current segment-date
00610         stdair::SegmentDate& lCloneSegmentDate = cloneSegmentDate (*lSD_ptr);
00611         stdair::FacBomManager::addToListAndMap (lCloneFlightDate,
      lCloneSegmentDate);
00612         stdair::FacBomManager::linkWithParent (lCloneFlightDate,
      lCloneSegmentDate);
00613
00614       }
00615     }
00616
00617     return lCloneFlightDate;
```

```
00618   }
00619
00620   // ////////////////////////////////////////////////////////////////////
00621   stdair::LegDate& InventoryBuilder::cloneLegDate (const stdair::LegDate&
      iLegDate) {
00622
00626     stdair::LegDate& lCloneLegDate =
00627       stdair::FacBom<stdair::LegDate>::instance().create (iLegDate);
00628
00629     // Check whether there are LegCabin objects
00630     const bool hasLegCabinList = stdair::BomManager::hasList<stdair::LegCabin>
      (iLegDate);
00631     if (hasLegCabinList == true) {
00632       // Browse the leg-cabins
00633       const stdair::LegCabinList_T& lLegCabinList =
00634         stdair::BomManager::getList<stdair::LegCabin> (iLegDate);
00635       for (stdair::LegCabinList_T::const_iterator itLC = lLegCabinList.begin();
00636            itLC != lLegCabinList.end(); ++itLC) {
00637         const stdair::LegCabin* lLC_ptr = *itLC;
00638         assert (lLC_ptr != NULL);
00639
00640         // Clone the current leg-cabin
00641         stdair::LegCabin& lCloneLegCabin = cloneLegCabin (*lLC_ptr);
00642         stdair::FacBomManager::addToListAndMap (lCloneLegDate, lCloneLegCabin);
00643         stdair::FacBomManager::linkWithParent (lCloneLegDate, lCloneLegCabin);
00644       }
00645     }
00646
00647     return lCloneLegDate;
00648   }
00649
00650   // ////////////////////////////////////////////////////////////////////
00651   stdair::LegCabin& InventoryBuilder::cloneLegCabin (const stdair::LegCabin&
      iLegCabin) {
00652
00656     stdair::LegCabin& lCloneLegCabin =
00657       stdair::FacBom<stdair::LegCabin>::instance().create (iLegCabin);
00658
00659     // Check whether there are Bucket objects
00660     const bool hasBucketList = stdair::BomManager::hasList<stdair::Bucket> (
      iLegCabin);
00661     if (hasBucketList == true) {
00662       // Browse the buckets
00663       const stdair::BucketList_T& lBucketList =
00664         stdair::BomManager::getList<stdair::Bucket> (iLegCabin);
00665       for (stdair::BucketList_T::const_iterator itBucket = lBucketList.begin();
00666            itBucket != lBucketList.end(); ++itBucket) {
00667         const stdair::Bucket* lBucket_ptr = *itBucket;
00668         assert (lBucket_ptr != NULL);
00669
00670         // Clone the current bucket
00671         stdair::Bucket& lCloneBucket = cloneBucket (*lBucket_ptr);
00672         stdair::FacBomManager::addToListAndMap (lCloneLegCabin, lCloneBucket);
00673         stdair::FacBomManager::linkWithParent (lCloneLegCabin, lCloneBucket);
00674       }
00675     }
00676
00677     return lCloneLegCabin;
00678   }
00679
00680   // ////////////////////////////////////////////////////////////////////
00681   stdair::Bucket& InventoryBuilder::cloneBucket (const stdair::Bucket& iBucket)
       {
00682
00686     stdair::Bucket& lCloneBucket =
00687       stdair::FacBom<stdair::Bucket>::instance().create (iBucket);
00688
00689     return lCloneBucket;
00690   }
00691
00692   // ////////////////////////////////////////////////////////////////////
00693   stdair::SegmentDate& InventoryBuilder::
00694   cloneSegmentDate (const stdair::SegmentDate& iSegmentDate) {
00695
00699     stdair::SegmentDate& lCloneSegmentDate =
00700       stdair::FacBom<stdair::SegmentDate>::instance().create (iSegmentDate);
00701
00702     // Check whether there are SegmentCabin objects
00703     const bool hasSegmentCabinList =
00704       stdair::BomManager::hasList<stdair::SegmentCabin> (iSegmentDate);
00705     if (hasSegmentCabinList == true) {
00706       // Browse the segment-cabins
00707       const stdair::SegmentCabinList_T& lSegmentCabinList =
00708         stdair::BomManager::getList<stdair::SegmentCabin> (iSegmentDate);
00709       for (stdair::SegmentCabinList_T::const_iterator itSC = lSegmentCabinList.
      begin();
00710            itSC != lSegmentCabinList.end(); ++itSC) {
```

```
00711         const stdair::SegmentCabin* lSC_ptr = *itSC;
00712         assert (lSC_ptr != NULL);
00713
00714         // Clone the current segment-cabin
00715         stdair::SegmentCabin& lCloneSegmentCabin = cloneSegmentCabin (*lSC_ptr)
      ;
00716         stdair::FacBomManager::addToListAndMap (lCloneSegmentDate,
      lCloneSegmentCabin);
00717         stdair::FacBomManager::linkWithParent (lCloneSegmentDate,
      lCloneSegmentCabin);
00718
00719         linkBookingClassesWithSegment (lCloneSegmentDate,
00720                                        lCloneSegmentCabin);
00721
00722       }
00723     }
00724     return lCloneSegmentDate;
00725   }
00726
00727   // ////////////////////////////////////////////////////////////////////
00728   void InventoryBuilder::
00729   linkBookingClassesWithSegment (stdair::SegmentDate& iCloneSegmentDate,
00730                                  stdair::SegmentCabin& iCloneSegmentCabin) {
00731
00732     // Browse the fare families to link the booking-classes to the
00733     // segment-cabin and to the segment-date
00734     const bool hasFareFamilyList =
00735       stdair::BomManager::hasList<stdair::FareFamily> (iCloneSegmentCabin);
00736     if (hasFareFamilyList == true) {
00737       const stdair::FareFamilyList_T& lCloneFFList =
00738         stdair::BomManager::getList<stdair::FareFamily> (iCloneSegmentCabin);
00739       for (stdair::FareFamilyList_T::const_iterator itCloneFF = lCloneFFList.
      begin();
00740            itCloneFF != lCloneFFList.end(); ++itCloneFF) {
00741         const stdair::FareFamily* lCloneFF_ptr = *itCloneFF;
00742         assert (lCloneFF_ptr != NULL);
00743
00744         // Browse the list of booking classes
00745         const bool hasBookingClasslist =
00746           stdair::BomManager::hasList<stdair::BookingClass> (*lCloneFF_ptr);
00747         if (hasBookingClasslist == true) {
00748           const stdair::BookingClassList_T& lCloneBCList =
00749             stdair::BomManager::getList<stdair::BookingClass> (*lCloneFF_ptr);
00750           for (stdair::BookingClassList_T::const_iterator itCloneBC =
00751                  lCloneBCList.begin();
00752                itCloneBC != lCloneBCList.end(); ++itCloneBC) {
00753             const stdair::BookingClass* lCloneBC_ptr = *itCloneBC;
00754             assert (lCloneBC_ptr != NULL);
00755
00756             // Link the booking-class to the segment-cabin
00757             stdair::FacBomManager::addToListAndMap (iCloneSegmentCabin,
00758                                                     *lCloneBC_ptr);
00759
00760             // Link the booking-class to the segment-date
00761             stdair::FacBomManager::addToListAndMap (iCloneSegmentDate,
00762                                                     *lCloneBC_ptr);
00763           }
00764         }
00765       }
00766     }
00767   }
00768
00769   // ////////////////////////////////////////////////////////////////////
00770   stdair::SegmentCabin& InventoryBuilder::
00771   cloneSegmentCabin (const stdair::SegmentCabin& iSegmentCabin) {
00772
00776     stdair::SegmentCabin& lCloneSegmentCabin =
00777       stdair::FacBom<stdair::SegmentCabin>::instance().create (iSegmentCabin);
00778
00779     // Check whether there are fare family objects
00780     const bool hasFareFamilyList =
00781       stdair::BomManager::hasList<stdair::FareFamily> (iSegmentCabin);
00782     if (hasFareFamilyList == true) {
00783       // Browse the fare families
00784       const stdair::FareFamilyList_T& lFareFamilyList =
00785        stdair::BomManager::getList<stdair::FareFamily> (iSegmentCabin);
00786       for (stdair::FareFamilyList_T::const_iterator itFF = lFareFamilyList.
      begin();
00787            itFF != lFareFamilyList.end(); ++itFF) {
00788         const stdair::FareFamily* lFF_ptr = *itFF;
00789         assert (lFF_ptr != NULL);
00790
00791         // Clone the current fare-family
00792         stdair::FareFamily& lCloneFareFamily = cloneFareFamily (*lFF_ptr);
00793         stdair::FacBomManager::addToListAndMap (lCloneSegmentCabin,
      lCloneFareFamily);
00794         stdair::FacBomManager::linkWithParent (lCloneSegmentCabin,
```

```
      lCloneFareFamily);
00795       }
00796     }
00797
00798     return lCloneSegmentCabin;
00799   }
00800
00801   // ////////////////////////////////////////////////////////////////////
00802   stdair::FareFamily& InventoryBuilder::
00803   cloneFareFamily (const stdair::FareFamily& iFareFamily) {
00807     stdair::FareFamily& lCloneFareFamily =
00808       stdair::FacBom<stdair::FareFamily>::instance().create (iFareFamily);
00809
00810     // Check whether there are booking classes objects
00811     const bool hasBookingClassList =
00812       stdair::BomManager::hasList<stdair::BookingClass> (iFareFamily);
00813     if (hasBookingClassList == true) {
00814       // Browse the list of booking classes
00815       const stdair::BookingClassList_T& lBookingClassList =
00816         stdair::BomManager::getList<stdair::BookingClass> (iFareFamily);
00817       for (stdair::BookingClassList_T::const_iterator itBookingClass =
00818              lBookingClassList.begin();
00819            itBookingClass != lBookingClassList.end(); ++itBookingClass) {
00820         const stdair::BookingClass* lBC_ptr = *itBookingClass;
00821         assert (lBC_ptr != NULL);
00822
00823         // Clone the current booking class
00824         stdair::BookingClass& lCloneBookingClass = cloneBookingClass (*lBC_ptr)
      ;
00825         stdair::FacBomManager::addToListAndMap (lCloneFareFamily,
      lCloneBookingClass);
00826         stdair::FacBomManager::linkWithParent (lCloneFareFamily,
      lCloneBookingClass);
00827       }
00828     }
00829
00830     return lCloneFareFamily;
00831   }
00832
00833   // ////////////////////////////////////////////////////////////////////
00834   stdair::BookingClass& InventoryBuilder::
00835   cloneBookingClass (const stdair::BookingClass& iBookingClass) {
00836
00840     stdair::BookingClass& lCloneBookingClass =
00841        stdair::FacBom<stdair::BookingClass>::instance().create (iBookingClass);
00842
00843     return lCloneBookingClass;
00844   }
00845 }
00846
```

23.135 airinv/command/InventoryBuilder.hpp File Reference

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

Classes

- class AIRINV::InventoryBuilder

  Class handling the generation / instantiation of the Inventory BOM.

Namespaces

- namespace stdair

  Forward declarations.
- namespace AIRINV
- namespace AIRINV::InventoryParserHelper

23.136 InventoryBuilder.hpp

```
00001 #ifndef __AIRINV_CMD_INVENTORYBUILDER_HPP
```

```
00002 #define __AIRINV_CMD_INVENTORYBUILDER_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // StdAir
00008 #include <stdair/command/CmdAbstract.hpp>
00009 // AirInv
00010 #include <airinv/AIRINV_Types.hpp>
00011
00013 namespace stdair {
00014   class BomRoot;
00015   class Inventory;
00016   class FlightDate;
00017   class LegDate;
00018   class LegCabin;
00019   class Bucket;
00020   class SegmentDate;
00021   class SegmentCabin;
00022   class FareFamily;
00023   struct ParsedKey;
00024   class BookingClass;
00025 }
00026
00027 namespace AIRINV {
00028
00030   struct FlightDateStruct;
00031   struct LegStruct;
00032   struct LegCabinStruct;
00033   struct BucketStruct;
00034   struct SegmentStruct;
00035   struct SegmentCabinStruct;
00036   struct FareFamilyStruct;
00037   struct BookingClassStruct;
00038   namespace InventoryParserHelper {
00039     struct doEndFlightDate;
00040   }
00041
00045   class InventoryBuilder : public stdair::CmdAbstract {
00051     friend class AIRINV_Service;
00052     friend struct InventoryParserHelper::doEndFlightDate
      ;
00053
00054   private:
00059     static void buildInventory (stdair::BomRoot&, const FlightDateStruct
      &);
00060
00065     static void buildFlightDate (stdair::Inventory&, const FlightDateStruct
      &);
00066
00071     static void buildLegDate (stdair::FlightDate&, const LegStruct&);
00072
00076     static void buildRoutingLegKey (stdair::FlightDate&);
00077
00082     static void buildLegCabin (stdair::LegDate&, const LegCabinStruct
      &);
00083
00088     static void buildBucket (stdair::LegCabin&, const BucketStruct&
      );
00089
00094     static void buildSegmentDate (stdair::FlightDate&, const SegmentStruct
      &);
00095
00100     static void buildSegmentCabin (stdair::SegmentDate&,
00101                                    const SegmentCabinStruct&)
      ;
00102
00107     static void buildFareFamily (stdair::SegmentCabin&,
00108                                  const FareFamilyStruct&);
00109
00114     static void buildBookingClass (stdair::FareFamily&,
00115                                    const BookingClassStruct&)
      ;
00116
00120     static void buildPartnerInventories (stdair::BomRoot&);
00121
00125     static void buildPartnerInventories (stdair::BomRoot&,
00126                                          stdair::Inventory&);
00127
00131     static void buildPartnerInventories (stdair::BomRoot&,
00132                                          stdair::Inventory&,
00133                                          stdair::FlightDate&);
00134
00135
00136     static void buildInventory (stdair::BomRoot&,
00137                                 stdair::Inventory&,
00138                                 const std::string& iFullKeyStr);
```

```
00139
00140     static void buildFlightDate (stdair::BomRoot&,
00141                                  stdair::Inventory&,
00142                                  const std::string& iFullKeyStr);
00143
00151     static stdair::FlightDate& cloneFlightDate (const stdair::FlightDate&);
00152
00160     static stdair::LegDate& cloneLegDate (const stdair::LegDate&);
00161
00169     static stdair::LegCabin& cloneLegCabin (const stdair::LegCabin&);
00170
00178     static stdair::Bucket& cloneBucket (const stdair::Bucket&);
00179
00187     static stdair::SegmentDate& cloneSegmentDate (const stdair::SegmentDate&);
00188
00196     static void linkBookingClassesWithSegment (stdair::SegmentDate&,
00197                                                stdair::SegmentCabin&);
00198
00206     static stdair::SegmentCabin& cloneSegmentCabin (const stdair::SegmentCabin&
      );
00207
00215     static stdair::FareFamily& cloneFareFamily (const stdair::FareFamily&);
00216
00224     static stdair::BookingClass& cloneBookingClass (const stdair::BookingClass&
      );
00225
00226   };
00227
00228 }
00229 #endif // __AIRINV_CMD_INVENTORYBUILDER_HPP
```

23.137 airinv/command/InventoryGenerator.cpp File Reference

```
#include <cassert>
#include <boost/date_time/date_iterator.hpp>
#include <stdair/stdair_types.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_SellUpCurves.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/SegmentDate.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/SimpleNestingStructure.hpp>
#include <stdair/bom/NestingNode.hpp>
#include <stdair/bom/Policy.hpp>
#include <stdair/bom/Bucket.hpp>
#include <stdair/bom/BomKeyManager.hpp>
#include <stdair/factory/FacBomManager.hpp>
#include <stdair/service/Logger.hpp>
#include <airinv/bom/FlightPeriodStruct.hpp>
#include <airinv/command/InventoryGenerator.hpp>
```

Namespaces

- namespace AIRINV

23.138 InventoryGenerator.cpp

```
00001 // //////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // Boost
00007 #include <boost/date_time/date_iterator.hpp>
00008 // StdAir
00009 #include <stdair/stdair_types.hpp>
00010 #include <stdair/basic/BasConst_Inventory.hpp>
00011 #include <stdair/basic/BasConst_SellUpCurves.hpp>
00012 #include <stdair/bom/BomManager.hpp>
00013 #include <stdair/bom/BomRoot.hpp>
00014 #include <stdair/bom/Inventory.hpp>
00015 #include <stdair/bom/AirlineFeature.hpp>
00016 #include <stdair/bom/FlightDate.hpp>
00017 #include <stdair/bom/SegmentDate.hpp>
00018 #include <stdair/bom/SegmentCabin.hpp>
00019 #include <stdair/bom/FareFamily.hpp>
00020 #include <stdair/bom/BookingClass.hpp>
00021 #include <stdair/bom/LegDate.hpp>
00022 #include <stdair/bom/LegCabin.hpp>
00023 #include <stdair/bom/SimpleNestingStructure.hpp>
00024 #include <stdair/bom/NestingNode.hpp>
00025 #include <stdair/bom/Policy.hpp>
00026 #include <stdair/bom/Bucket.hpp>
00027 #include <stdair/bom/BomKeyManager.hpp>
00028 #include <stdair/factory/FacBomManager.hpp>
00029 #include <stdair/service/Logger.hpp>
00030 // AirInv
00031 #include <airinv/bom/FlightPeriodStruct.hpp>
00032 #include <airinv/command/InventoryGenerator.hpp
      >
00033
00034 namespace AIRINV {
00035
00036   // //////////////////////////////////////////////////////////////////////
00037   void InventoryGenerator::
00038   createFlightDate (stdair::BomRoot& ioBomRoot,
00039                     const FlightPeriodStruct& iFlightPeriod) {
00040     const stdair::AirlineCode_T& lAirlineCode = iFlightPeriod._airlineCode;
00041
00042     // Instantiate an inventory object (if not exist)
00043     // for the given key (airline code)
00044     stdair::Inventory* lInventory_ptr = stdair::BomManager::
00045       getObjectPtr<stdair::Inventory> (ioBomRoot, lAirlineCode);
00046     if (lInventory_ptr == NULL) {
00047       stdair::InventoryKey lKey (lAirlineCode);
00048       lInventory_ptr =
00049         &stdair::FacBom<stdair::Inventory>::instance().create (lKey);
00050       stdair::FacBomManager::addToListAndMap (ioBomRoot, *lInventory_ptr);
00051       stdair::FacBomManager::linkWithParent (ioBomRoot, *lInventory_ptr);
00052
00053       // Add the airline feature object to the inventory
00054       const stdair::AirlineFeatureKey lAirlineFeatureKey (lAirlineCode);
00055       stdair::AirlineFeature& lAirlineFeature =
00056         stdair::FacBom<stdair::AirlineFeature>::instance().create (
      lAirlineFeatureKey);
00057       stdair::FacBomManager::setAirlineFeature (*lInventory_ptr,
00058                                                 lAirlineFeature);
00059       stdair::FacBomManager::linkWithParent (*lInventory_ptr, lAirlineFeature);
00060       // Link the airline feature object with the top of the BOM tree
00061       stdair::FacBomManager::addToListAndMap (ioBomRoot, lAirlineFeature);
00062     }
00063     assert (lInventory_ptr != NULL);
00064
00065     // Generate all the dates corresponding to the period
00066     // and create the corresponding flight-dates.
00067     const stdair::DatePeriod_T lDateRange = iFlightPeriod._dateRange;
00068
00069     for (boost::gregorian::day_iterator itDate = lDateRange.begin();
00070          itDate != lDateRange.end(); ++itDate) {
00071       const stdair::Date_T& currentDate = *itDate;
00072
00073       // Retrieve, for the current day, the Day-Of-the-Week (thanks to Boost)
00074       const unsigned short currentDoW = currentDate.day_of_week().as_number();
00075
00076       // The FlightPeriod structure stores which Days (-Of-the-Week) are
00077       // active within the week. For each day (Mon., Tue., etc.), a boolean
00078       // states whether the Flight is active for that day.
00079       const stdair::DoWStruct& lDoWList = iFlightPeriod._dow;
00080       const bool isDoWActive = lDoWList.getStandardDayOfWeek (currentDoW);
00081
00082       if (isDoWActive == true) {
00083         createFlightDate (ioBomRoot, *lInventory_ptr, currentDate,
```

```
00084                           iFlightPeriod);
00085       }
00086     }
00087   }
00088
00089   // ////////////////////////////////////////////////////////////////////
00090   void InventoryGenerator::
00091   createFlightDate (stdair::BomRoot& ioBomRoot,
00092                     stdair::Inventory& ioInventory,
00093                     const stdair::Date_T& iFlightDate,
00094                     const FlightPeriodStruct& iFlightPeriod) {
00095     // Create the FlightDateKey
00096     const stdair::FlightNumber_T& lFlightNumber = iFlightPeriod._flightNumber;
00097     stdair::FlightDateKey lFlightDateKey (lFlightNumber, iFlightDate);
00098
00099     // DEBUG
00100     // STDAIR_LOG_DEBUG ("Creating flight-date: " <<
      lFlightDateKey.toString());
00101
00102     // Check that the flight-date object is not already existing. If a
00103     // FlightDate object with the same key has already been created,
00104     // it means that the schedule input file is invalid (two flight-periods
00105     // are overlapping).
00106     stdair::FlightDate* lFlightDate_ptr = stdair::BomManager::
00107       getObjectPtr<stdair::FlightDate> (ioInventory, lFlightDateKey.toString())
    ;
00108     if (lFlightDate_ptr != NULL) {
00109       std::ostringstream oMessage;
00110       oMessage << ioInventory.describeKey() << ", "
00111                << lFlightDate_ptr->describeKey();
00112       throw FlightDateDuplicationException (oMessage.str());
00113     }
00114
00115     // Instantiate a fligh-date object with the given key (flight number and
00116     // flight date)
00117     lFlightDate_ptr =
00118       &stdair::FacBom<stdair::FlightDate>::instance().create (lFlightDateKey);
00119     stdair::FacBomManager::addToListAndMap (ioInventory, *lFlightDate_ptr);
00120     stdair::FacBomManager::linkWithParent (ioInventory, *lFlightDate_ptr);
00121
00122     // Iterate on the leg-dates
00123     stdair::Duration_T currentOffTime (0, 0, 0);
00124     stdair::AirportCode_T previousOffPoint;
00125     const LegStructList_T& lLegList = iFlightPeriod._legList;
00126     for (LegStructList_T::const_iterator itLeg = lLegList.begin();
00127          itLeg != lLegList.end(); ++itLeg) {
00128       const LegStruct& lLeg = *itLeg;
00129
00130       // Create the leg-branch of the flight-date BOM
00131       stdair::LegDate& lLegDate =
00132         createLegDate (*lFlightDate_ptr, iFlightDate, lLeg);
00133
00134       // TODO: Check that the boarding date/time of the next leg is greated
00135       // than the off date/time of the current leg. Throw an exception
00136       // otherwise.
00137
00138       // TODO: specify, in the schedule input file specifications, that the
00139       // legs should be given in their natural order.
00140       // Then, replace the assertion by a thrown exception.
00141       //
00142       // Check that the legs are given in their natural order. If the schedule
00143       // input does not respect that assumption, the following assertion will
00144       // fail.
00145       if (itLeg != lLegList.begin()) {
00146         const stdair::AirportCode_T& currentBoardingPoint =
00147           lLegDate.getBoardingPoint();
00148         assert (currentBoardingPoint == previousOffPoint);
00149       }
00150
00151       // Set the local variable for the next iteration
00152       previousOffPoint = lLegDate.getOffPoint();
00153     }
00154
00155     // Iterate on the segment structures
00156     const SegmentStructList_T& lSegmentList = iFlightPeriod.
    _segmentList;
00157     for (SegmentStructList_T::const_iterator itSegment = lSegmentList.begin();
00158          itSegment != lSegmentList.end(); ++itSegment) {
00159       const SegmentStruct& lSegment = *itSegment;
00160
00161       createSegmentDate (ioBomRoot, *lFlightDate_ptr, lSegment);
00162     }
00163
00164     createRoutingLegKey (*lFlightDate_ptr);
00165   }
00166
00167   // ////////////////////////////////////////////////////////////////////
```

```
00168   void InventoryGenerator::
00169   createRoutingLegKey (stdair::FlightDate& ioFlightDate) {
00170
00171     // Browse the list of segment-dates and create direct accesses
00172     // within each segment-date.
00173     const stdair::SegmentDateList_T& lSegmentDateList =
00174       stdair::BomManager::getList<stdair::SegmentDate> (ioFlightDate);
00175     for (stdair::SegmentDateList_T::const_iterator itSegmentDate =
00176            lSegmentDateList.begin();
00177          itSegmentDate != lSegmentDateList.end(); ++itSegmentDate) {
00178
00179       stdair::SegmentDate* lCurrentSegmentDate_ptr = *itSegmentDate;
00180       assert (lCurrentSegmentDate_ptr != NULL);
00181
00182       const stdair::AirportCode_T& lBoardingPoint =
00183         lCurrentSegmentDate_ptr->getBoardingPoint();
00184
00185       stdair::AirportCode_T currentBoardingPoint = lBoardingPoint;
00186       const stdair::AirportCode_T& lOffPoint =
00187         lCurrentSegmentDate_ptr->getOffPoint();
00188
00189       // Add a sanity check so as to ensure that the loop stops. If
00190       // there are more than MAXIMAL_NUMBER_OF_LEGS legs, there is
00191       // an issue somewhere in the code (not in the parser, as the
00192       // segments are derived from the legs thanks to the
00193       // FlightPeriodStruct::buildSegments() method).
00194       unsigned short i = 1;
00195       while (currentBoardingPoint != lOffPoint
00196              && i <= stdair::MAXIMAL_NUMBER_OF_LEGS_IN_FLIGHT) {
00197         // Retrieve the (unique) LegDate getting that Boarding Point
00198         stdair::LegDate& lLegDate = stdair::BomManager::
00199           getObject<stdair::LegDate> (ioFlightDate, currentBoardingPoint);
00200
00201         // Link the SegmentDate and LegDate together
00202         const std::string& lRoutingKeyStr = lLegDate.describeRoutingKey();
00203         lCurrentSegmentDate_ptr->addLegKey(lRoutingKeyStr);
00204
00205         // Prepare the next iteration
00206         currentBoardingPoint = lLegDate.getOffPoint();
00207         ++i;
00208       }
00209       assert (i <= stdair::MAXIMAL_NUMBER_OF_LEGS_IN_FLIGHT);
00210     }
00211   }
00212
00213   // ////////////////////////////////////////////////////////////////////
00214   stdair::LegDate& InventoryGenerator::
00215   createLegDate (stdair::FlightDate& ioFlightDate,
00216                  const stdair::Date_T& iReferenceDate,
00217                  const LegStruct& iLeg) {
00218     // Create the leg-date corresponding to the boarding point.
00219     stdair::LegDateKey lKey (iLeg._boardingPoint);
00220     stdair::LegDate& lLegDate =
00221       stdair::FacBom<stdair::LegDate>::instance().create (lKey);
00222     stdair::FacBomManager::addToListAndMap (ioFlightDate, lLegDate);
00223     stdair::FacBomManager::linkWithParent (ioFlightDate, lLegDate);
00224
00225     // Set the leg-date attributes
00226     iLeg.fill (iReferenceDate, lLegDate);
00227
00228     // Iterate on the cabins
00229     const LegCabinStructList_T& lCabinList = iLeg.
    _cabinList;
00230     for (LegCabinStructList_T::const_iterator itCabin = lCabinList.begin();
00231          itCabin != lCabinList.end(); ++itCabin) {
00232       const LegCabinStruct& lCabin = *itCabin;
00233
00234       // Create the leg-cabin-branch of the leg-date
00235       createLegCabin (lLegDate, lCabin);
00236     }
00237
00238     return lLegDate;
00239   }
00240
00241   // ////////////////////////////////////////////////////////////////////
00242   void InventoryGenerator::
00243   createLegCabin (stdair::LegDate& ioLegDate,
00244                   const LegCabinStruct& iCabin) {
00245     // Instantiate an leg-cabin object with the corresponding cabin code
00246     const stdair::LegCabinKey lKey (iCabin._cabinCode);
00247     stdair::LegCabin& lLegCabin =
00248       stdair::FacBom<stdair::LegCabin>::instance().create (lKey);
00249     stdair::FacBomManager::addToListAndMap (ioLegDate, lLegCabin);
00250     stdair::FacBomManager::linkWithParent (ioLegDate, lLegCabin);
00251
00252     // Set the Leg-Cabin attributes
00253     iCabin.fill (lLegCabin);
```

```
00254
00255     // Iterate on the bucket
00256     const BucketStructList_T& lBucketList = iCabin.
      _bucketList;
00257     for (BucketStructList_T::const_iterator itBucket = lBucketList.begin();
00258          itBucket != lBucketList.end(); ++itBucket) {
00259       const BucketStruct& lBucket = *itBucket;
00260
00261       // Create the bucket of the leg-cabin
00262       createBucket (lLegCabin, lBucket);
00263     }
00264   }
00265
00266   // ////////////////////////////////////////////////////////////////////
00267   void InventoryGenerator::createBucket (stdair::LegCabin& ioLegCabin,
00268                                          const BucketStruct& iBucket) {
00269     // Instantiate a bucket object with the corresponding seat index
00270     const stdair::BucketKey lKey (iBucket._seatIndex);
00271     stdair::Bucket& lBucket =
00272       stdair::FacBom<stdair::Bucket>::instance().create (lKey);
00273     stdair::FacBomManager::addToListAndMap (ioLegCabin, lBucket);
00274     stdair::FacBomManager::linkWithParent (ioLegCabin, lBucket);
00275
00276     // Set the Bucket attributes
00277     iBucket.fill (lBucket);
00278   }
00279
00280   // ////////////////////////////////////////////////////////////////////
00281   void InventoryGenerator::
00282   createSegmentDate (stdair::BomRoot& ioBomRoot,
00283                      stdair::FlightDate& ioFlightDate,
00284                      const SegmentStruct& iSegment) {
00285     // Set the segment-date primary key
00286     const stdair::AirportCode_T& lBoardingPoint = iSegment._boardingPoint;
00287     const stdair::AirportCode_T& lOffPoint = iSegment._offPoint;
00288     stdair::SegmentDateKey lSegmentDateKey (lBoardingPoint, lOffPoint);
00289     // Instantiate an segment-date object with the key.
00290     stdair::SegmentDate& lSegmentDate =
00291       stdair::FacBom<stdair::SegmentDate>::instance().create (lSegmentDateKey);
00292     stdair::FacBomManager::addToListAndMap (ioFlightDate, lSegmentDate);
00293     stdair::FacBomManager::linkWithParent (ioFlightDate, lSegmentDate);
00294
00295     // Set the segment-date attributes
00296     iSegment.fill (lSegmentDate);
00297
00298     // Iterate on the Cabins
00299     const SegmentCabinStructList_T& lCabinList =
      iSegment._cabinList;
00300     for (SegmentCabinStructList_T::const_iterator itCabin =
00301            lCabinList.begin(); itCabin != lCabinList.end(); ++itCabin) {
00302       const SegmentCabinStruct& lCabin = *itCabin;
00303
00304       // Create the segment-cabin-branch of the segment-date BOM
00305       createSegmentCabin (ioBomRoot, lSegmentDate, lCabin);
00306     }
00307   }
00308
00309   // ////////////////////////////////////////////////////////////////////
00310   void InventoryGenerator::
00311   createSegmentCabin (stdair::BomRoot& ioBomRoot,
00312                       stdair::SegmentDate& ioSegmentDate,
00313                       const SegmentCabinStruct& iCabin) {
00314
00315     // Instantiate an segment-cabin object with the corresponding cabin code
00316     stdair::SegmentCabinKey lKey (iCabin._cabinCode);
00317     stdair::SegmentCabin& lSegmentCabin =
00318       stdair::FacBom<stdair::SegmentCabin>::instance().create (lKey);
00319
00320     // Link the segment-cabin to its parent, the segment-date
00321     stdair::FacBomManager::addToListAndMap (ioSegmentDate, lSegmentCabin);
00322     stdair::FacBomManager::linkWithParent (ioSegmentDate, lSegmentCabin);
00323
00324     // Set the segment-cabin attributes
00325     iCabin.fill (lSegmentCabin);
00326
00327     // Create the list of fare families
00328     for (FareFamilyStructList_T::const_iterator itFareFamily =
00329            iCabin._fareFamilies.begin();
00330          itFareFamily != iCabin._fareFamilies.end(); itFareFamily++) {
00331       const FareFamilyStruct& lFareFamilyStruct = *itFareFamily;
00332
00333       // Create the fare families and the booking classes.
00334       createFareFamily (ioBomRoot, lSegmentCabin, lFareFamilyStruct);
00335     }
00336
00337     const unsigned int lNbOfFareFamilies = iCabin._fareFamilies.size();
00338     if (lNbOfFareFamilies > 1) {
```

```
00339       lSegmentCabin.activateFareFamily();
00340     }
00341
00342     // Create the display nesting structure.
00343     createDisplayNestingStructure (lSegmentCabin);
00344   }
00345
00346   // ////////////////////////////////////////////////////////////////////
00347   void InventoryGenerator::
00348   createFareFamily (stdair::BomRoot& ioBomRoot,
00349                     stdair::SegmentCabin& ioSegmentCabin,
00350                     const FareFamilyStruct& iFF) {
00351     // Instantiate an segment-cabin object with the corresponding cabin code
00352     stdair::FareFamilyKey lKey (iFF._familyCode);
00353     stdair::FareFamily& lFareFamily =
00354       stdair::FacBom<stdair::FareFamily>::instance().create (lKey);
00355
00356     // Link the fare family to its parent, the segment-cabin
00357     stdair::FacBomManager::addToListAndMap (ioSegmentCabin, lFareFamily);
00358     stdair::FacBomManager::linkWithParent (ioSegmentCabin, lFareFamily);
00359
00360     // Set the fare family attributes
00361     iFF.fill (lFareFamily);
00362     const stdair::FRAT5Curve_T& lFRAT5Curve =
00363       ioBomRoot.getFRAT5Curve (iFF._frat5CurveKey);
00364     lFareFamily.setFrat5Curve (lFRAT5Curve);
00365     const stdair::FFDisutilityCurve_T& lDisutilityCurve =
00366       ioBomRoot.getFFDisutilityCurve (iFF._ffDisutilityCurveKey);
00367     lFareFamily.setDisutilityCurve (lDisutilityCurve);
00368
00369     // Iterate on the classes
00370     const stdair::ClassList_String_T& lClassList = iFF._classes;
00371     for (stdair::ClassList_String_T::const_iterator itClass =
00372            lClassList.begin(); itClass != lClassList.end(); ++itClass) {
00373       // Transform the single-character class code into a STL string
00374       std::ostringstream ostr;
00375       ostr << *itClass;
00376       const stdair::ClassCode_T lClassCode (ostr.str());
00377
00378       // Create the booking class branch of the segment-cabin BOM
00379       createClass (lFareFamily, lClassCode);
00380     }
00381   }
00382
00383   // ////////////////////////////////////////////////////////////////////
00384   void InventoryGenerator::createClass (stdair::FareFamily& ioFareFamily,
00385                                         const stdair::ClassCode_T& iClassCode)
      {
00386
00387     // Instantiate a booking class object with the given class code
00388     const stdair::BookingClassKey lClassKey (iClassCode);
00389     stdair::BookingClass& lClass =
00390       stdair::FacBom<stdair::BookingClass>::instance().create (lClassKey);
00391
00392     // Link the booking-class to the fare family
00393     stdair::FacBomManager::addToListAndMap (ioFareFamily, lClass);
00394     stdair::FacBomManager::linkWithParent (ioFareFamily, lClass);
00395
00396     // Link the booking-class to the segment-cabin
00397     stdair::SegmentCabin& lSegmentCabin =
00398       stdair::BomManager::getParent<stdair::SegmentCabin> (ioFareFamily);
00399     stdair::FacBomManager::addToListAndMap (lSegmentCabin, lClass);
00400
00401     // Link the booking-class to the segment-date
00402     stdair::SegmentDate& lSegmentDate =
00403       stdair::BomManager::getParent<stdair::SegmentDate> (lSegmentCabin);
00404     stdair::FacBomManager::addToListAndMap (lSegmentDate, lClass);
00405   }
00406
00407   // ////////////////////////////////////////////////////////////////////
00408   void InventoryGenerator::
00409   createDisplayNestingStructure (stdair::SegmentCabin& ioSegmentCabin) {
00410     // Create the nesting structure.
00411     stdair::NestingStructureKey lKey (stdair::DISPLAY_NESTING_STRUCTURE_CODE);
00412     stdair::SimpleNestingStructure& lNestingStructure =
00413       stdair::FacBom<stdair::SimpleNestingStructure>::instance().create(lKey);
00414     stdair::FacBomManager::addToListAndMap (ioSegmentCabin, lNestingStructure);
00415     stdair::FacBomManager::linkWithParent (ioSegmentCabin, lNestingStructure);
00416
00417     // Browse the list of booking classes and create the nesting structure
00418     // based on that list. Each nesting node consists of a booking class.
00419     const stdair::BookingClassList_T& lBCList =
00420       stdair::BomManager::getList<stdair::BookingClass>(ioSegmentCabin);
00421     for (stdair::BookingClassList_T::const_iterator itBC = lBCList.begin();
00422          itBC != lBCList.end(); ++itBC) {
00423       stdair::BookingClass* lBC_ptr = *itBC;
00424       assert (lBC_ptr != NULL);
```

```
00425
00426       // Create a nesting node
00427       stdair::NestingNodeCode_T lNodeCode (lBC_ptr->describeKey());
00428       stdair::NestingNodeKey lNodeKey (lNodeCode);
00429       stdair::NestingNode& lNestingNode =
00430         stdair::FacBom<stdair::NestingNode>::instance().create (lNodeKey);
00431       stdair::FacBomManager::addToList (lNestingStructure, lNestingNode);
00432       stdair::FacBomManager::linkWithParent (lNestingStructure, lNestingNode);
00433
00434       // Add the booking class to the node.
00435       stdair::FacBomManager::addToList (lNestingNode, *lBC_ptr);
00436     }
00437   }
00438 }
```

23.139 airinv/command/InventoryGenerator.hpp File Reference

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

Classes

- class AIRINV::InventoryGenerator

  Class handling the generation / instantiation of the Inventory BOM.

Namespaces

- namespace stdair

  Forward declarations.
- namespace AIRINV
- namespace AIRINV::ScheduleParserHelper

23.140 InventoryGenerator.hpp

```
00001 #ifndef __AIRINV_CMD_INVENTORYGENERATOR_HPP
00002 #define __AIRINV_CMD_INVENTORYGENERATOR_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // StdAir
00008 #include <stdair/command/CmdAbstract.hpp>
00009 // Airinv
00010 #include <airinv/AIRINV_Types.hpp>
00011
00013 namespace stdair {
00014   class BomRoot;
00015   class Inventory;
00016   class FlightDate;
00017   class LegDate;
00018   class LegCabin;
00019   class SegmentDate;
00020   class SegmentCabin;
00021   class FareFamily;
00022 }
00023
00024 namespace AIRINV {
00025
00026   // Forward declarations
00027   struct FlightPeriodStruct;
00028   struct LegStruct;
00029   struct SegmentStruct;
00030   struct LegCabinStruct;
00031   struct SegmentCabinStruct;
00032   struct FareFamilyStruct;
00033   struct BucketStruct;
00034   namespace ScheduleParserHelper {
00035     struct doEndFlight;
00036   }
00037
```

```
00042   class InventoryGenerator : public stdair::CmdAbstract {
00048     friend class FlightPeriodFileParser;
00049     friend class FFFlightPeriodFileParser;
00050     friend struct ScheduleParserHelper::doEndFlight
    ;
00051     friend class ScheduleParser;
00052
00053   private:
00058     static void createFlightDate (stdair::BomRoot&,
00059                                   const FlightPeriodStruct&);
00060
00064     static void createFlightDate (stdair::BomRoot&, stdair::Inventory&,
00065                                   const stdair::Date_T&,
00066                                   const FlightPeriodStruct&);
00067
00071     static void createRoutingLegKey (stdair::FlightDate&);
00072
00076     static stdair::LegDate& createLegDate (stdair::FlightDate&,
00077                                            const stdair::Date_T&,
00078                                            const LegStruct&);
00079
00083     static void createLegCabin (stdair::LegDate&, const LegCabinStruct
    &);
00084
00088     static void createBucket (stdair::LegCabin&, const BucketStruct
    &);
00089
00093     static void createSegmentDate (stdair::BomRoot&, stdair::FlightDate&,
00094                                    const SegmentStruct&);
00095
00099     static void createSegmentCabin (stdair::BomRoot&, stdair::SegmentDate&,
00100                                     const SegmentCabinStruct&
    );
00101
00105     static void createFareFamily (stdair::BomRoot&, stdair::SegmentCabin&,
00106                                   const FareFamilyStruct&);
00107
00111     static void createClass (stdair::FareFamily&,
00112                              const stdair::ClassCode_T&);
00113
00117     static void createDisplayNestingStructure (stdair::SegmentCabin&);
00118   };
00119
00120 }
00121 #endif // __AIRINV_CMD_INVENTORYGENERATOR_HPP
```

23.141 airinv/command/InventoryManager.cpp File Reference

```
#include <exception>
```

```
#include <algorithm>
#include <boost/make_shared.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomKeyManager.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/SegmentSnapshotTable.hpp>
#include <stdair/bom/TravelSolutionStruct.hpp>
#include <stdair/bom/FareOptionStruct.hpp>
#include <stdair/bom/EventStruct.hpp>
#include <stdair/bom/SnapshotStruct.hpp>
#include <stdair/bom/RMEventStruct.hpp>
#include <stdair/bom/BomRetriever.hpp>
#include <stdair/factory/FacBomManager.hpp>
#include <stdair/factory/FacBom.hpp>
#include <stdair/service/Logger.hpp>
#include <sevmgr/SEVMGR_Service.hpp>
#include <airinv/AIRINV_Types.hpp>
#include <airinv/bom/BomRootHelper.hpp>
#include <airinv/bom/InventoryHelper.hpp>
#include <airinv/bom/FlightDateHelper.hpp>
#include <airinv/bom/SegmentCabinHelper.hpp>
#include <airinv/command/InventoryManager.hpp>
```

Namespaces

- namespace AIRINV

23.142 InventoryManager.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <exception>
00006 #include <algorithm> // To use min
00007 // Boost
00008 #include <boost/make_shared.hpp>
00009 // StdAir
00010 #include <stdair/basic/BasConst_Inventory.hpp>
00011 #include <stdair/basic/BasConst_BomDisplay.hpp>
00012 #include <stdair/bom/BomManager.hpp>
00013 #include <stdair/bom/BomKeyManager.hpp>
00014 #include <stdair/bom/BomRoot.hpp>
00015 #include <stdair/bom/Inventory.hpp>
00016 #include <stdair/bom/FlightDate.hpp>
00017 #include <stdair/bom/SegmentDate.hpp>
00018 #include <stdair/bom/SegmentCabin.hpp>
00019 #include <stdair/bom/LegDate.hpp>
00020 #include <stdair/bom/LegCabin.hpp>
00021 #include <stdair/bom/FareFamily.hpp>
00022 #include <stdair/bom/BookingClass.hpp>
00023 #include <stdair/bom/SegmentSnapshotTable.hpp>
00024 #include <stdair/bom/TravelSolutionStruct.hpp>
00025 #include <stdair/bom/FareOptionStruct.hpp>
```

```
00026 #include <stdair/bom/EventStruct.hpp>
00027 #include <stdair/bom/SnapshotStruct.hpp>
00028 #include <stdair/bom/RMEventStruct.hpp>
00029 #include <stdair/bom/FareFamily.hpp> // Contains the definition of
      FareFamilyList_T
00030 #include <stdair/bom/BookingClass.hpp> //
00031 #include <stdair/bom/BomRetriever.hpp>
00032 #include <stdair/factory/FacBomManager.hpp>
00033 #include <stdair/factory/FacBom.hpp>
00034 #include <stdair/service/Logger.hpp>// SEvMgr
00035 #include <sevmgr/SEVMGR_Service.hpp>
00036 // AirInv
00037 #include <airinv/AIRINV_Types.hpp>
00038 #include <airinv/bom/BomRootHelper.hpp>
00039 #include <airinv/bom/InventoryHelper.hpp>
00040 #include <airinv/bom/FlightDateHelper.hpp>
00041 #include <airinv/bom/SegmentCabinHelper.hpp>
00042 #include <airinv/command/InventoryManager.hpp
      >
00043
00044 namespace AIRINV {
00045
00046   // ////////////////////////////////////////////////////////////////////
00047   void InventoryManager::
00048   calculateAvailability (const stdair::BomRoot& iBomRoot,
00049                          stdair::TravelSolutionStruct& ioTravelSolution) {
00050
00051     stdair::PartnershipTechnique::EN_PartnershipTechnique
      lENPartnershipTechnique =
00052       stdair::PartnershipTechnique::NONE;
00053
00054     // Browse the list of segments and get the availability for the
00055     // children classes.
00056     const stdair::SegmentPath_T& lSegmentPath =
00057       ioTravelSolution.getSegmentPath();
00058     for (stdair::SegmentPath_T::const_iterator itSK = lSegmentPath.begin();
00059          itSK != lSegmentPath.end(); ++itSK) {
00060       const std::string& lSegmentKey = *itSK;
00061       const stdair::InventoryKey lInvKey =
00062         stdair::BomKeyManager::extractInventoryKey (lSegmentKey);
00063       stdair::Inventory& lInventory =
00064         stdair::BomManager::getObject<stdair::Inventory>(iBomRoot,
00065                                                          lInvKey.toString());
00066
00067       lENPartnershipTechnique = lInventory.getPartnershipTechnique();
00068
00069       switch (lENPartnershipTechnique) {
00070
00071       case stdair::PartnershipTechnique::NONE:{
00072         InventoryHelper::calculateAvailability
      (lInventory, lSegmentKey,
00073                                                 ioTravelSolution);
00074         break;
00075       }
00076       default:{
00077         InventoryHelper::getYieldAndBidPrice
      (lInventory, lSegmentKey,
00078                                               ioTravelSolution);
00079         break;
00080       }
00081       }
00082
00083     }
00084
00085     switch (lENPartnershipTechnique) {
00086     case stdair::PartnershipTechnique::NONE:{
00087       // Compute the availabitliy for each fare option using the AU's.
00088       calculateAvailabilityByAU (ioTravelSolution);
00089       break;
00090     }
00091     case stdair::PartnershipTechnique::RAE_DA:
00092     case stdair::PartnershipTechnique::RAE_YP:{
00093       // 1. Compute the availability for each fare option using RAE
00094       calculateAvailabilityByRAE (ioTravelSolution);
00095       break;
00096     }
00097     case stdair::PartnershipTechnique::IBP_DA:
00098     case stdair::PartnershipTechnique::IBP_YP:{
00099       // 2. Compute the availability for each fare option using protective IBP
00100       calculateAvailabilityByProtectiveIBP (ioTravelSolution);
00101       break;
00102     }
00103     case stdair::PartnershipTechnique::IBP_YP_U:
00104     case stdair::PartnershipTechnique::RMC:
00105     case stdair::PartnershipTechnique::A_RMC:{
00106       // 3. Compute the availability for each fare option using IBP
00107       calculateAvailabilityByIBP (ioTravelSolution);
```

```
00108       break;
00109     }
00110     default: {
00111       assert (false);
00112       break;
00113     }
00114     }
00115
00116   }
00117
00118   // ////////////////////////////////////////////////////////////////////
00119   void InventoryManager::
00120   calculateAvailabilityByAU (stdair::TravelSolutionStruct& ioTravelSolution) {
00121
00122     // MODIF: segment path string for availability display
00123     std::ostringstream oStr;
00124     const stdair::SegmentPath_T& lSP = ioTravelSolution.getSegmentPath();
00125     for (stdair::SegmentPath_T::const_iterator itSP = lSP.begin();
00126          itSP != lSP.end(); itSP++) {
00127       oStr << *itSP << ";";
00128     }
00129
00130     // Browse the fare options
00131     stdair::FareOptionList_T& lFOList = ioTravelSolution.getFareOptionListRef()
      ;
00132     for (stdair::FareOptionList_T::iterator itFO = lFOList.begin();
00133          itFO != lFOList.end(); ++itFO) {
00134
00135       stdair::FareOptionStruct& lFO = *itFO;
00136
00137       // Check the availability
00138       const stdair::ClassList_StringList_T& lClassPath = lFO.getClassPath();
00139
00140       const stdair::ClassAvailabilityMapHolder_T& lClassAvailabilityMapHolder =
00141         ioTravelSolution.getClassAvailabilityMapHolder();
00142
00143       // Initialise the flag stating whether the availability is enough
00144       stdair::Availability_T lAvl =
00145         std::numeric_limits<stdair::Availability_T>::max();
00146
00147       // Sanity check: the travel solution must contain two lists,
00148       // one for the booking class availabilities, the other for the
00149       // fare options.
00150       assert (lClassAvailabilityMapHolder.empty() == false
00151               && lClassPath.empty() == false);
00152
00153       // List of booking class availability maps (one map per segment)
00154       stdair::ClassAvailabilityMapHolder_T::const_iterator itCAMH =
00155         lClassAvailabilityMapHolder.begin();
00156
00157       // List of fare options
00158       stdair::ClassList_StringList_T::const_iterator itClassList =
00159         lClassPath.begin();
00160
00161       // Browse both lists at the same time, i.e., one element per segment
00162       for (; itCAMH != lClassAvailabilityMapHolder.end()
00163              && itClassList != lClassPath.end(); ++itCAMH, ++itClassList) {
00164
00165         // Retrieve the booking class list for the current segment
00166         const stdair::ClassList_String_T& lCurrentClassList = *itClassList;
00167         assert (lCurrentClassList.size() > 0);
00168
00169         // TODO: instead of just extracting the first booking class,
00170         //       perform a choice on the full list of classes.
00171         // Extract one booking class key (class code)
00172         stdair::ClassCode_T lFirstClass;
00173         lFirstClass.append (lCurrentClassList, 0, 1);
00174
00175         // Retrieve the booking class map for the current segment
00176         const stdair::ClassAvailabilityMap_T& lClassAvlMap = *itCAMH;
00177
00178         // Retrieve the availability of the chosen booking class
00179         const stdair::ClassAvailabilityMap_T::const_iterator itClassAvl =
00180           lClassAvlMap.find (lFirstClass);
00181
00182         if (itClassAvl == lClassAvlMap.end()) {
00183           // DEBUG
00184           STDAIR_LOG_DEBUG ("No availability has been set up for the class '"
00185                             << lFirstClass << "'. Travel solution: "
00186                             << ioTravelSolution.display());
00187         }
00188         assert (itClassAvl != lClassAvlMap.end());
00189
00190         const stdair::Availability_T& lCurrentAvl = itClassAvl->second;
00191         if (lAvl > lCurrentAvl) {
00192           lAvl = lCurrentAvl;
00193         }
```

```
00194       }
00195
00196       lFO.setAvailability (lAvl);
00197
00198       //MODIF: availability display
00199       STDAIR_LOG_DEBUG ("Fare option " << lFO.describe() << ", "
00200                         << "Availability " << lFO.getAvailability() << ", "
00201                         << "Segment Path " << oStr.str());
00202     }
00203   }
00204
00205   // \todo: the following code must be either re-written or removed.
00206   //        There is indeed a lot of code duplication.
00207   // ////////////////////////////////////////////////////////////////////
00208   void InventoryManager::
00209   calculateAvailabilityByRAE (stdair::TravelSolutionStruct& ioTravelSolution) {
00210
00211     std::ostringstream oStr;
00212     const stdair::SegmentPath_T& lSP = ioTravelSolution.getSegmentPath();
00213     for (stdair::SegmentPath_T::const_iterator itSP = lSP.begin();
00214          itSP != lSP.end(); itSP++) {
00215       oStr << *itSP << ";";
00216     }
00217
00218     //Retrieve bid price vector and yield maps
00219     const stdair::ClassYieldMapHolder_T& lClassYieldMapHolder =
00220       ioTravelSolution.getClassYieldMapHolder();
00221     const stdair::ClassBpvMapHolder_T& lClassBpvMapHolder =
00222       ioTravelSolution.getClassBpvMapHolder();
00223
00224     //Retrieve the list of fare options and browse it
00225     stdair::FareOptionList_T& lFOList = ioTravelSolution.getFareOptionListRef()
    ;
00226     for (stdair::FareOptionList_T::iterator itFO = lFOList.begin();
00227          itFO != lFOList.end(); ++itFO) {
00228
00229       stdair::FareOptionStruct& lFO = *itFO;
00230
00231       stdair::ClassYieldMapHolder_T::const_iterator itCYM =
00232         lClassYieldMapHolder.begin();
00233       stdair::ClassBpvMapHolder_T::const_iterator itCBPM =
00234         lClassBpvMapHolder.begin();
00235
00236       const stdair::ClassList_StringList_T& lClassPath = lFO.getClassPath();
00237
00238
00239       // Sanity checks
00240       assert (lClassPath.size() == lClassYieldMapHolder.size());
00241       assert (lClassPath.size() == lClassBpvMapHolder.size());
00242
00243       // Browse class path, class-yield maps, class-(bid price vector) maps.
00244       // Each iteration corresponds to one segment.
00245
00246       std::ostringstream oCPStr;
00247       for (stdair::ClassList_StringList_T::const_iterator itCL =
00248              lClassPath.begin();
00249            itCL != lClassPath.end(); ++itCL, ++itCYM, ++itCBPM) {
00250
00251         // Class path determination
00252         if (itCL == lClassPath.begin()) {
00253           oCPStr << *itCL;
00254
00255         } else {
00256           oCPStr << "-" << *itCL;
00257         }
00258
00259         const stdair::ClassList_String_T& lCL = *itCL;
00260         stdair::ClassCode_T lCC;
00261         lCC.append (lCL, 0, 1);
00262
00263         const stdair::ClassYieldMap_T& lCYM = *itCYM;
00264         stdair::ClassYieldMap_T::const_iterator itCCCYM = lCYM.find (lCC);
00265         assert (itCCCYM != lCYM.end());
00266
00267         const stdair::ClassBpvMap_T& lCBPM = *itCBPM;
00268         stdair::ClassBpvMap_T::const_iterator itCCCBPM = lCBPM.find (lCC);
00269         assert (itCCCBPM != lCBPM.end());
00270
00271         const stdair::BidPriceVector_T* lBidPriceVector_ptr = itCCCBPM->second;
00272         assert (lBidPriceVector_ptr != NULL);
00273
00274         // Initialization of fare option availability
00275         if (itCL == lClassPath.begin()) {
00276           lFO.setAvailability (lBidPriceVector_ptr->size());
00277         }
00278
00279         // Availability update
```

```
00280         if (lFO.getAvailability() > 0) {
00281
00282           //Segment availability calculation
00283           stdair::BidPriceVector_T lReverseBPV (lBidPriceVector_ptr->size());
00284           std::reverse_copy (lBidPriceVector_ptr->begin(),
00285                              lBidPriceVector_ptr->end(),
00286                              lReverseBPV.begin());
00287
00288           const stdair::YieldValue_T& lYield = itCCCYM->second;
00289           stdair::BidPriceVector_T::const_iterator lBidPrice =
00290             std::upper_bound (lReverseBPV.begin(), lReverseBPV.end(), lYield);
00291
00292           const stdair::Availability_T lAvl = lBidPrice - lReverseBPV.begin();
00293
00294           // Availability update
00295           lFO.setAvailability (std::min (lFO.getAvailability(), lAvl));
00296         }
00297       }
00298
00299       // DEBUG
00300       STDAIR_LOG_DEBUG ("Fare option: " << lFO.describe() << ", "
00301                         << "Availability: " << lFO.getAvailability() << ", "
00302                         << "Segment Path: " << oStr.str() << ", ");
00303     }
00304   }
00305
00306   // \todo: the following code must be either re-written or removed.
00307   //        There is indeed a lot of code duplication.
00308   // ////////////////////////////////////////////////////////////////////
00309   void InventoryManager::
00310   calculateAvailabilityByIBP (stdair::TravelSolutionStruct& ioTravelSolution) {
00311     std::ostringstream oStr;
00312
00313     // Yield valuation coefficient for multi-segment travel solutions
00314     double alpha = 1.0;
00315
00316     const stdair::SegmentPath_T& lSP = ioTravelSolution.getSegmentPath();
00317     for (stdair::SegmentPath_T::const_iterator itSP = lSP.begin();
00318          itSP != lSP.end(); itSP++) {
00319       oStr << *itSP << ";";
00320     }
00321
00322     //Retrieve bid price vector and yield maps
00323     const stdair::ClassYieldMapHolder_T& lClassYieldMapHolder =
00324       ioTravelSolution.getClassYieldMapHolder();
00325     const stdair::ClassBpvMapHolder_T& lClassBpvMapHolder =
00326       ioTravelSolution.getClassBpvMapHolder();
00327
00328     // Retrieve the list of fare options and browse it
00329     stdair::FareOptionList_T& lFOList = ioTravelSolution.getFareOptionListRef()
    ;
00330     for (stdair::FareOptionList_T::iterator itFO = lFOList.begin();
00331          itFO != lFOList.end(); ++itFO) {
00332
00333       stdair::FareOptionStruct& lFO = *itFO;
00334
00335       stdair::ClassYieldMapHolder_T::const_iterator itCYM =
00336         lClassYieldMapHolder.begin();
00337       stdair::ClassBpvMapHolder_T::const_iterator itCBPM =
00338         lClassBpvMapHolder.begin();
00339
00340       const stdair::ClassList_StringList_T& lClassPath = lFO.getClassPath();
00341
00342       // Sanity checks
00343       assert (lClassPath.size() == lClassYieldMapHolder.size());
00344       assert (lClassPath.size() == lClassBpvMapHolder.size());
00345
00346       // Yield is taken to be equal to fare (connecting flights)
00347
00348       // \todo: take yield instead
00349       stdair::YieldValue_T lTotalYield = lFO.getFare();
00350       // Bid price initialisation
00351       stdair::BidPrice_T lTotalBidPrice = 0;
00352
00353       // Browse class path, class-yield maps, class-(bid price vector) maps.
00354       // Each iteration corresponds to one segment.
00355
00356       std::ostringstream oCPStr;
00357       for (stdair::ClassList_StringList_T::const_iterator itCL =
00358              lClassPath.begin();
00359            itCL != lClassPath.end(); ++itCL, ++itCYM, ++itCBPM) {
00360
00361         // Class path determination
00362         if (itCL == lClassPath.begin()) {
00363           oCPStr << *itCL;
00364
00365         } else {
```

```
00366           oCPStr << "-" << *itCL;
00367         }
00368
00369         const stdair::ClassList_String_T& lCL = *itCL;
00370         stdair::ClassCode_T lCC;
00371         lCC.append (lCL, 0, 1);
00372
00373         const stdair::ClassYieldMap_T& lCYM = *itCYM;
00374         stdair::ClassYieldMap_T::const_iterator itCCCYM = lCYM.find (lCC);
00375         assert (itCCCYM != lCYM.end());
00376
00377         const stdair::ClassBpvMap_T& lCBPM = *itCBPM;
00378         stdair::ClassBpvMap_T::const_iterator itCCCBPM = lCBPM.find (lCC);
00379         assert (itCCCBPM != lCBPM.end());
00380
00381         const stdair::BidPriceVector_T* lBidPriceVector_ptr = itCCCBPM->second;
00382         assert (lBidPriceVector_ptr != NULL);
00383
00384         //Initialization of fare option availability
00385         if (itCL == lClassPath.begin()) {
00386           lFO.setAvailability (lBidPriceVector_ptr->size());
00387         }
00388
00389         // Availability update
00390         if (lFO.getAvailability() > 0) {
00391           //Segment availability calculation
00392           stdair::BidPriceVector_T lReverseBPV (lBidPriceVector_ptr->size());
00393           std::reverse_copy (lBidPriceVector_ptr->begin(),
00394                              lBidPriceVector_ptr->end(), lReverseBPV.begin());
00395
00396           const stdair::YieldValue_T& lYield = itCCCYM->second;
00397           stdair::BidPriceVector_T::const_iterator lBidPrice =
00398             std::upper_bound (lReverseBPV.begin(), lReverseBPV.end(), lYield);
00399
00400           const stdair::Availability_T lAvl = lBidPrice - lReverseBPV.begin();
00401
00402           // Availability update
00403           lFO.setAvailability (std::min(lFO.getAvailability(), lAvl));
00404         }
00405
00406         // Total bid price calculation
00407         if (lBidPriceVector_ptr->size() > 0) {
00408           lTotalBidPrice += lBidPriceVector_ptr->back();
00409
00410         } else {
00411           lTotalBidPrice = std::numeric_limits<stdair::BidPrice_T>::max();
00412         }
00413
00414         // Total yield calculation (has been replaced by total fare).
00415         //lTotalYield += lYield;
00416       }
00417       // Multi-segment bid price control
00418
00419       if (lClassPath.size() > 1) {
00420         if (lFO.getAvailability() > 0) {
00421           const stdair::Availability_T lAvl =
00422             alpha * lTotalYield >= lTotalBidPrice;
00423           lFO.setAvailability (lAvl * lFO.getAvailability());
00424
00425         } else {
00426           const stdair::Availability_T lAvl =
00427             alpha * lTotalYield >= lTotalBidPrice;
00428           lFO.setAvailability (lAvl);
00429         }
00430
00431         // DEBUG
00432         STDAIR_LOG_DEBUG ("Class: " << oCPStr.str()
00433                           << ", " << "Yield: " << alpha*lTotalYield << ", "
00434                           << "Bid price: " << lTotalBidPrice << ", "
00435                           << "Remaining capacity: " << "Undefined" << " "
00436                           << "Segment date: " << oStr.str());
00437       }
00438
00439       // DEBUG
00440       STDAIR_LOG_DEBUG ("Fare option " << lFO.describe() << ", "
00441                         << "Availability " << lFO.getAvailability() << ", "
00442                         << "Segment Path " << oStr.str() << ", ");
00443     }
00444   }
00445
00446   // \todo: the following code must be either re-written or removed.
00447   //        There is indeed a lot of code duplication.
00448   // ////////////////////////////////////////////////////////////////////
00449   void InventoryManager::
00450   calculateAvailabilityByProtectiveIBP (stdair::TravelSolutionStruct&
      ioTravelSolution) {
00451     std::ostringstream oStr;
```

```
00452
00453     // Yield valuation coefficient for multi-segment travel solutions
00454     double alpha = 1.0;
00455
00456     const stdair::SegmentPath_T& lSP = ioTravelSolution.getSegmentPath();
00457     for (stdair::SegmentPath_T::const_iterator itSP = lSP.begin();
00458          itSP != lSP.end(); itSP++) {
00459       oStr << *itSP << ";";
00460     }
00461
00462     //Retrieve bid price vector and yield maps
00463     const stdair::ClassYieldMapHolder_T& lClassYieldMapHolder =
00464       ioTravelSolution.getClassYieldMapHolder();
00465     const stdair::ClassBpvMapHolder_T& lClassBpvMapHolder =
00466       ioTravelSolution.getClassBpvMapHolder();
00467
00468     //Retrieve the list of fare options and browse it
00469     stdair::FareOptionList_T& lFOList = ioTravelSolution.getFareOptionListRef()
      ;
00470     for (stdair::FareOptionList_T::iterator itFO = lFOList.begin();
00471          itFO != lFOList.end(); ++itFO) {
00472
00473       stdair::FareOptionStruct& lFO = *itFO;
00474
00475       stdair::ClassYieldMapHolder_T::const_iterator itCYM =
00476         lClassYieldMapHolder.begin();
00477       stdair::ClassBpvMapHolder_T::const_iterator itCBPM =
00478         lClassBpvMapHolder.begin();
00479
00480       const stdair::ClassList_StringList_T& lClassPath = lFO.getClassPath();
00481
00482       // Sanity checks
00483       assert (lClassPath.size() == lClassYieldMapHolder.size());
00484       assert (lClassPath.size() == lClassBpvMapHolder.size());
00485
00486       // Yield is taken to be equal to fare (connecting flights)
00487       // TODO : take yield instead
00488       stdair::YieldValue_T lTotalYield = lFO.getFare();
00489       // Bid price initialisation
00490       stdair::BidPrice_T lTotalBidPrice = 0;
00491       // Maximal bid price initialisation
00492       stdair::BidPrice_T lMaxBidPrice = 0;
00493
00494       //Browse class path, class-yield maps, class-(bid price vector) maps.
00495       //Each iteration corresponds to one segment.
00496
00497       std::ostringstream oCPStr;
00498       for (stdair::ClassList_StringList_T::const_iterator itCL =
00499              lClassPath.begin();
00500            itCL != lClassPath.end(); ++itCL, ++itCYM, ++itCBPM) {
00501
00502         // Class path determination
00503         if (itCL == lClassPath.begin()) {
00504           oCPStr << *itCL;
00505
00506         } else {
00507           oCPStr << "-" << *itCL;
00508         }
00509
00510         const stdair::ClassList_String_T& lCL = *itCL;
00511         stdair::ClassCode_T lCC;
00512         lCC.append (lCL, 0, 1);
00513
00514         const stdair::ClassYieldMap_T& lCYM = *itCYM;
00515         stdair::ClassYieldMap_T::const_iterator itCCCYM = lCYM.find (lCC);
00516         assert (itCCCYM != lCYM.end());
00517
00518         const stdair::YieldValue_T& lYield = itCCCYM->second;
00519         const stdair::ClassBpvMap_T& lCBPM = *itCBPM;
00520         stdair::ClassBpvMap_T::const_iterator itCCCBPM = lCBPM.find (lCC);
00521         assert (itCCCBPM != lCBPM.end());
00522
00523         const stdair::BidPriceVector_T* lBidPriceVector_ptr = itCCCBPM->second;
00524         assert (lBidPriceVector_ptr != NULL);
00525
00526         // Initialization of fare option availability
00527         if (itCL == lClassPath.begin()) {
00528           lFO.setAvailability (lBidPriceVector_ptr->size());
00529         }
00530
00531         // Availability update
00532         if (lFO.getAvailability() > 0) {
00533
00534           //Segment availability calculation
00535           stdair::BidPriceVector_T lReverseBPV (lBidPriceVector_ptr->size());
00536           std::reverse_copy (lBidPriceVector_ptr->begin(),
00537                              lBidPriceVector_ptr->end(), lReverseBPV.begin());
```

```
00538
00539           stdair::BidPriceVector_T::const_iterator lBidPrice =
00540             std::upper_bound (lReverseBPV.begin(), lReverseBPV.end(), lYield);
00541
00542           const stdair::Availability_T lAvl = lBidPrice - lReverseBPV.begin();
00543
00544           // Availability update
00545           lFO.setAvailability (std::min(lFO.getAvailability(), lAvl));
00546
00547         }
00548
00549         // Total bid price calculation
00550         if (lBidPriceVector_ptr->size() > 0) {
00551           lTotalBidPrice += lBidPriceVector_ptr->back();
00552
00553           if (lMaxBidPrice < lBidPriceVector_ptr->back()) {
00554             lMaxBidPrice = lBidPriceVector_ptr->back();
00555           }
00556
00557         } else {
00558           lTotalBidPrice = std::numeric_limits<stdair::BidPrice_T>::max();
00559         }
00560
00561         // Total yield calculation (has been replaced by total fare).
00562         //lTotalYield += lYield;
00563       }
00564       // Multi-segment bid price control
00565
00566       // Protective IBP (maximin): guarantees the minimal yield for each
     airline
00567       // Proration factors are all equal to 1/{number of partners}.
00568
00569       lTotalBidPrice = std::max (lMaxBidPrice * lClassPath.size(),
00570                                  lTotalBidPrice);
00571
00572       if (lClassPath.size() > 1) {
00573         if (lFO.getAvailability() > 0) {
00574           const stdair::Availability_T lAvl =
00575             alpha * lTotalYield >= lTotalBidPrice;
00576           lFO.setAvailability (lAvl * lFO.getAvailability());
00577
00578         } else {
00579           const stdair::Availability_T lAvl =
00580             alpha * lTotalYield >= lTotalBidPrice;
00581           lFO.setAvailability (lAvl);
00582         }
00583
00584         // DEBUG
00585         STDAIR_LOG_DEBUG ("Class: " << oCPStr.str()
00586                           << ", " << "Yield: " << alpha*lTotalYield << ", "
00587                           << "Bid price: " << lTotalBidPrice << ", "
00588                           << "Remaining capacity: " << "Undefined" << " "
00589                           << "Segment date: " << oStr.str());
00590       }
00591
00592       // DEBUG
00593       STDAIR_LOG_DEBUG ("Fare option " << lFO.describe() << ", "
00594                         << "Availability " << lFO.getAvailability() << ", "
00595                         << "Segment Path " << oStr.str() << ", ");
00596     }
00597   }
00598
00599   //MODIF
00600   // ////////////////////////////////////////////////////////////////////
00601   void InventoryManager::setDefaultBidPriceVector
      (stdair::BomRoot& ioBomRoot) {
00602
00603     const stdair::InventoryList_T& lInvList =
00604       stdair::BomManager::getList<stdair::Inventory> (ioBomRoot);
00605     for (stdair::InventoryList_T::const_iterator itInv = lInvList.begin();
00606          itInv != lInvList.end(); ++itInv) {
00607       stdair::Inventory* lCurrentInv_ptr = *itInv;
00608       assert (lCurrentInv_ptr != NULL);
00609
00610       // Set the default bid price for own cabins.
00611       setDefaultBidPriceVector (*lCurrentInv_ptr);
00612
00613       // Check if the inventory contains images of partner inventories.
00614       // If so, set the default bid price for their cabins.
00615       if (stdair::BomManager::hasList<stdair::Inventory> (*lCurrentInv_ptr)) {
00616         const stdair::InventoryList_T& lPartnerInvList =
00617           stdair::BomManager::getList<stdair::Inventory> (*lCurrentInv_ptr);
00618
00619         for (stdair::InventoryList_T::const_iterator itPartnerInv =
00620                lPartnerInvList.begin();
00621              itPartnerInv != lPartnerInvList.end(); ++itPartnerInv) {
00622           stdair::Inventory* lCurrentPartnerInv_ptr = *itPartnerInv;
```

```
00623           assert (lCurrentPartnerInv_ptr != NULL);
00624
00625           setDefaultBidPriceVector (*
      lCurrentPartnerInv_ptr);
00626         }
00627       }
00628     }
00629   }
00630
00631   // ////////////////////////////////////////////////////////////////////
00632   void InventoryManager::
00633   setDefaultBidPriceVector (stdair::Inventory&
      ioInventory) {
00634
00635     const stdair::FlightDateList_T& lFlightDateList =
00636       stdair::BomManager::getList<stdair::FlightDate> (ioInventory);
00637     for (stdair::FlightDateList_T::const_iterator itFlightDate =
00638            lFlightDateList.begin();
00639          itFlightDate != lFlightDateList.end(); ++itFlightDate) {
00640       stdair::FlightDate* lCurrentFlightDate_ptr = *itFlightDate;
00641       assert (lCurrentFlightDate_ptr != NULL);
00642
00643       // Check if the flight date holds a list of leg dates.
00644       // If so retrieve it and initialise the bid price vectors of their
       cabins.
00645       if (stdair::BomManager::hasList<stdair::LegDate> (*lCurrentFlightDate_ptr
      )) {
00646         const stdair::LegDateList_T& lLegDateList =
00647           stdair::BomManager::getList<stdair::LegDate>
      (*lCurrentFlightDate_ptr);
00648         for (stdair::LegDateList_T::const_iterator itLegDate =
00649                lLegDateList.begin();
00650              itLegDate != lLegDateList.end(); ++itLegDate) {
00651           stdair::LegDate* lCurrentLegDate_ptr = *itLegDate;
00652           assert (lCurrentLegDate_ptr != NULL);
00653
00654           const stdair::LegCabinList_T& lLegCabinList =
00655             stdair::BomManager::getList<stdair::LegCabin>
      (*lCurrentLegDate_ptr);
00656           for (stdair::LegCabinList_T::const_iterator itLegCabin =
00657                  lLegCabinList.begin();
00658                itLegCabin != lLegCabinList.end(); ++itLegCabin) {
00659             stdair::LegCabin* lCurrentLegCabin_ptr = *itLegCabin;
00660             assert (lCurrentLegCabin_ptr != NULL);
00661
00662             const stdair::CabinCapacity_T& lCabinCapacity =
00663               lCurrentLegCabin_ptr->getPhysicalCapacity();
00664             lCurrentLegCabin_ptr->emptyBidPriceVector();
00665
00666             stdair::BidPriceVector_T& lBPV =
00667               lCurrentLegCabin_ptr->getBidPriceVector();
00668
00669             //for (stdair::CabinCapacity_T k = 0;k!=lCabinCapacity;k++)
       {lBPV.push_back(400 + 300/sqrt(k+1));}
00670             for (stdair::CabinCapacity_T k = 0; k != lCabinCapacity; k++) {
00671               lBPV.push_back (400);
00672             }
00673
00674             lCurrentLegCabin_ptr->setPreviousBidPrice (lBPV.back());
00675             lCurrentLegCabin_ptr->setCurrentBidPrice (lBPV.back());
00676           }
00677         }
00678       }
00679     }
00680   }
00681
00682   // ////////////////////////////////////////////////////////////////////
00683   bool InventoryManager::sell (stdair::Inventory& ioInventory,
00684                                const std::string& iSegmentDateKey,
00685                                const stdair::ClassCode_T& iClassCode,
00686                                const stdair::PartySize_T& iPartySize) {
00687
00688     // Make the sale within the inventory.
00689     return InventoryHelper::sell (ioInventory,
      iSegmentDateKey,
00690                                   iClassCode, iPartySize);
00691   }
00692
00693   // ////////////////////////////////////////////////////////////////////
00694   bool InventoryManager::sell (const stdair::BookingClassID_T& iClassID,
00695                                const stdair::PartySize_T& iPartySize) {
00696
00697     // Make the sale within the inventory.
00698     return InventoryHelper::sell (iClassID, iPartySize);
00699   }
00700
00701   // ////////////////////////////////////////////////////////////////////
```

```
00702   bool InventoryManager::cancel (stdair::Inventory& ioInventory,
00703                                  const std::string& iSegmentDateKey,
00704                                  const stdair::ClassCode_T& iClassCode,
00705                                  const stdair::PartySize_T& iPartySize) {
00706
00707     // Make the cancellation within the inventory.
00708     return InventoryHelper::cancel (ioInventory,
      iSegmentDateKey,
00709                                     iClassCode, iPartySize);
00710   }
00711
00712   // ////////////////////////////////////////////////////////////////////
00713   bool InventoryManager::cancel (const stdair::BookingClassID_T& iClassID,
00714                                  const stdair::PartySize_T& iPartySize) {
00715
00716     // Make the cancellation within the inventory.
00717     return InventoryHelper::cancel (iClassID, iPartySize
      );
00718   }
00719
00720   // ////////////////////////////////////////////////////////////////////
00721   void InventoryManager::
00722   updateBookingControls (stdair::FlightDate& ioFlightDate) {
00723
00724     // Forward the call to FlightDateHelper.
00725     FlightDateHelper::updateBookingControls
       (ioFlightDate);
00726   }
00727
00728   // ////////////////////////////////////////////////////////////////////
00729   void InventoryManager::
00730   recalculateAvailability (stdair::FlightDate& ioFlightDate) {
00731
00732     // Forward the call to FlightDateHelper.
00733     FlightDateHelper::recalculateAvailability
       (ioFlightDate);
00734   }
00735
00736   // ////////////////////////////////////////////////////////////////////
00737   void InventoryManager::takeSnapshots(const stdair::Inventory& iInventory,
00738                                        const stdair::DateTime_T& iSnapshotTime)
      {
00739
00740     // Make the snapshots within the inventory
00741     InventoryHelper::takeSnapshots (iInventory,
      iSnapshotTime);
00742   }
00743
00744   // ////////////////////////////////////////////////////////////////////
00745   void InventoryManager::
00746   createDirectAccesses (const stdair::BomRoot& iBomRoot)
      {
00747
00748     // Browse the list of inventories and create direct accesses
00749     // within each inventory.
00750     const stdair::InventoryList_T& lInvList =
00751       stdair::BomManager::getList<stdair::Inventory> (iBomRoot);
00752     for (stdair::InventoryList_T::const_iterator itInv = lInvList.begin();
00753          itInv != lInvList.end(); ++itInv) {
00754       stdair::Inventory* lCurrentInv_ptr = *itInv;
00755       assert (lCurrentInv_ptr != NULL);
00756
00757       createDirectAccesses (iBomRoot, *lCurrentInv_ptr);
00758     }
00759
00760     // Browse the list of inventories and create partner accesses
00761     // within each inventory.
00762     for (stdair::InventoryList_T::const_iterator itInv = lInvList.begin();
00763          itInv != lInvList.end(); ++itInv) {
00764       stdair::Inventory* lCurrentInv_ptr = *itInv;
00765       assert (lCurrentInv_ptr != NULL);
00766
00767       createPartnerAccesses (iBomRoot, *lCurrentInv_ptr);
00768     }
00769
00770     // Fill some attributes of segment-date with the routing legs.
00771     BomRootHelper::fillFromRouting (iBomRoot);
00772   }
00773
00774   // ////////////////////////////////////////////////////////////////////
00775   void InventoryManager::
00776   createDirectAccesses (const stdair::BomRoot& iBomRoot,
00777                         stdair::Inventory& ioInventory) {
00778
00779     // Browse the list of flight-dates and create direct accesses
00780     // within each flight-date.
00781     const stdair::FlightDateList_T& lFlightDateList =
```

```
00782       stdair::BomManager::getList<stdair::FlightDate> (ioInventory);
00783     for (stdair::FlightDateList_T::const_iterator itFlightDate =
00784            lFlightDateList.begin();
00785          itFlightDate != lFlightDateList.end(); ++itFlightDate) {
00786       stdair::FlightDate* lCurrentFlightDate_ptr = *itFlightDate;
00787       assert (lCurrentFlightDate_ptr != NULL);
00788
00789       createDirectAccesses (iBomRoot, ioInventory, *
      lCurrentFlightDate_ptr);
00790     }
00791
00792     // Browse the list of inventories and create direct accesses
00793     // within each inventory.
00794     const bool hasInventoryList =
00795       stdair::BomManager::hasList<stdair::Inventory> (ioInventory);
00796     if (hasInventoryList == true) {
00797       const stdair::InventoryList_T& lInvList =
00798         stdair::BomManager::getList<stdair::Inventory> (ioInventory);
00799       for (stdair::InventoryList_T::const_iterator itInv = lInvList.begin();
00800            itInv != lInvList.end(); ++itInv) {
00801         stdair::Inventory* lCurrentInv_ptr = *itInv;
00802         assert (lCurrentInv_ptr != NULL);
00803
00804         createDirectAccesses (iBomRoot, *lCurrentInv_ptr);
00805       }
00806     }
00807   }
00808
00809   // ////////////////////////////////////////////////////////////////////
00810   void InventoryManager::
00811   createDirectAccesses (const stdair::BomRoot& ioBomRoot,
00812                         stdair::Inventory& ioInventory,
00813                         stdair::FlightDate& ioFlightDate) {
00814
00815     bool areSegmentAndRoutingLegLinked = false;
00816
00817     // Browse the list of segment-dates and create direct accesses
00818     // within each segment-date.
00819     const stdair::SegmentDateList_T& lSegmentDateList =
00820       stdair::BomManager::getList<stdair::SegmentDate> (ioFlightDate);
00821     for (stdair::SegmentDateList_T::const_iterator itSegmentDate =
00822            lSegmentDateList.begin();
00823          itSegmentDate != lSegmentDateList.end(); ++itSegmentDate) {
00824
00825       stdair::SegmentDate* lCurrentSegmentDate_ptr = *itSegmentDate;
00826       assert (lCurrentSegmentDate_ptr != NULL);
00827
00828       // Get the routing leg keys list
00829       const stdair::RoutingLegKeyList_T& lRoutingLegKeyList =
00830         lCurrentSegmentDate_ptr->getLegKeyList ();
00831
00832       // Browse the list of routing leg keys and try to create direct accesses
00833       // with each corresponding leg date.
00834       for (stdair::RoutingLegKeyList_T::const_iterator itRoutingLegKey =
00835            lRoutingLegKeyList.begin();
00836          itRoutingLegKey != lRoutingLegKeyList.end(); ++itRoutingLegKey) {
00837
00838         // Try to retrieve the routing LegDate within the flight date
00839         stdair::LegDate* lLegDate_ptr = stdair::BomRetriever::
00840           retrieveOperatingLegDateFromLongKey (ioFlightDate, *itRoutingLegKey);
00841
00842         if (lLegDate_ptr != NULL) {
00843
00844           // Link the SegmentDate and LegDate together
00845           stdair::FacBomManager::addToListAndMap (*lCurrentSegmentDate_ptr,
00846                                                   *lLegDate_ptr);
00847           stdair::FacBomManager::addToListAndMap (*lLegDate_ptr,
00848                                                   *lCurrentSegmentDate_ptr);
00849           areSegmentAndRoutingLegLinked = true;
00850         }
00851       }
00852       if (areSegmentAndRoutingLegLinked == true) {
00853         createDirectAccesses (*lCurrentSegmentDate_ptr);
00854       }
00855     }
00856   }
00857
00858   // ////////////////////////////////////////////////////////////////////
00859   void InventoryManager::
00860   createDirectAccesses (stdair::SegmentDate&
      ioSegmentDate) {
00861
00862     // Browse the list of segment-cabins and create direct accesses
00863     // within each segment-cabin.
00864     const stdair::SegmentCabinList_T& lSegmentCabinList =
00865       stdair::BomManager::getList<stdair::SegmentCabin> (ioSegmentDate);
00866     for (stdair::SegmentCabinList_T::const_iterator itSegmentCabin =
```

```
00867           lSegmentCabinList.begin();
00868         itSegmentCabin != lSegmentCabinList.end(); ++itSegmentCabin) {
00869
00870       //
00871       stdair::SegmentCabin* lCurrentSegmentCabin_ptr = *itSegmentCabin;
00872       assert (lCurrentSegmentCabin_ptr != NULL);
00873
00874       //
00875       const stdair::CabinCode_T& lCabinCode =
00876           lCurrentSegmentCabin_ptr->getCabinCode();
00877
00878       // Iterate on the routing legs
00879       const stdair::LegDateList_T& lLegDateList =
00880         stdair::BomManager::getList<stdair::LegDate> (ioSegmentDate);
00881       for (stdair::LegDateList_T::const_iterator itLegDate =
00882              lLegDateList.begin();
00883            itLegDate != lLegDateList.end(); ++itLegDate) {
00884
00885         const stdair::LegDate* lCurrentLegDate_ptr = *itLegDate;
00886         assert (lCurrentLegDate_ptr != NULL);
00887
00888         // Retrieve the LegCabin getting the same class of service
00889         // (cabin code) as the SegmentCabin.
00890         stdair::LegCabin& lLegCabin = stdair::BomManager::
00891           getObject<stdair::LegCabin> (*lCurrentLegDate_ptr, lCabinCode);
00892
00903         stdair::FacBomManager::addToListAndMap (*lCurrentSegmentCabin_ptr,
00904                                                 lLegCabin,
00905                                                 lLegCabin.getFullerKey());
00906
00917         stdair::FacBomManager::
00918           addToListAndMap (lLegCabin, *lCurrentSegmentCabin_ptr,
00919                            lCurrentSegmentCabin_ptr->getFullerKey());
00920       }
00921     }
00922   }
00923
00924   // ////////////////////////////////////////////////////////////////////
00925   void InventoryManager::
00926   createPartnerAccesses (const stdair::BomRoot& iBomRoot
      ,
00927                          stdair::Inventory& ioInventory) {
00928
00929     // Browse the list of flight-dates and create partner accesses
00930     // within each flight-date.
00931     const stdair::FlightDateList_T& lFlightDateList =
00932       stdair::BomManager::getList<stdair::FlightDate> (ioInventory);
00933     for (stdair::FlightDateList_T::const_iterator itFlightDate =
00934            lFlightDateList.begin();
00935          itFlightDate != lFlightDateList.end(); ++itFlightDate) {
00936       stdair::FlightDate* lCurrentFlightDate_ptr = *itFlightDate;
00937       assert (lCurrentFlightDate_ptr != NULL);
00938
00939       createPartnerAccesses (iBomRoot, ioInventory, *
      lCurrentFlightDate_ptr);
00940     }
00941   }
00942
00943   // ////////////////////////////////////////////////////////////////////
00944   void InventoryManager::
00945   createPartnerAccesses (const stdair::BomRoot&
      ioBomRoot,
00946                          stdair::Inventory& ioInventory,
00947                          stdair::FlightDate& ioFlightDate) {
00948
00949     // Browse the list of segment-dates and create partner accesses
00950     // within each segment-date.
00951     const stdair::SegmentDateList_T& lSegmentDateList =
00952       stdair::BomManager::getList<stdair::SegmentDate> (ioFlightDate);
00953     for (stdair::SegmentDateList_T::const_iterator itSegmentDate =
00954            lSegmentDateList.begin();
00955          itSegmentDate != lSegmentDateList.end(); ++itSegmentDate) {
00956
00957       stdair::SegmentDate* lCurrentSegmentDate_ptr = *itSegmentDate;
00958       assert (lCurrentSegmentDate_ptr != NULL);
00959
00960       // Get the routing leg keys list
00961       const stdair::RoutingLegKeyList_T& lRoutingLegKeyList =
00962         lCurrentSegmentDate_ptr->getLegKeyList ();
00963
00964       // Browse the list of routing leg keys and try to create partner accesses
00965       // with each corresponding leg date.
00966       for (stdair::RoutingLegKeyList_T::const_iterator itRoutingLegKey =
00967            lRoutingLegKeyList.begin();
00968          itRoutingLegKey != lRoutingLegKeyList.end(); ++itRoutingLegKey) {
00969
00970         // Try to retrieve the LegDate getting that Boarding Point within the
```

```
00971         // flight date
00972         stdair::LegDate* lLegDate_ptr = stdair::BomRetriever::
00973           retrieveOperatingLegDateFromLongKey (ioFlightDate, *itRoutingLegKey);
00974
00975         // If there is no LegDate getting that Boarding Point within the flight
00976         // date, the segment is operating by a partner
00977         if (lLegDate_ptr == NULL) {
00978
00979           // Retrieve the (unique) operating LegDate getting that Boarding
       Point
00980           // within the partner inventory
00981           std::ostringstream lRoutingSegmentKey;
00982           lRoutingSegmentKey << *itRoutingLegKey << ";"
00983                              << lCurrentSegmentDate_ptr->getOffPoint();
00984
00985           stdair::SegmentDate* lPartnerOperatingSegmentDate_ptr =
00986             stdair::BomRetriever::
00987             retrievePartnerSegmentDateFromLongKey (ioInventory,
00988                                                    lRoutingSegmentKey.str());
00989           assert (lPartnerOperatingSegmentDate_ptr != NULL);
00990
00991           // Link the current segment date with its operating one
00992           stdair::FacBomManager::linkWithOperating (*lCurrentSegmentDate_ptr,
00993                                                     *
      lPartnerOperatingSegmentDate_ptr);
00994
00995           stdair::SegmentDate* lOperatingSegmentDate_ptr =
00996             stdair::BomRetriever::
00997             retrieveSegmentDateFromLongKey (ioBomRoot,
00998                                             lRoutingSegmentKey.str());
00999           assert(lOperatingSegmentDate_ptr != NULL);
01000
01001           stdair::Inventory* lInventory_ptr =
01002             stdair::BomRetriever::
01003             retrieveInventoryFromLongKey (ioBomRoot, *itRoutingLegKey);
01004           assert (lInventory_ptr != NULL);
01005
01006           std::ostringstream lRoutingSegment;
01007           lRoutingSegment << ioFlightDate.getAirlineCode() << ";"
01008                           << ioFlightDate.describeKey() << ";"
01009                           << lCurrentSegmentDate_ptr->getBoardingPoint() << ";"
01010                           << lCurrentSegmentDate_ptr->getOffPoint();
01011
01012           stdair::SegmentDate* lPartnerMarketingSegmentDate_ptr =
01013             stdair::BomRetriever::
01014             retrievePartnerSegmentDateFromLongKey (*lInventory_ptr,
      lRoutingSegment.str());
01015           assert(lPartnerMarketingSegmentDate_ptr  != NULL);
01016
01017           stdair::FacBomManager::addToList (*lOperatingSegmentDate_ptr, *
      lPartnerMarketingSegmentDate_ptr);
01018
01019         }
01020       }
01021     }
01022   }
01023
01024
01025   // ////////////////////////////////////////////////////////////////////
01026   void InventoryManager::
01027   buildSimilarSegmentCabinSets (const
      stdair::BomRoot& iBomRoot) {
01028     // Browse the list of inventories and create direct accesses
01029     // within each inventory.
01030     const stdair::InventoryList_T& lInvList =
01031       stdair::BomManager::getList<stdair::Inventory> (iBomRoot);
01032     for (stdair::InventoryList_T::const_iterator itInv = lInvList.begin();
01033          itInv != lInvList.end(); ++itInv) {
01034       stdair::Inventory* lCurrentInv_ptr = *itInv;
01035       assert (lCurrentInv_ptr != NULL);
01036
01037       buildSimilarSegmentCabinSets (*
      lCurrentInv_ptr);
01038     }
01039   }
01040
01041   // ////////////////////////////////////////////////////////////////////
01042   void InventoryManager::
01043   buildSimilarSegmentCabinSets (stdair::Inventory
      & ioInventory) {
01044     // For instance, we consider two flight-dates are
01045     // similar if they have the same flight number and the same
01046     // day-of-the-week departure.
01047
01048     // Browse the segment-cabin list and build the sets of segment-cabin
01049     // which have the same flight number and origin-destination
01050     SimilarSegmentCabinSetMap_T lSSCSM;
```

```
01051
01052     // Browsing the flight-date list
01053     const stdair::FlightDateList_T& lFlightDateList =
01054       stdair::BomManager::getList<stdair::FlightDate> (ioInventory);
01055     for (stdair::FlightDateList_T::const_iterator itFD= lFlightDateList.begin()
      ;
01056          itFD != lFlightDateList.end(); ++itFD) {
01057       const stdair::FlightDate* lFD_ptr = *itFD;
01058       assert (lFD_ptr != NULL);
01059       const stdair::FlightNumber_T& lFlightNumber = lFD_ptr->getFlightNumber();
01060
01061       // Browsing the segment-date list and retrieve the departure
01062       // date, the origine and the destination of the segment
01063       const stdair::SegmentDateList_T& lSegmentDateList =
01064         stdair::BomManager::getList<stdair::SegmentDate> (*lFD_ptr);
01065       for (stdair::SegmentDateList_T::const_iterator itSD =
01066              lSegmentDateList.begin(); itSD != lSegmentDateList.end(); ++itSD)
      {
01067         const stdair::SegmentDate* lSD_ptr = *itSD;
01068         assert (lSD_ptr != NULL);
01069
01070         const stdair::Date_T& lDepartureDate = lSD_ptr->getBoardingDate();
01071         const stdair::AirportCode_T& lOrigin = lSD_ptr->getBoardingPoint();
01072         const stdair::AirportCode_T& lDestination = lSD_ptr->getOffPoint();
01073
01074         // Browsing the segment-cabin list and retrieve the cabin code and
01075         // build the corresponding key map.
01076         const stdair::SegmentCabinList_T& lSegmentCabinList =
01077           stdair::BomManager::getList<stdair::SegmentCabin> (*lSD_ptr);
01078         for (stdair::SegmentCabinList_T::const_iterator itSC =
01079                lSegmentCabinList.begin();
01080              itSC != lSegmentCabinList.end(); ++itSC) {
01081           stdair::SegmentCabin* lSC_ptr = *itSC;
01082           assert (lSC_ptr != NULL);
01083
01084           std::ostringstream oStr;
01085           oStr << lFlightNumber << lDepartureDate.day_of_week()
01086                << lOrigin << lDestination << lSC_ptr->getCabinCode();
01087           const std::string lMapKey = oStr.str();
01088
01089           // Add the segment cabin to the similar segment cabin set map.
01090           SimilarSegmentCabinSetMap_T::iterator itSSCS = lSSCSM.find (lMapKey);
01091           if (itSSCS == lSSCSM.end()) {
01092             DepartureDateSegmentCabinMap_T
      lDDSCMap;
01093             lDDSCMap.insert (DepartureDateSegmentCabinMap_T
      ::
01094                              value_type (lDepartureDate, lSC_ptr));
01095             lSSCSM.insert (SimilarSegmentCabinSetMap_T
      ::
01096                            value_type (lMapKey, lDDSCMap));
01097           } else {
01098             DepartureDateSegmentCabinMap_T&
      lDDSCMap = itSSCS->second;
01099             lDDSCMap.insert (DepartureDateSegmentCabinMap_T
      ::
01100                              value_type (lDepartureDate, lSC_ptr));
01101           }
01102         }
01103       }
01104     }
01105
01106     // Initialise the segment data table.
01107     stdair::TableID_T lTableID = 1;
01108     for (SimilarSegmentCabinSetMap_T::const_iterator itSSCS = lSSCSM.begin();
01109          itSSCS != lSSCSM.end(); ++itSSCS, ++lTableID) {
01110       const DepartureDateSegmentCabinMap_T&
      lDDSCMap = itSSCS->second;
01111
01112       buildSegmentSnapshotTable (ioInventory, lTableID
      , lDDSCMap);
01113     }
01114   }
01115
01116   // ////////////////////////////////////////////////////////////////////
01117   void InventoryManager::
01118   buildSegmentSnapshotTable (stdair::Inventory&
      ioInventory,
01119                              const stdair::TableID_T& iTableID,
01120                              const DepartureDateSegmentCabinMap_T
      & iDDSCMap) {
01121     // Build an empty segment data table.
01122     const stdair::SegmentSnapshotTableKey lKey (iTableID);
01123     stdair::SegmentSnapshotTable& lSegmentSnapshotTable =
01124       stdair::FacBom<stdair::SegmentSnapshotTable>::instance().create (lKey);
01125     stdair::FacBomManager::addToListAndMap (ioInventory, lSegmentSnapshotTable)
      ;
```

```
01126
01127     // Build the value type index map.
01128     DepartureDateSegmentCabinMap_T::const_iterator itDDSC = iDDSCMap.begin();
01129     assert (itDDSC != iDDSCMap.end());
01130     const stdair::SegmentCabin* lSegmentCabin_ptr = itDDSC->second;
01131
01132     // Browse the booking class list and build the value type for the classes
01133     // as well as for the cabin (Q-equivalent).
01134     stdair::ClassIndexMap_T lClassIndexMap;
01135     stdair::ClassIndex_T lClassIndex = 0;
01136     std::ostringstream lSCMapKey;
01137     lSCMapKey << stdair::DEFAULT_SEGMENT_CABIN_VALUE_TYPE
01138               << lSegmentCabin_ptr->describeKey();
01139     lClassIndexMap.insert (stdair::ClassIndexMap_T::
01140                            value_type (lSCMapKey.str(), lClassIndex));
01141     ++lClassIndex;
01142
01143     // Browse the booking class list
01144     const stdair::BookingClassList_T& lBCList =
01145       stdair::BomManager::getList<stdair::BookingClass>(*lSegmentCabin_ptr);
01146     for (stdair::BookingClassList_T::const_iterator itBC= lBCList.begin();
01147          itBC != lBCList.end(); ++itBC) {
01148       const stdair::BookingClass* lBookingClass_ptr = *itBC;
01149       assert (lBookingClass_ptr != NULL);
01150       lClassIndexMap.
01151         insert (stdair::ClassIndexMap_T::
01152                 value_type(lBookingClass_ptr->describeKey(),lClassIndex));
01153       ++lClassIndex;
01154     }
01155
01156     // Build the segment-cabin index map
01157     stdair::SegmentCabinIndexMap_T lSegmentCabinIndexMap;
01158     stdair::SegmentDataID_T lSegmentDataID = 0;
01159     for (; itDDSC != iDDSCMap.end(); ++itDDSC, ++lSegmentDataID) {
01160       stdair::SegmentCabin* lCurrentSC_ptr = itDDSC->second;
01161       assert (lCurrentSC_ptr != NULL);
01162       lSegmentCabinIndexMap.
01163         insert (stdair::SegmentCabinIndexMap_T::value_type (lCurrentSC_ptr,
01164                                                             lSegmentDataID));
01165
01166       // Added the data table to the segment-cabin.
01167       lCurrentSC_ptr->setSegmentSnapshotTable (lSegmentSnapshotTable);
01168     }
01169
01170     // Initialise the segment data table.
01171     lSegmentSnapshotTable.initSnapshotBlocks(lSegmentCabinIndexMap,
01172                                              lClassIndexMap);
01173   }
01174
01175   // ////////////////////////////////////////////////////////////////////
01176   void InventoryManager::initSnapshotEvents (SEVMGR::SEVMGR_ServicePtr_T
      ioSEVMGR_ServicePtr,
01177                                              const stdair::Date_T& iStartDate,
01178                                              const stdair::Date_T& iEndDate) {
01179     const stdair::Duration_T lTimeZero (0, 0, 0);
01180     const stdair::Duration_T lOneDayDuration (24, 0, 0);
01181     const stdair::DateTime_T lBeginSnapshotTime (iStartDate, lTimeZero);
01182     const stdair::DateTime_T lEndSnapshotTime (iEndDate, lTimeZero);
01183
01184     // TODO: remove the defaut airline code.
01185     stdair::NbOfEvents_T lNbOfSnapshots = 0.0;
01186     for (stdair::DateTime_T lSnapshotTime = lBeginSnapshotTime;
01187          lSnapshotTime < lEndSnapshotTime; lSnapshotTime += lOneDayDuration) {
01188       // Create the snapshot event structure
01189       stdair::SnapshotPtr_T lSnapshotStruct =
01190         boost::make_shared<stdair::SnapshotStruct>(stdair::DEFAULT_AIRLINE_CODE
      ,
01191                                                    lSnapshotTime);
01192       // Create the event structure
01193       stdair::EventStruct lEventStruct (stdair::EventType::SNAPSHOT,
01194                                         lSnapshotStruct);
01195
01203     ioSEVMGR_ServicePtr->addEvent (lEventStruct);
01204     ++lNbOfSnapshots;
01205     }
01206
01207     // Update the status of snap shots within the event queue.
01208     ioSEVMGR_ServicePtr->addStatus (stdair::EventType::SNAPSHOT, lNbOfSnapshots
      );
01209   }
01210
01211   // ////////////////////////////////////////////////////////////////////
01212   void InventoryManager::
01213   initRMEvents (const stdair::Inventory& iInventory,
01214                 stdair::RMEventList_T& ioRMEventList,
01215                 const stdair::Date_T& iStartDate,
01216                 const stdair::Date_T& iEndDate) {
```

```
01217     const stdair::Duration_T lTimeZero (0, 0, 0);
01218     const stdair::Duration_T lTime (0, 0, 10);
01219     const stdair::Duration_T lOneDayDuration (24, 0, 0);
01220     const stdair::DateTime_T lEarliestEventTime (iStartDate, lTimeZero);
01221     const stdair::DateTime_T lLatestEventTime (iEndDate, lTimeZero);
01222
01223     const stdair::AirlineCode_T& lAirlineCode = iInventory.getAirlineCode();
01224
01225     // Browse the list of flight-dates and initialise the RM events for
01226     // each flight-date.
01227     const stdair::FlightDateList_T& lFDList =
01228       stdair::BomManager::getList<stdair::FlightDate> (iInventory);
01229     for (stdair::FlightDateList_T::const_iterator itFD = lFDList.begin();
01230          itFD != lFDList.end(); ++itFD) {
01231       const stdair::FlightDate* lFD_ptr = *itFD;
01232       assert (lFD_ptr != NULL);
01233
01234       // Retrive the departure date and initialise the RM events with
01235       // the data collection points of the inventory.
01236       const stdair::Date_T& lDepartureDate = lFD_ptr->getDepartureDate();
01237       const stdair::DateTime_T lDepatureDateTime (lDepartureDate, lTime);
01238       for (stdair::DCPList_T::const_iterator itDCP =
01239              stdair::DEFAULT_DCP_LIST.begin();
01240            itDCP != stdair::DEFAULT_DCP_LIST.end(); ++itDCP) {
01241         const stdair::DCP_T& lDCP = *itDCP;
01242
01243         // Create the event time and check if it is in the validate interval
01244         const stdair::DateTime_T lEventTime =
01245           lDepatureDateTime - lOneDayDuration * lDCP;
01246         if (lEventTime >= lEarliestEventTime && lEventTime <= lLatestEventTime)
     {
01247           const stdair::KeyDescription_T lKeyDes = lFD_ptr->describeKey();
01248           stdair::RMEventStruct lRMEvent (lAirlineCode, lKeyDes, lEventTime);
01249           ioRMEventList.push_back (lRMEvent);
01250         }
01251       }
01252     }
01253   }
01254
01255   // ////////////////////////////////////////////////////////////////////
01256   void InventoryManager::
01257   addRMEventsToEventQueue (SEVMGR::SEVMGR_ServicePtr_T ioSEVMGR_ServicePtr,
01258                            stdair::RMEventList_T& ioRMEventList) {
01259     // Browse the RM event list and add them to the queue.
01260     for (stdair::RMEventList_T::iterator itRMEvent = ioRMEventList.begin();
01261          itRMEvent != ioRMEventList.end(); ++itRMEvent) {
01262       stdair::RMEventStruct& lRMEvent = *itRMEvent;
01263       stdair::RMEventPtr_T lRMEventPtr =
01264         boost::make_shared<stdair::RMEventStruct> (lRMEvent);
01265       stdair::EventStruct lEventStruct (stdair::EventType::RM, lRMEventPtr);
01266       ioSEVMGR_ServicePtr->addEvent (lEventStruct);
01267     }
01268
01269     // Update the status of RM events within the event queue.
01270     const stdair::Count_T lRMEventListSize = ioRMEventList.size();
01271     ioSEVMGR_ServicePtr->addStatus (stdair::EventType::RM, lRMEventListSize);
01272
01273   }
01274
01275   // ////////////////////////////////////////////////////////////////////
01276   void InventoryManager::
01277   initialiseYieldBasedNestingStructures
      (const stdair::BomRoot& iBomRoot) {
01278     // DEBUG
01279     STDAIR_LOG_DEBUG ("Initialise the yield-based nesting structure for "
01280                       << "all segment-cabins");
01281
01282     // Browse the list of inventories
01283     const stdair::InventoryList_T& lInvList =
01284       stdair::BomManager::getList<stdair::Inventory> (iBomRoot);
01285
01286     // Browse the inventories
01287     for (stdair::InventoryList_T::const_iterator itInv = lInvList.begin();
01288          itInv != lInvList.end(); ++itInv) {
01289       stdair::Inventory* lCurrentInv_ptr = *itInv;
01290       assert (lCurrentInv_ptr != NULL);
01291       const stdair::FlightDateList_T& lFlightDateList =
01292         stdair::BomManager::getList<stdair::FlightDate> (*lCurrentInv_ptr);
01293
01294       // Browse the flight dates
01295       for (stdair::FlightDateList_T::const_iterator itFD =
01296              lFlightDateList.begin(); itFD != lFlightDateList.end(); ++itFD) {
01297         const stdair::FlightDate* lFD_ptr = *itFD;
01298         assert (lFD_ptr != NULL);
01299         const stdair::SegmentDateList_T& lSegmentDateList =
01300           stdair::BomManager::getList<stdair::SegmentDate> (*lFD_ptr);
01301
```

```
01302         // Browse the segment dates
01303         for (stdair::SegmentDateList_T::const_iterator itSD =
01304              lSegmentDateList.begin(); itSD != lSegmentDateList.end(); ++itSD)
      {
01305           const stdair::SegmentDate* lSD_ptr = *itSD;
01306           assert (lSD_ptr != NULL);
01307           const stdair::SegmentCabinList_T& lSegmentCabinList =
01308             stdair::BomManager::getList<stdair::SegmentCabin> (*lSD_ptr);
01309
01310           // Browse the segment cabins
01311           for (stdair::SegmentCabinList_T::const_iterator itSC =
01312                lSegmentCabinList.begin(); itSC != lSegmentCabinList.end();
01313                ++itSC) {
01314             stdair::SegmentCabin* lSC_ptr = *itSC;
01315             assert (lSC_ptr != NULL);
01316
01317             // Initialise the nesting structure of the segment cabin
01318             SegmentCabinHelper::initYieldBasedNestingStructure
       (*lSC_ptr);
01319           }
01320         }
01321       }
01322     }
01323   }
01324
01325   // ////////////////////////////////////////////////////////////////////
01326   void InventoryManager::
01327   initialiseListsOfUsablePolicies (const
      stdair::BomRoot& iBomRoot) {
01328     // Browse the list of inventories
01329     const stdair::InventoryList_T& lInvList =
01330       stdair::BomManager::getList<stdair::Inventory> (iBomRoot);
01331
01332     // Browse the inventories
01333     for (stdair::InventoryList_T::const_iterator itInv = lInvList.begin();
01334          itInv != lInvList.end(); ++itInv) {
01335       stdair::Inventory* lCurrentInv_ptr = *itInv;
01336       assert (lCurrentInv_ptr != NULL);
01337
01338       // Create the policies if the optimisation method uses Marginal
01339       // Revenue Transformation
01340       stdair::PreOptimisationMethod::EN_PreOptimisationMethod lPreOptMethod =
01341         lCurrentInv_ptr->getPreOptimisationMethod();
01342
01343       if (lPreOptMethod == stdair::PreOptimisationMethod::MRT) {
01344         const stdair::FlightDateList_T& lFlightDateList =
01345           stdair::BomManager::getList<stdair::FlightDate> (*lCurrentInv_ptr);
01346
01347         // Browse the flight dates
01348         for (stdair::FlightDateList_T::const_iterator itFD =
01349                lFlightDateList.begin(); itFD != lFlightDateList.end(); ++itFD)
      {
01350           const stdair::FlightDate* lFD_ptr = *itFD;
01351           assert (lFD_ptr != NULL);
01352           const stdair::SegmentDateList_T& lSegmentDateList =
01353             stdair::BomManager::getList<stdair::SegmentDate> (*lFD_ptr);
01354
01355           // Browse the segment dates
01356           for (stdair::SegmentDateList_T::const_iterator itSD =
01357                  lSegmentDateList.begin();
01358                itSD != lSegmentDateList.end(); ++itSD) {
01359             const stdair::SegmentDate* lSD_ptr = *itSD;
01360             assert (lSD_ptr != NULL);
01361             const stdair::SegmentCabinList_T& lSegmentCabinList =
01362               stdair::BomManager::getList<stdair::SegmentCabin> (*lSD_ptr);
01363
01364             // Browse the segment cabins
01365             for (stdair::SegmentCabinList_T::const_iterator itSC =
01366                    lSegmentCabinList.begin(); itSC != lSegmentCabinList.end();
01367                  ++itSC) {
01368               stdair::SegmentCabin* lSC_ptr = *itSC;
01369               assert (lSC_ptr != NULL);
01370
01371               if (lSC_ptr->getFareFamilyStatus() == true) {
01372                 // Initialise the nesting structure of the segment cabin
01373                 SegmentCabinHelper::initListOfUsablePolicies
       (*lSC_ptr);
01374               }
01375             }
01376           }
01377         }
01378       }
01379     }
01380   }
01381
01382 }
```

23.143 airinv/command/InventoryManager.hpp File Reference

```
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/RMEventTypes.hpp>
#include <stdair/bom/BomIDTypes.hpp>
#include <sevmgr/SEVMGR_Types.hpp>
```

Classes

- class AIRINV::InventoryManager

Namespaces

- namespace stdair

  Forward declarations.
- namespace AIRINV

Typedefs

- typedef std::map< const stdair::Date_T, stdair::SegmentCabin * > AIRINV::DepartureDateSegmentCabinMap_T
- typedef std::map< const std::string, DepartureDateSegmentCabinMap_T > AIRINV::SimilarSegmentCabinSetMap_T

23.144 InventoryManager.hpp

```
00001 #ifndef __AIRINV_CMD_INVENTORYMANAGER_HPP
00002 #define __AIRINV_CMD_INVENTORYMANAGER_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // STDAIR
00010 #include <stdair/stdair_basic_types.hpp>
00011 #include <stdair/bom/RMEventTypes.hpp>
00012 #include <stdair/bom/BomIDTypes.hpp>
00013 // SEvMgr
00014 #include <sevmgr/SEVMGR_Types.hpp>
00015
00016 // Forward declarations
00017 namespace stdair {
00018   class BomRoot;
00019   class Inventory;
00020   class FlightDate;
00021   class SegmentDate;
00022   class SegmentCabin;
00023   class EventQueue;
00024   struct TravelSolutionStruct;
00025 }
00026
00027 namespace AIRINV {
00028
00029   // ///////////////// Type definitions ////////////////
00030   typedef std::map<const stdair::Date_T,
00031                    stdair::SegmentCabin*> DepartureDateSegmentCabinMap_T
      ;
00032   typedef std::map<const std::string,
00033                    DepartureDateSegmentCabinMap_T
      > SimilarSegmentCabinSetMap_T;
00034
00036   class InventoryManager {
00037     friend class AIRINV_Master_Service;
```

```
00038     friend class AIRINV_Service;
00039
00040   private:
00042     static void initSnapshotEvents (SEVMGR::SEVMGR_ServicePtr_T,
00043                                     const stdair::Date_T&,
00044                                     const stdair::Date_T&);
00045
00047     static void initRMEvents (const stdair::Inventory&, stdair::RMEventList_T&,
00048                               const stdair::Date_T&, const stdair::Date_T&);
00049
00051     static void addRMEventsToEventQueue (SEVMGR::SEVMGR_ServicePtr_T,
00052                                          stdair::RMEventList_T&);
00053
00055     static void calculateAvailability (const stdair::BomRoot&,
00056                                        stdair::TravelSolutionStruct&);
00057
00059     static void calculateAvailabilityByAU (stdair::TravelSolutionStruct&);
00060
00062     static void calculateAvailabilityByRAE (stdair::TravelSolutionStruct&);
00063
00068     static void calculateAvailabilityByIBP (stdair::TravelSolutionStruct&);
00069
00076     static void calculateAvailabilityByProtectiveIBP (
      stdair::TravelSolutionStruct&);
00077
00079     static bool sell (stdair::Inventory&, const std::string& iSegmentDateKey,
00080                       const stdair::ClassCode_T&, const stdair::PartySize_T&);
00081
00083     static bool sell (const stdair::BookingClassID_T&,
00084                       const stdair::PartySize_T&);
00085
00087     static bool cancel (stdair::Inventory&, const std::string& iSegmentDateKey,
00088                         const stdair::ClassCode_T&, const stdair::PartySize_T&)
      ;
00089
00091     static bool cancel (const stdair::BookingClassID_T&,
00092                         const stdair::PartySize_T&);
00093
00095     static void takeSnapshots (const stdair::Inventory&,
00096                                const stdair::DateTime_T&);
00097
00099     static void updateBookingControls (stdair::FlightDate&);
00100
00102     static void recalculateAvailability (stdair::FlightDate&);
00103
00104   public:
00107     static void createDirectAccesses (const stdair::BomRoot
      &);
00108     static void createDirectAccesses (const stdair::BomRoot
      &,
00109                                       stdair::Inventory&);
00110     static void createDirectAccesses (const stdair::BomRoot
      &,
00111                                       stdair::Inventory&, stdair::FlightDate&);
00112     static void createDirectAccesses (stdair::SegmentDate&)
      ;
00113
00116     static void createPartnerAccesses (const
      stdair::BomRoot&,
00117                                        stdair::Inventory&);
00118     static void createPartnerAccesses (stdair::FlightDate&
      );
00119     static void createPartnerAccesses (const
      stdair::BomRoot&,
00120                                        stdair::Inventory&, stdair::FlightDate&)
      ;
00121
00122
00125     static void buildSimilarSegmentCabinSets (const
       stdair::BomRoot&);
00126     static void buildSimilarSegmentCabinSets (
      stdair::Inventory&);
00127     static void buildSegmentSnapshotTable (
      stdair::Inventory&,
00128                                            const stdair::TableID_T&,
00129                                            const DepartureDateSegmentCabinMap_T
      &);
00130
00131
00133     static void setDefaultBidPriceVector (
      stdair::BomRoot&);
00134     static void setDefaultBidPriceVector (
      stdair::Inventory&);
00135
00137     static void initialiseYieldBasedNestingStructures
       (const stdair::BomRoot&);
00138
```

```
00140     static void initialiseListsOfUsablePolicies
      (const stdair::BomRoot&);
00141
00142   private:
00144     InventoryManager() {}
00145     InventoryManager(const InventoryManager&) {
    }
00147     ~InventoryManager() {}
00148   };
00149
00150 }
00151 #endif // __AIRINV_CMD_INVENTORYMANAGER_HPP
```

23.145 airinv/command/InventoryParser.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/service/Logger.hpp>
#include <airinv/command/InventoryParserHelper.hpp>
#include <airinv/command/InventoryParser.hpp>
#include <airinv/command/InventoryManager.hpp>
```

Namespaces

- namespace AIRINV

23.146 InventoryParser.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasFileMgr.hpp>
00009 #include <stdair/bom/BomRoot.hpp>
00010 #include <stdair/service/Logger.hpp>
00011 // Airinv
00012 #include <airinv/command/InventoryParserHelper.hpp
      >
00013 #include <airinv/command/InventoryParser.hpp>
00014 #include <airinv/command/InventoryManager.hpp
      >
00015
00016 namespace AIRINV {
00017
00018   // ////////////////////////////////////////////////////////////////////
00019   void InventoryParser::
00020   buildInventory (const InventoryFilePath&
    iInventoryFilename,
00021                   stdair::BomRoot& ioBomRoot) {
00022
00023     const stdair::Filename_T lFilename = iInventoryFilename.name();
00024
00025     // Check that the file path given as input corresponds to an actual file
00026     const bool doesExistAndIsReadable =
00027       stdair::BasFileMgr::doesExistAndIsReadable (lFilename);
00028     if (doesExistAndIsReadable == false) {
00029       std::ostringstream oMessage;
00030       oMessage << "The inventory input file, '" << lFilename
00031                << "', can not be retrieved on the file-system";
00032       STDAIR_LOG_ERROR (oMessage.str());
00033       throw InventoryInputFileNotFoundException
       (oMessage.str());
00034     }
00035
00036     // Initialise the inventory file parser.
00037     InventoryFileParser lInventoryParser (ioBomRoot,
    lFilename);
00038
```

```
00039     // Parse the CSV-formatted inventory input file, and generate the
00040     // corresponding Inventory-related objects.
00041     lInventoryParser.buildInventory();
00042   }
00043
00044 }
```

23.147 airinv/command/InventoryParser.hpp File Reference

```
#include <stdair/stdair_basic_types.hpp>
#include <stdair/command/CmdAbstract.hpp>
#include <airinv/AIRINV_Types.hpp>
```

Classes

- class AIRINV::InventoryParser

  Class wrapping the parser entry point.

Namespaces

- namespace stdair

  Forward declarations.
- namespace AIRINV

23.148 InventoryParser.hpp

```
00001 #ifndef __AIRINV_CMD_INVENTORYPARSER_HPP
00002 #define __AIRINV_CMD_INVENTORYPARSER_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009 #include <stdair/command/CmdAbstract.hpp>
00010 // AirInv
00011 #include <airinv/AIRINV_Types.hpp>
00012
00014 namespace stdair {
00015   class BomRoot;
00016 }
00017
00018 namespace AIRINV {
00019
00023   class InventoryParser : public stdair::CmdAbstract {
00024   public:
00034     static void buildInventory (const InventoryFilePath
      &,
00035                                 stdair::BomRoot&);
00036   };
00037 }
00038 #endif // __AIRINV_CMD_INVENTORYPARSER_HPP
```

23.149 airinv/command/InventoryParserHelper.cpp File Reference

```
#include <cassert>
#include <stdair/service/Logger.hpp>
#include <stdair/stdair_exceptions.hpp>
#include <airinv/command/InventoryBuilder.hpp>
#include <airinv/command/InventoryParserHelper.hpp>
```

Namespaces

- namespace AIRINV
- namespace AIRINV::InventoryParserHelper

Functions

- repeat_p_t AIRINV::InventoryParserHelper::airline_code_p (chset_t("0-9A-Z").derived(), 2, 3)
- bounded1_4_p_t AIRINV::InventoryParserHelper::flight_number_p (uint1_4_p.derived(), 0u, 9999u)
- bounded2_p_t AIRINV::InventoryParserHelper::year_p (uint2_p.derived(), 0u, 99u)
- bounded2_p_t AIRINV::InventoryParserHelper::month_p (uint2_p.derived(), 1u, 12u)
- bounded2_p_t AIRINV::InventoryParserHelper::day_p (uint2_p.derived(), 1u, 31u)
- repeat_p_t AIRINV::InventoryParserHelper::dow_p (chset_t("0-1").derived().derived(), 7, 7)
- repeat_p_t AIRINV::InventoryParserHelper::airport_p (chset_t("0-9A-Z").derived(), 3, 3)
- bounded1_2_p_t AIRINV::InventoryParserHelper::hours_p (uint1_2_p.derived(), 0u, 24u)
- bounded2_p_t AIRINV::InventoryParserHelper::minutes_p (uint2_p.derived(), 0u, 59u)
- bounded2_p_t AIRINV::InventoryParserHelper::seconds_p (uint2_p.derived(), 0u, 59u)
- chset_t AIRINV::InventoryParserHelper::cabin_code_p ("A-Z")
- chset_t AIRINV::InventoryParserHelper::class_code_p ("A-Z")
- chset_t AIRINV::InventoryParserHelper::passenger_type_p ("A-Z")
- repeat_p_t AIRINV::InventoryParserHelper::class_code_list_p (chset_t("A-Z").derived(), 1, 26)
- bounded1_3_p_t AIRINV::InventoryParserHelper::stay_duration_p (uint1_3_p.derived(), 0u, 999u)

Variables

- int1_p_t AIRINV::InventoryParserHelper::int1_p
- uint2_p_t AIRINV::InventoryParserHelper::uint2_p
- uint1_2_p_t AIRINV::InventoryParserHelper::uint1_2_p
- uint1_3_p_t AIRINV::InventoryParserHelper::uint1_3_p
- uint4_p_t AIRINV::InventoryParserHelper::uint4_p
- uint1_4_p_t AIRINV::InventoryParserHelper::uint1_4_p
- int1_p_t AIRINV::InventoryParserHelper::family_code_p

23.150 InventoryParserHelper.cpp

```
00001 // //////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // StdAir
00007 #include <stdair/service/Logger.hpp>
00008 #include <stdair/stdair_exceptions.hpp>
00009 // Airinv
00010 #include <airinv/command/InventoryBuilder.hpp
      >
00011 //#define BOOST_SPIRIT_DEBUG
00012 #include <airinv/command/InventoryParserHelper.hpp
      >
00013
00014 //
00015 namespace bsc = boost::spirit::classic;
00016
00017 namespace AIRINV {
00018
00019   namespace InventoryParserHelper {
00020
00021     // //////////////////////////////////////////////////////////////////
00022     //  Semantic actions
00023     // //////////////////////////////////////////////////////////////////
00024
00025     ParserSemanticAction::
00026     ParserSemanticAction (FlightDateStruct
      & ioFlightDate)
00027       : _flightDate (ioFlightDate) {
```

```
00028     }
00029
00030     // ////////////////////////////////////////////////////////////////////
00031     storeSnapshotDate::
00032     storeSnapshotDate (FlightDateStruct&
      ioFlightDate)
00033       : ParserSemanticAction (ioFlightDate) {
00034     }
00035
00036     // ////////////////////////////////////////////////////////////////////
00037     void storeSnapshotDate::operator() (
      iterator_t iStr,
00038                                         iterator_t iStrEnd) const {
00039       _flightDate._flightDate = _flightDate.
      getDate();
00040     }
00041
00042     // ////////////////////////////////////////////////////////////////////
00043     storeAirlineCode::
00044     storeAirlineCode (FlightDateStruct&
      ioFlightDate)
00045       : ParserSemanticAction (ioFlightDate) {
00046     }
00047
00048     // ////////////////////////////////////////////////////////////////////
00049     void storeAirlineCode::operator() (iterator_t
       iStr,
00050                                        iterator_t iStrEnd) const {
00051       const stdair::AirlineCode_T lAirlineCode (iStr, iStrEnd);
00052       _flightDate._airlineCode = lAirlineCode;
00053
00054       // As that's the beginning of a new flight, all the list must be reset
00055       // 1. Leg branch of the tree
00056       _flightDate._legList.clear();
00057       _flightDate._itLeg._cabinList.clear();
00058       _flightDate._itLegCabin._bucketList.
      clear();
00059       _flightDate._itBucket._yieldRangeUpperValue
       = 0.0;
00060
00061       // 2. Segment branch of the tree
00062       _flightDate._segmentList.clear();
00063       _flightDate._itSegment._cabinList.clear();
00064       _flightDate._itSegmentCabin._itFareFamily
      ._classList.clear();
00065       _flightDate._itSegmentCabin._fareFamilies
      .clear();
00066       _flightDate._itBookingClass._classCode
       = "";
00067     }
00068
00069     // ////////////////////////////////////////////////////////////////////
00070     storeFlightNumber::storeFlightNumber (
      FlightDateStruct& ioFlightDate)
00071       : ParserSemanticAction (ioFlightDate) {
00072     }
00073
00074     // ////////////////////////////////////////////////////////////////////
00075     void storeFlightNumber::operator() (unsigned
      int iNumber) const {
00076       _flightDate._flightNumber = iNumber;
00077     }
00078
00079     // ////////////////////////////////////////////////////////////////////
00080     storeFlightDate::storeFlightDate (
      FlightDateStruct& ioFlightDate)
00081       : ParserSemanticAction (ioFlightDate) {
00082     }
00083
00084     // ////////////////////////////////////////////////////////////////////
00085     void storeFlightDate::operator() (iterator_t
       iStr,
00086                                       iterator_t iStrEnd) const {
00087       _flightDate._flightDate = _flightDate.
      getDate();
00088     }
00089
00090     // ////////////////////////////////////////////////////////////////////
00091     storeFlightTypeCode::storeFlightTypeCode
      (FlightDateStruct& ioFlightDate)
00092       : ParserSemanticAction (ioFlightDate) {
00093     }
00094
00095     // ////////////////////////////////////////////////////////////////////
00096     void storeFlightTypeCode::operator() (
      iterator_t iStr,
00097                                           iterator_t iStrEnd) const {
```

```
00098       const std::string lFlightTypeCodeStr (iStr, iStrEnd);
00099       const FlightTypeCode lFlightTypeCode (lFlightTypeCodeStr);
00100       _flightDate._flightTypeCode = lFlightTypeCode.
      getCode();
00101       //STDAIR_LOG_DEBUG ("FlightType code: " << lFlightTypeCode);
00102     }
00103
00104     // ////////////////////////////////////////////////////////////////
00105     storeFlightVisibilityCode::
00106     storeFlightVisibilityCode (FlightDateStruct
      & ioFlightDate)
00107       : ParserSemanticAction (ioFlightDate) {
00108     }
00109
00110     // ////////////////////////////////////////////////////////////////
00111     void storeFlightVisibilityCode::operator()
      (iterator_t iStr,
00112                                                 iterator_t iStrEnd)
       const {
00113       const std::string lFlightVisibilityCodeStr (iStr, iStrEnd);
00114       const FlightVisibilityCode lFlightVisibilityCode(
      lFlightVisibilityCodeStr);
00115       _flightDate._flightVisibilityCode =
      lFlightVisibilityCode.getCode();
00116       //STDAIR_LOG_DEBUG ("FlightVisibility code: " << lFlightVisibilityCode);
00117     }
00118
00119     // ////////////////////////////////////////////////////////////////
00120     storeLegBoardingPoint::
00121     storeLegBoardingPoint (FlightDateStruct
      & ioFlightDate)
00122       : ParserSemanticAction (ioFlightDate) {
00123     }
00124
00125     // ////////////////////////////////////////////////////////////////
00126     void storeLegBoardingPoint::operator() (
      iterator_t iStr,
00127                                             iterator_t iStrEnd) const
       {
00128       stdair::AirportCode_T lBoardingPoint (iStr, iStrEnd);
00129
00130       // ////////////////////
00131       // If this is not the first leg-date of the flight-date,
00132       // the already parsed leg-date must be added to the flight-date.
00133       if (_flightDate._itLeg._cabinList.empty() ==
      false) {
00134         _flightDate._itLegCabin._bucketList.
      push_back (_flightDate._itBucket);
00135         _flightDate._itLeg._cabinList.push_back (
      _flightDate._itLegCabin);
00136         _flightDate._legList.push_back (_flightDate
      ._itLeg);
00137       }
00138
00139       // As that's the beginning of a new leg-date,
00140       // (re-)initialise the leg-cabin branch of the tree
00141       _flightDate._itLeg._cabinList.clear();
00142       _flightDate._itLegCabin._cabinCode = "";
00143       _flightDate._itLegCabin._bucketList.
      clear();
00144       _flightDate._itBucket._yieldRangeUpperValue
       = 0.0;
00145
00146       // ////////////////////
00147       // Set the (default) operating airline and flight number
00148       _flightDate._itLeg._airlineCode =
      _flightDate._airlineCode;
00149       _flightDate._itLeg._flightNumber =
      _flightDate._flightNumber;
00150
00151       // Set the (new) boarding point
00152       _flightDate._itLeg._boardingPoint =
      lBoardingPoint;
00153
00154       // Add the airport code if it is not already stored in the airport lists
00155       _flightDate.addAirport (lBoardingPoint);
00156     }
00157
00158     // ////////////////////////////////////////////////////////////////
00159     storeLegOffPoint::
00160     storeLegOffPoint (FlightDateStruct&
      ioFlightDate)
00161       : ParserSemanticAction (ioFlightDate) {
00162     }
00163
00164     // ////////////////////////////////////////////////////////////////
00165     void storeLegOffPoint::operator() (iterator_t
```

```
      iStr,
00166                                       iterator_t iStrEnd) const {
00167       stdair::AirportCode_T lOffPoint (iStr, iStrEnd);
00168       _flightDate._itLeg._offPoint = lOffPoint;
00169
00170       // Add the airport code if it is not already stored in the airport lists
00171       _flightDate.addAirport (lOffPoint);
00172     }
00173
00174     // ////////////////////////////////////////////////////////////////////
00175     storeOperatingAirlineCode::
00176     storeOperatingAirlineCode (FlightDateStruct
      & ioFlightDate)
00177       : ParserSemanticAction (ioFlightDate) {
00178     }
00179
00180     // ////////////////////////////////////////////////////////////////////
00181     void storeOperatingAirlineCode::operator()
      (iterator_t iStr,
00182                                               iterator_t iStrEnd)
       const {
00183       const stdair::AirlineCode_T lAirlineCode (iStr, iStrEnd);
00184       if (lAirlineCode.size() == 2) {
00185         _flightDate._itLeg._airlineCode =
      lAirlineCode;
00186       }
00187       //STDAIR_LOG_DEBUG ("Airline code: " << lAirlineCode);
00188     }
00189
00190     // ////////////////////////////////////////////////////////////////////
00191     storeOperatingFlightNumber::
00192     storeOperatingFlightNumber (FlightDateStruct
      & ioFlightDate)
00193       : ParserSemanticAction (ioFlightDate) {
00194     }
00195
00196     // ////////////////////////////////////////////////////////////////////
00197     void storeOperatingFlightNumber::operator()
      (unsigned int iNumber) const {
00198       _flightDate._itLeg._flightNumber = iNumber;
00199       //STDAIR_LOG_DEBUG ("Flight number: " << iNumber);
00200     }
00201
00202
00203     // ////////////////////////////////////////////////////////////////////
00204     storeBoardingDate::storeBoardingDate (
      FlightDateStruct& ioFlightDate)
00205       : ParserSemanticAction (ioFlightDate) {
00206     }
00207
00208     // ////////////////////////////////////////////////////////////////////
00209     void storeBoardingDate::operator() (
      iterator_t iStr,
00210                                         iterator_t iStrEnd) const {
00211       _flightDate._itLeg._boardingDate =
      _flightDate.getDate();
00212     }
00213
00214     // ////////////////////////////////////////////////////////////////////
00215     storeBoardingTime::storeBoardingTime (
      FlightDateStruct& ioFlightDate)
00216       : ParserSemanticAction (ioFlightDate) {
00217     }
00218
00219     // ////////////////////////////////////////////////////////////////////
00220     void storeBoardingTime::operator() (
      iterator_t iStr,
00221                                         iterator_t iStrEnd) const {
00222       _flightDate._itLeg._boardingTime =
      _flightDate.getTime();
00223
00224       // Reset the number of seconds
00225       _flightDate._itSeconds = 0;
00226
00227       // Reset the date off-set
00228       _flightDate._dateOffSet = 0;
00229     }
00230
00231     // ////////////////////////////////////////////////////////////////////
00232     storeOffDate::storeOffDate (FlightDateStruct
      & ioFlightDate)
00233       : ParserSemanticAction (ioFlightDate) {
00234     }
00235
00236     // ////////////////////////////////////////////////////////////////////
00237     void storeOffDate::operator() (iterator_t
       iStr, iterator_t iStrEnd) const {
```

```
00238       _flightDate._itLeg._offDate = _flightDate
    .getDate();
00239     }
00240
00241     // ////////////////////////////////////////////////////////////////////
00242     storeOffTime::storeOffTime (FlightDateStruct
    & ioFlightDate)
00243       : ParserSemanticAction (ioFlightDate) {
00244     }
00245
00246     // ////////////////////////////////////////////////////////////////////
00247     void storeOffTime::operator() (iterator_t
     iStr, iterator_t iStrEnd) const {
00248       _flightDate._itLeg._offTime = _flightDate
    .getTime();
00249
00250       // Reset the number of seconds
00251       _flightDate._itSeconds = 0;
00252     }
00253
00254     // ////////////////////////////////////////////////////////////////////
00255     storeLegCabinCode::storeLegCabinCode (
    FlightDateStruct& ioFlightDate)
00256       : ParserSemanticAction (ioFlightDate) {
00257     }
00258
00259     // ////////////////////////////////////////////////////////////////////
00260     void storeLegCabinCode::operator() (char
    iChar) const {
00261
00262       // ////////////////////
00263       // If this is not the first leg-cabin of the leg-date,
00264       // the already parsed leg-cabin must be added to the leg-date.
00265       if (_flightDate._itLegCabin._cabinCode !=
     "") {
00266         if (_flightDate._itLegCabin._bucketList
    .empty() == false) {
00267           _flightDate._itLegCabin._bucketList.
    push_back (_flightDate._itBucket);
00268         }
00269         _flightDate._itLeg._cabinList.push_back (
    _flightDate._itLegCabin);
00270       }
00271
00272       // (Re-)initialise the leg-cabin branch of the tree
00273       _flightDate._itLegCabin._bucketList.
    clear();
00274       _flightDate._itBucket._yieldRangeUpperValue
     = 0.0;
00275
00276
00277       // ////////////////////
00278       _flightDate._itLegCabin._cabinCode =
    iChar;
00279       //std::cout << "Cabin code: " << iChar << std::endl;
00280     }
00281
00282     // ////////////////////////////////////////////////////////////////////
00283     storeSaleableCapacity::
00284     storeSaleableCapacity (FlightDateStruct
    & ioFlightDate)
00285       : ParserSemanticAction (ioFlightDate) {
00286     }
00287
00288     // ////////////////////////////////////////////////////////////////////
00289     void storeSaleableCapacity::operator() (
    double iReal) const {
00290       _flightDate._itLegCabin._saleableCapacity
     = iReal;
00291       //std::cout << "Saleable capacity: " << iReal << std::endl;
00292     }
00293
00294     // ////////////////////////////////////////////////////////////////////
00295     storeAU::storeAU (FlightDateStruct&
    ioFlightDate)
00296       : ParserSemanticAction (ioFlightDate) {
00297     }
00298
00299     // ////////////////////////////////////////////////////////////////////
00300     void storeAU::operator() (double iReal) const {
00301       _flightDate._itLegCabin._au = iReal;
00302       //std::cout << "AU: " << iReal << std::endl;
00303     }
00304
00305     // ////////////////////////////////////////////////////////////////////
00306     storeUPR::storeUPR (FlightDateStruct&
    ioFlightDate)
```

```
00307       : ParserSemanticAction (ioFlightDate) {
00308     }
00309
00310     // ////////////////////////////////////////////////////////////////////
00311     void storeUPR::operator() (double iReal) const {
00312       _flightDate._itLegCabin._upr = iReal;
00313       //std::cout << "UPR: " << iReal << std::endl;
00314     }
00315
00316     // ////////////////////////////////////////////////////////////////////
00317     storeBookingCounter::storeBookingCounter
       (FlightDateStruct& ioFlightDate)
00318       : ParserSemanticAction (ioFlightDate) {
00319     }
00320
00321     // ////////////////////////////////////////////////////////////////////
00322     void storeBookingCounter::operator() (
      double iReal) const {
00323       _flightDate._itLegCabin._nbOfBookings
      = iReal;
00324       //std::cout << "Nb of bookings: " << iReal << std::endl;
00325     }
00326
00327     // ////////////////////////////////////////////////////////////////////
00328     storeNAV::storeNAV (FlightDateStruct&
      ioFlightDate)
00329       : ParserSemanticAction (ioFlightDate) {
00330     }
00331
00332     // ////////////////////////////////////////////////////////////////////
00333     void storeNAV::operator() (double iReal) const {
00334       _flightDate._itLegCabin._nav = iReal;
00335       //std::cout << "NAV: " << iReal << std::endl;
00336     }
00337
00338     // ////////////////////////////////////////////////////////////////////
00339     storeGAV::storeGAV (FlightDateStruct&
      ioFlightDate)
00340       : ParserSemanticAction (ioFlightDate) {
00341     }
00342
00343     // ////////////////////////////////////////////////////////////////////
00344     void storeGAV::operator() (double iReal) const {
00345       _flightDate._itLegCabin._gav = iReal;
00346       //std::cout << "GAV: " << iReal << std::endl;
00347     }
00348
00349     // ////////////////////////////////////////////////////////////////////
00350     storeACP::storeACP (FlightDateStruct&
      ioFlightDate)
00351       : ParserSemanticAction (ioFlightDate) {
00352     }
00353
00354     // ////////////////////////////////////////////////////////////////////
00355     void storeACP::operator() (double iReal) const {
00356       _flightDate._itLegCabin._acp = iReal;
00357       //std::cout << "ACP: " << iReal << std::endl;
00358     }
00359
00360     // ////////////////////////////////////////////////////////////////////
00361     storeETB::storeETB (FlightDateStruct&
      ioFlightDate)
00362       : ParserSemanticAction (ioFlightDate) {
00363     }
00364
00365     // ////////////////////////////////////////////////////////////////////
00366     void storeETB::operator() (double iReal) const {
00367       _flightDate._itLegCabin._etb = iReal;
00368       //std::cout << "ETB: " << iReal << std::endl;
00369     }
00370
00371     // ////////////////////////////////////////////////////////////////////
00372     storeYieldUpperRange::storeYieldUpperRange
       (FlightDateStruct& ioFlightDate)
00373       : ParserSemanticAction (ioFlightDate) {
00374     }
00375
00376     // ////////////////////////////////////////////////////////////////////
00377     void storeYieldUpperRange::operator() (
      double iReal) const {
00378       // If this is not the first bucket of the leg-cabin,
00379       // the already parsed bucket must be added to the leg-cabin.
00380       if (_flightDate._itBucket._yieldRangeUpperValue
        != 0.0) {
00381         _flightDate._itLegCabin._bucketList.
      push_back (_flightDate._itBucket);
00382       }
```

```
00383
00384
00385       // ////////////////////
00386       _flightDate._itBucket._yieldRangeUpperValue
     = iReal;
00387       //std::cout << "Yield Upper Range Value: " << iReal << std::endl;
00388     }
00389
00390     // ////////////////////////////////////////////////////////////////
00391     storeBucketAvaibality::
00392     storeBucketAvaibality (FlightDateStruct
     & ioFlightDate)
00393       : ParserSemanticAction (ioFlightDate) {
00394     }
00395
00396     // ////////////////////////////////////////////////////////////////
00397     void storeBucketAvaibality::operator() (
     double iReal) const {
00398       _flightDate._itBucket._availability =
     iReal;
00399       //std::cout << "Availability: " << iReal << std::endl;
00400     }
00401
00402     // ////////////////////////////////////////////////////////////////
00403     storeSeatIndex::storeSeatIndex (
     FlightDateStruct& ioFlightDate)
00404       : ParserSemanticAction (ioFlightDate) {
00405     }
00406
00407     // ////////////////////////////////////////////////////////////////
00408     void storeSeatIndex::operator() (double iReal)
      const {
00409       _flightDate._itBucket._seatIndex = iReal;
00410       //std::cout << "Seat Index: " << iReal << std::endl;
00411     }
00412
00413     // ////////////////////////////////////////////////////////////////
00414     storeSegmentBoardingPoint::
00415     storeSegmentBoardingPoint (FlightDateStruct
     & ioFlightDate)
00416       : ParserSemanticAction (ioFlightDate) {
00417     }
00418
00419     // ////////////////////////////////////////////////////////////////
00420     void storeSegmentBoardingPoint::operator()
     (iterator_t iStr,
00421                                             iterator_t iStrEnd)
      const {
00422       stdair::AirportCode_T lBoardingPoint (iStr, iStrEnd);
00423
00424       // ////////////////////
00425       // When the first segment-date is read, it means that the leg section
00426       // is over. The parsed leg can therefore be added to the list.
00427       if (_flightDate._itLeg._cabinList.empty() ==
     false) {
00428         _flightDate._itLegCabin._bucketList.
     push_back (_flightDate._itBucket);
00429         _flightDate._itLeg._cabinList.push_back (
     _flightDate._itLegCabin);
00430         _flightDate._legList.push_back (_flightDate
     ._itLeg);
00431
00432         // (Re-)initialise the leg-date branch of the tree
00433         _flightDate._itLeg._cabinList.clear();
00434         _flightDate._itLegCabin._cabinCode = ""
     ;
00435         _flightDate._itLeg._cabinList.clear();
00436         _flightDate._itLegCabin._bucketList.
     clear();
00437       }
00438
00439
00440       // ////////////////////
00441       // If this is not the first segment-date of the flight-date,
00442       // the already parsed segment-date must be added to the flight-date.
00443       if (_flightDate._itSegment._cabinList.
     empty() == false) {
00444         _flightDate._itSegmentCabin._itFareFamily
     ._classList.push_back (_flightDate._itBookingClass
     );
00445         _flightDate._itSegmentCabin._fareFamilies
     .push_back (_flightDate._itSegmentCabin._itFareFamily
     );
00446         _flightDate._itSegment._cabinList.
     push_back (_flightDate._itSegmentCabin);
00447         _flightDate._segmentList.push_back (_flightDate
     ._itSegment);
```

```
00448       }
00449
00450       // As that's the beginning of a new segment-date,
00451       // (re-)initialise the segment-cabin branch of the tree
00452       _flightDate._itSegment._cabinList.clear();
00453       _flightDate._itSegmentCabin._itFareFamily
    ._classList.clear();
00454       _flightDate._itSegmentCabin._fareFamilies
    .clear();
00455       _flightDate._itBookingClass._classCode
     = "";
00456
00457
00458       // ////////////////////
00459       _flightDate._itSegment._boardingPoint
    = lBoardingPoint;
00460       //std::cout << "Board point: " << lBoardingPoint << std::endl;
00461     }
00462
00463     // ////////////////////////////////////////////////////////////////
00464     storeSegmentOffPoint::storeSegmentOffPoint
     (FlightDateStruct& ioFlightDate)
00465       : ParserSemanticAction (ioFlightDate) {
00466     }
00467
00468     // ////////////////////////////////////////////////////////////////
00469     void storeSegmentOffPoint::operator() (
    iterator_t iStr,
00470                                            iterator_t iStrEnd) const
    {
00471       stdair::AirportCode_T lOffPoint (iStr, iStrEnd);
00472       _flightDate._itSegment._offPoint =
    lOffPoint;
00473       //std::cout << "Off point: " << lOffPoint << std::endl;
00474     }
00475
00476     // ////////////////////////////////////////////////////////////////
00477     storeSegmentCabinCode::
00478     storeSegmentCabinCode (FlightDateStruct
    & ioFlightDate)
00479       : ParserSemanticAction (ioFlightDate) {
00480     }
00481
00482     // ////////////////////////////////////////////////////////////////
00483     void storeSegmentCabinCode::operator() (
    char iChar) const {
00484
00485       // Reset the list of fare families, as it is a new segment-cabin
00486       _flightDate._itSegmentCabin._fareFamilies
    .clear();
00487
00488       // ////////////////////
00489       // If this is not the first segment-cabin of the segment-date,
00490       // the already parsed segment-cabin must be added to the segment-date.
00491       if (_flightDate._itSegmentCabin._itFareFamily
    ._classList.empty() == false){
00492         _flightDate._itSegmentCabin._itFareFamily
    ._classList.
00493           push_back (_flightDate._itBookingClass);
00494         _flightDate._itSegmentCabin._fareFamilies
    .
00495           push_back (_flightDate._itSegmentCabin.
    _itFareFamily);
00496         _flightDate._itSegment._cabinList.
00497           push_back (_flightDate._itSegmentCabin);
00498       }
00499
00500       // (Re-)initialise the booking-class branch of the tree
00501       _flightDate._itSegmentCabin._fareFamilies
    .clear();
00502       _flightDate._itSegmentCabin._itFareFamily
    ._classList.clear();
00503       _flightDate._itBookingClass._classCode
     = "";
00504
00505
00506       // ////////////////////
00507       _flightDate._itSegmentCabin._cabinCode
     = iChar;
00508       //std::cout << "Segment-cabin code: " << iChar << std::endl;
00509     }
00510
00511     // ////////////////////////////////////////////////////////////////
00512     storeSegmentCabinBookingCounter::
00513     storeSegmentCabinBookingCounter (
    FlightDateStruct& ioFlightDate)
00514       : ParserSemanticAction (ioFlightDate) {
```

```
00515     }
00516
00517     // ////////////////////////////////////////////////////////////////////
00518     void storeSegmentCabinBookingCounter::operator()
      (double iReal) const {
00519       _flightDate._itSegmentCabin._nbOfBookings
       = iReal;
00520       //std::cout << "Nb of bookings: " << iReal << std::endl;
00521     }
00522
00523     // ////////////////////////////////////////////////////////////////////
00524     storeClassCode::storeClassCode (
      FlightDateStruct& ioFlightDate)
00525       : ParserSemanticAction (ioFlightDate) {
00526     }
00527
00528     // ////////////////////////////////////////////////////////////////////
00529     void storeClassCode::operator() (char iChar)
       const {
00530       // If this is not the first booking-class of the segment-cabin,
00531       // the already parsed booking-class must be added to the segment-cabin.
00532       if (_flightDate._itBookingClass._classCode
       != "") {
00533         _flightDate._itSegmentCabin._itFareFamily
      ._classList.
00534           push_back (_flightDate._itBookingClass);
00535       }
00536
00537       // ////////////////////
00538       _flightDate._itBookingClass._classCode
       = iChar;
00539       //std::cout << "Booking class code: " << iChar << std::endl;
00540     }
00541
00542     // ////////////////////////////////////////////////////////////////////
00543     storeSubclassCode::storeSubclassCode (
      FlightDateStruct& ioFlightDate)
00544       : ParserSemanticAction (ioFlightDate) {
00545     }
00546
00547     // ////////////////////////////////////////////////////////////////////
00548     void storeSubclassCode::operator() (unsigned
      int iNumber) const {
00549       _flightDate._itBookingClass._subclassCode
       = iNumber;
00550       //std::cout << "Sub-class code: " << iNumber << std::endl;
00551     }
00552
00553     // ////////////////////////////////////////////////////////////////////
00554     storeParentClassCode::
00555     storeParentClassCode (FlightDateStruct
      & ioFlightDate)
00556       : ParserSemanticAction (ioFlightDate) {
00557     }
00558
00559     // ////////////////////////////////////////////////////////////////////
00560     void storeParentClassCode::operator() (
      char iChar) const {
00561       _flightDate._itBookingClass._parentClassCode
       = iChar;
00562       //std::cout << "Parent booking class code: " << iChar << std::endl;
00563     }
00564
00565     // ////////////////////////////////////////////////////////////////////
00566     storeParentSubclassCode::
00567     storeParentSubclassCode (FlightDateStruct
      & ioFlightDate)
00568       : ParserSemanticAction (ioFlightDate) {
00569     }
00570
00571     // ////////////////////////////////////////////////////////////////////
00572     void storeParentSubclassCode::operator()
      (unsigned int iNumber) const {
00573       _flightDate._itBookingClass._parentSubclassCode
       = iNumber;
00574       //std::cout << "Parent sub-class code: " << iNumber << std::endl;
00575     }
00576
00577     // ////////////////////////////////////////////////////////////////////
00578     storeCumulatedProtection::
00579     storeCumulatedProtection (FlightDateStruct
      & ioFlightDate)
00580       : ParserSemanticAction (ioFlightDate) {
00581     }
00582
00583     // ////////////////////////////////////////////////////////////////////
00584     void storeCumulatedProtection::operator()
```

```
(double iReal) const {
00585       _flightDate._itBookingClass.
_cumulatedProtection = iReal;
00586       //std::cout << "Cumulated protection: " << iReal << std::endl;
00587     }
00588
00589     // ////////////////////////////////////////////////////////////////////
00590     storeProtection::storeProtection (
FlightDateStruct& ioFlightDate)
00591       : ParserSemanticAction (ioFlightDate) {
00592     }
00593
00594     // ////////////////////////////////////////////////////////////////////
00595     void storeProtection::operator() (double iReal)
 const {
00596       _flightDate._itBookingClass._protection
 = iReal;
00597       //std::cout << "Protection: " << iReal << std::endl;
00598     }
00599
00600     // ////////////////////////////////////////////////////////////////////
00601     storeNego::storeNego (FlightDateStruct&
 ioFlightDate)
00602       : ParserSemanticAction (ioFlightDate) {
00603     }
00604
00605     // ////////////////////////////////////////////////////////////////////
00606     void storeNego::operator() (double iReal) const {
00607       _flightDate._itBookingClass._nego = iReal;

00608       //std::cout << "Negotiated allotment: " << iReal << std::endl;
00609     }
00610
00611     // ////////////////////////////////////////////////////////////////////
00612     storeNoShow::storeNoShow (FlightDateStruct
& ioFlightDate)
00613       : ParserSemanticAction (ioFlightDate) {
00614     }
00615
00616     // ////////////////////////////////////////////////////////////////////
00617     void storeNoShow::operator() (double iReal) const {
00618       _flightDate._itBookingClass._noShowPercentage
 = iReal;
00619       //std::cout << "No-Show percentage: " << iReal << std::endl;
00620     }
00621
00622     // ////////////////////////////////////////////////////////////////////
00623     storeOverbooking::storeOverbooking (
FlightDateStruct& ioFlightDate)
00624       : ParserSemanticAction (ioFlightDate) {
00625     }
00626
00627     // ////////////////////////////////////////////////////////////////////
00628     void storeOverbooking::operator() (double
iReal) const {
00629       _flightDate._itBookingClass.
_overbookingPercentage = iReal;
00630       //std::cout << "Overbooking percentage: " << iReal << std::endl;
00631     }
00632
00633     // ////////////////////////////////////////////////////////////////////
00634     storeNbOfBkgs::storeNbOfBkgs (FlightDateStruct
& ioFlightDate)
00635       : ParserSemanticAction (ioFlightDate) {
00636     }
00637
00638     // ////////////////////////////////////////////////////////////////////
00639     void storeNbOfBkgs::operator() (double iReal)
 const {
00640       _flightDate._itBookingClass._nbOfBookings
 = iReal;
00641       //std::cout << "Nb of bookings: " << iReal << std::endl;
00642     }
00643
00644     // ////////////////////////////////////////////////////////////////////
00645     storeNbOfGroupBkgs::storeNbOfGroupBkgs
 (FlightDateStruct& ioFlightDate)
00646       : ParserSemanticAction (ioFlightDate) {
00647     }
00648
00649     // ////////////////////////////////////////////////////////////////////
00650     void storeNbOfGroupBkgs::operator() (double
iReal) const {
00651       _flightDate._itBookingClass._nbOfGroupBookings
 = iReal;
00652       //std::cout << "Nb of group bookings: " << iReal << std::endl;
00653     }
```

```
00654
00655     // ////////////////////////////////////////////////////////////////////
00656     storeNbOfPendingGroupBkgs::
00657     storeNbOfPendingGroupBkgs (FlightDateStruct
      & ioFlightDate)
00658       : ParserSemanticAction (ioFlightDate) {
00659     }
00660
00661     // ////////////////////////////////////////////////////////////////////
00662     void storeNbOfPendingGroupBkgs::operator()
      (double iReal) const {
00663       _flightDate._itBookingClass.
      _nbOfPendingGroupBookings = iReal;
00664       //std::cout << "Nb of pending group bookings: " << iReal << std::endl;
00665     }
00666
00667     // ////////////////////////////////////////////////////////////////////
00668     storeNbOfStaffBkgs::storeNbOfStaffBkgs
       (FlightDateStruct& ioFlightDate)
00669       : ParserSemanticAction (ioFlightDate) {
00670     }
00671
00672     // ////////////////////////////////////////////////////////////////////
00673     void storeNbOfStaffBkgs::operator() (double
      iReal) const {
00674       _flightDate._itBookingClass._nbOfStaffBookings
       = iReal;
00675       //std::cout << "Nb of staff bookings: " << iReal << std::endl;
00676     }
00677
00678     // ////////////////////////////////////////////////////////////////////
00679     storeNbOfWLBkgs::storeNbOfWLBkgs (
      FlightDateStruct& ioFlightDate)
00680       : ParserSemanticAction (ioFlightDate) {
00681     }
00682
00683     // ////////////////////////////////////////////////////////////////////
00684     void storeNbOfWLBkgs::operator() (double iReal)
       const {
00685       _flightDate._itBookingClass._nbOfWLBookings
       = iReal;
00686       //std::cout << "Nb of wait-list bookings: " << iReal << std::endl;
00687     }
00688
00689     // ////////////////////////////////////////////////////////////////////
00690     storeClassETB::storeClassETB (FlightDateStruct
      & ioFlightDate)
00691       : ParserSemanticAction (ioFlightDate) {
00692     }
00693
00694     // ////////////////////////////////////////////////////////////////////
00695     void storeClassETB::operator() (double iReal)
       const {
00696       _flightDate._itBookingClass._etb = iReal;
00697       //std::cout << "Class-level ETB: " << iReal << std::endl;
00698     }
00699
00700     // ////////////////////////////////////////////////////////////////////
00701     storeClassAvailability::
00702     storeClassAvailability (FlightDateStruct
      & ioFlightDate)
00703       : ParserSemanticAction (ioFlightDate) {
00704     }
00705
00706     // ////////////////////////////////////////////////////////////////////
00707     void storeClassAvailability::operator()
      (double iReal) const {
00708       _flightDate._itBookingClass.
      _netClassAvailability = iReal;
00709       //std::cout << "Net class availability: " << iReal << std::endl;
00710     }
00711
00712     // ////////////////////////////////////////////////////////////////////
00713     storeSegmentAvailability::
00714     storeSegmentAvailability (FlightDateStruct
      & ioFlightDate)
00715       : ParserSemanticAction (ioFlightDate) {
00716     }
00717
00718     // ////////////////////////////////////////////////////////////////////
00719     void storeSegmentAvailability::operator()
      (double iReal) const {
00720       _flightDate._itBookingClass.
      _segmentAvailability = iReal;
00721       //std::cout << "Segment availability: " << iReal << std::endl;
00722     }
00723
```

```
00724     // ////////////////////////////////////////////////////////////////////
00725     storeRevenueAvailability::
00726     storeRevenueAvailability (FlightDateStruct
      & ioFlightDate)
00727       : ParserSemanticAction (ioFlightDate) {
00728     }
00729
00730     // ////////////////////////////////////////////////////////////////////
00731     void storeRevenueAvailability::operator()
      (double iReal) const {
00732       _flightDate._itBookingClass.
      _netRevenueAvailability = iReal;
00733       //std::cout << "Net revenue availability: " << iReal << std::endl;
00734     }
00735
00736     // ////////////////////////////////////////////////////////////////////
00737     storeFamilyCode::storeFamilyCode (
      FlightDateStruct& ioFlightDate)
00738       : ParserSemanticAction (ioFlightDate) {
00739     }
00740
00741     // ////////////////////////////////////////////////////////////////////
00742     void storeFamilyCode::operator() (int iCode)
       const {
00743       std::ostringstream ostr;
00744       ostr << iCode;
00745       _flightDate._itSegmentCabin._itFareFamily
      ._familyCode = ostr.str();
00746     }
00747
00748     // ////////////////////////////////////////////////////////////////////
00749     storeFClasses::storeFClasses (FlightDateStruct
      & ioFlightDate)
00750       : ParserSemanticAction (ioFlightDate) {
00751     }
00752
00753     // ////////////////////////////////////////////////////////////////////
00754     void storeFClasses::operator() (iterator_t
       iStr,
00755                                     iterator_t iStrEnd) const {
00756       std::string lClasses (iStr, iStrEnd);
00757       _flightDate._itSegmentCabin._itFareFamily
      ._classes = lClasses;
00758
00759       // The list of classes is the last (according to the arrival order
00760       // within the schedule input file) detail of the segment cabin. Hence,
00761       // when a list of classes is parsed, it means that the full segment
00762       // cabin details have already been parsed as well: the segment cabin
00763       // can thus be added to the segment.
00764       _flightDate._itSegmentCabin._itFareFamily
      ._classList.
00765         push_back (_flightDate._itBookingClass);
00766       _flightDate._itSegmentCabin._fareFamilies
      .
00767         push_back (_flightDate._itSegmentCabin.
      _itFareFamily);
00768       _flightDate._itSegment._cabinList.
      push_back (_flightDate._itSegmentCabin);
00769
00770       // As that's the beginning of a new segment-cabin,
00771       // (re-)initialise the segment-cabin branch of the tree
00772       _flightDate._itSegmentCabin._itFareFamily
      ._classList.clear();
00773       _flightDate._itSegmentCabin._fareFamilies
      .clear();
00774       _flightDate._itBookingClass._classCode
       = "";
00775     }
00776
00777     // ////////////////////////////////////////////////////////////////////
00778     doEndFlightDate::doEndFlightDate (
      stdair::BomRoot& ioBomRoot,
00779                                       FlightDateStruct&
      ioFlightDate,
00780                                       unsigned int& ioNbOfFlights)
00781       : ParserSemanticAction (ioFlightDate), _bomRoot (
      ioBomRoot),
00782         _nbOfFlights (ioNbOfFlights) {
00783     }
00784
00785     // ////////////////////////////////////////////////////////////////////
00786     // void doEndFlightDate::operator() (char iChar) const {
00787     void doEndFlightDate::operator() (iterator_t
       iStr,
00788                                       iterator_t iStrEnd) const {
00789
00790       // ////////////////////
```

```
00791       // The segment-date section is now over. It means that the
00792       // already parsed segment-date must be added to the flight-date.
00793       if (_flightDate._itSegment._cabinList.
      empty() == false) {
00794         _flightDate._segmentList.push_back (_flightDate
      ._itSegment);
00795       }
00796
00797       // As that's the beginning of a new flight-date,
00798       // (re-)initialise the segment-cabin branch of the tree
00799       _flightDate._itSegment._cabinList.clear();
00800
00801
00802       // ////////////////////
00803       //if (_nbOfFlights % 1000 == 0) {
00804         // DEBUG: Display the result
00805         //STDAIR_LOG_DEBUG ("FlightDate #" << _nbOfFlights
00806         //                  << ": " << _flightDate.describe());
00807         //}
00808
00809       // Build the FlightDate BOM objects
00810       InventoryBuilder::buildInventory (_bomRoot, _flightDate
      );
00811
00812       //
00813       ++_nbOfFlights;
00814     }
00815
00816
00817     // ////////////////////////////////////////////////////////////////////
00818     //
00819     //  Utility Parsers
00820     //
00821     // ////////////////////////////////////////////////////////////////////
00823     int1_p_t int1_p;
00824
00826     uint2_p_t uint2_p;
00827
00829     uint1_2_p_t uint1_2_p;
00830
00832     uint1_3_p_t uint1_3_p;
00833
00835     uint4_p_t uint4_p;
00836
00838     uint1_4_p_t uint1_4_p;
00839
00841     repeat_p_t airline_code_p (chset_t("0-9A-Z")
      .derived(), 2, 3);
00842
00844     bounded1_4_p_t flight_number_p (uint1_4_p
      .derived(), 0u, 9999u);
00845
00847     bounded2_p_t year_p (uint2_p.derived(), 0u, 99u);
00848
00850     bounded2_p_t month_p (uint2_p.derived(), 1u, 12u)
      ;
00851
00853     bounded2_p_t day_p (uint2_p.derived(), 1u, 31u);
00854
00856     repeat_p_t dow_p (chset_t("0-1").derived().derived(),
       7, 7);
00857
00859     repeat_p_t airport_p (chset_t("0-9A-Z").derived()
      , 3, 3);
00860
00862     bounded1_2_p_t hours_p (uint1_2_p.derived(),
      0u, 24u);
00863
00865     bounded2_p_t minutes_p (uint2_p.derived(), 0u,
      59u);
00866
00868     bounded2_p_t seconds_p (uint2_p.derived(), 0u,
      59u);
00869
00871     chset_t cabin_code_p ("A-Z");
00872
00874     chset_t class_code_p ("A-Z");
00875
00877     chset_t passenger_type_p ("A-Z");
00878
00880     int1_p_t family_code_p;
00881
00883     repeat_p_t class_code_list_p (chset_t("
      A-Z").derived(), 1, 26);
00884
00886     bounded1_3_p_t stay_duration_p (uint1_3_p
      .derived(), 0u, 999u);
```

```
00887
00888
00889     // ////////////////////////////////////////////////////////////////////
00890     //  (Boost Spirit) Grammar Definition
00891     // ////////////////////////////////////////////////////////////////////
00892
00893     // ////////////////////////////////////////////////////////////////////
00894     InventoryParser::InventoryParser (
      stdair::BomRoot& ioBomRoot,
00895                                       FlightDateStruct&
      ioFlightDate,
00896                                       unsigned int& ioNbOfFlights)
00897       : _bomRoot (ioBomRoot), _flightDate (ioFlightDate),
00898         _nbOfFlights (ioNbOfFlights) {
00899     }
00900
00901     // ////////////////////////////////////////////////////////////////////
00902     template<typename ScannerT>
00903     InventoryParser::definition<ScannerT>::
00904     definition (InventoryParser const& self) {
00905
00906       flight_date_list = *( not_to_be_parsed | flight_date )
00907         ;
00908
00909       not_to_be_parsed =
00910         bsc::lexeme_d[ bsc::comment_p("//") | bsc::comment_p("/*", "*/")
00911                        | bsc::space_p ]
00912         ;
00913
00914       flight_date = flight_key
00915         >> leg_list
00916         >> segment_list
00917         >> flight_date_end[doEndFlightDate (self._bomRoot, self.
      _flightDate,
00918                                             self._nbOfFlights)]
00919         ;
00920
00921       flight_date_end = bsc::ch_p(';')
00922         ;
00923
00924       flight_key = date[storeSnapshotDate(self._flightDate)]
00925         >> '/' >> airline_code
00926         >> '/' >> flight_number
00927         >> '/' >> date[storeFlightDate(self._flightDate)]
00928         >> '/' >> flight_type_code[storeFlightTypeCode(self.
      _flightDate)]
00929         >> !( '/' >> flight_visibility_code[storeFlightVisibilityCode
      (self._flightDate)])
00930         ;
00931
00932       airline_code =
00933         bsc::lexeme_d[(airline_code_p)[storeAirlineCode
      (self._flightDate)]]
00934         ;
00935
00936       flight_number =
00937         bsc::lexeme_d[(flight_number_p)[storeFlightNumber
      (self._flightDate)]]
00938         ;
00939
00940       date =
00941         bsc::lexeme_d[(day_p)[bsc::assign_a(self._flightDate._itDay)]
00942                       >> (month_p)[bsc::assign_a(self._flightDate.
      _itMonth)]
00943                       >> (year_p)[bsc::assign_a(self._flightDate._itYear)
      ]]
00944         ;
00945
00946       flight_type_code =
00947         ( bsc::chseq_p("INT") | bsc::chseq_p("DOM") | bsc::chseq_p("GRD") )
00948         ;
00949
00950       flight_visibility_code =
00951         ( bsc::chseq_p("HID") | bsc::chseq_p("PSD") )
00952         ;
00953
00954       leg_list = +( '/' >> leg )
00955         ;
00956
00957       leg = !( operating_leg_details >> ';' )
00958         >> leg_key >> ';' >> leg_details >> leg_cabin_list
00959         ;
00960
00961       operating_leg_details =
00962         bsc::lexeme_d[(airline_code_p)[storeOperatingAirlineCode
      (self._flightDate)] ]
00963         >> ";"
```

```
00964         >> bsc::lexeme_d[(flight_number_p)[
      storeOperatingFlightNumber(self._flightDate)] ]
00965         ;
00966
00967       leg_key = (airport_p)[storeLegBoardingPoint
      (self._flightDate)]
00968         >> ';' >> (airport_p)[storeLegOffPoint(self.
      _flightDate)]
00969         ;
00970
00971       leg_details = date[storeBoardingDate(self._flightDate)]
00972         >> ';' >> time[storeBoardingTime(self._flightDate)]
00973         >> ';' >> date[storeOffDate(self._flightDate)]
00974         >> ';' >> time[storeOffTime(self._flightDate)]
00975         ;
00976
00977       leg_cabin_list = +( ';' >> leg_cabin_details >> !bucket_list )
00978         ;
00979
00980       leg_cabin_details = (cabin_code_p)[storeLegCabinCode
      (self._flightDate)]
00981         >> ',' >> (bsc::ureal_p)[storeSaleableCapacity(
      self._flightDate)]
00982         >> ',' >> (bsc::real_p)[storeAU(self._flightDate)]
00983         >> ',' >> (bsc::real_p)[storeUPR(self._flightDate)]
00984         >> ',' >> (bsc::real_p)[storeBookingCounter(self.
      _flightDate)]
00985         >> ',' >> (bsc::real_p)[storeNAV(self._flightDate)]
00986         >> ',' >> (bsc::real_p)[storeGAV(self._flightDate)]
00987         >> ',' >> (bsc::ureal_p)[storeACP(self._flightDate)]
00988         >> ',' >> (bsc::real_p)[storeETB(self._flightDate)]
00989         ;
00990
00991       time =
00992         bsc::lexeme_d[
00993             (hours_p)[bsc::assign_a(self._flightDate._itHours)]
00994          >> (minutes_p)[bsc::assign_a(self._flightDate._itMinutes)]
00995          >> !((seconds_p)[bsc::assign_a(self._flightDate._itSeconds)])
00996                                             ]
00997         ;
00998
00999       bucket_list = +( ',' >> bucket_details )
01000         ;
01001
01002       bucket_details =
01003         (bsc::ureal_p)[storeYieldUpperRange(self.
      _flightDate)]
01004         >> ':' >> (bsc::real_p)[storeBucketAvaibality(self
      ._flightDate)]
01005         >> ':' >> (uint1_3_p)[storeSeatIndex(self.
      _flightDate)];
01006
01007       segment_list = +( '/' >> segment )
01008         ;
01009
01010       segment = segment_key >> segment_cabin_list
01011         ;
01012
01013       segment_key = (airport_p)[storeSegmentBoardingPoint
      (self._flightDate)]
01014         >> ';' >> (airport_p)[storeSegmentOffPoint
      (self._flightDate)]
01015         ;
01016
01017       segment_cabin_list =
01018         +( ';' >> segment_cabin_key >> ','
01019            >> segment_cabin_details >> class_list >> family_cabin_list )
01020         ;
01021
01022       family_cabin_list =
01023         +( ';' >> family_cabin_details)
01024         ;
01025
01026       segment_cabin_key =
01027         (cabin_code_p)[storeSegmentCabinCode(
      self._flightDate)]
01028         ;
01029
01030       segment_cabin_details =
01031         (bsc::ureal_p)[storeSegmentCabinBookingCounter
      (self._flightDate)]
01032         ;
01033
01034       class_list = +( ',' >> class_key >> '|' >> class_details )
01035         ;
01036
01037       class_key = (class_code_p)[storeClassCode(self.
```

```
      _flightDate)]
01038         ;
01039
01040       parent_subclass_code =
01041         (class_code_p)[storeParentClassCode(
      self._flightDate)]
01042         >> (uint1_2_p)[storeParentSubclassCode(
      self._flightDate)]
01043         ;
01044
01045       class_protection =
01046         (bsc::ureal_p)[storeProtection(self._flightDate)]
01047         ;
01048
01049       class_nego =
01050         (bsc::ureal_p)[storeNego(self._flightDate)]
01051         ;
01052
01053       class_details = (uint1_2_p)[storeSubclassCode(
      self._flightDate)]
01054         >> ':' >> (bsc::ureal_p)[storeCumulatedProtection
      (self._flightDate)]
01055         >> ':' >> !( parent_subclass_code )
01056         >> ':' >> !( class_protection )
01057         >> ':' >> (bsc::ureal_p)[storeNoShow(self._flightDate)]
01058         >> ':' >> (bsc::ureal_p)[storeOverbooking(self.
      _flightDate)]
01059         >> ':' >> (bsc::ureal_p)[storeNbOfBkgs(self._flightDate)]
01060         >> ':' >> (bsc::ureal_p)[storeNbOfGroupBkgs(self.
      _flightDate)]
01061         >> ':' >> (bsc::ureal_p)[storeNbOfPendingGroupBkgs
      (self._flightDate)]
01062         >> ':' >> (bsc::ureal_p)[storeNbOfStaffBkgs(self.
      _flightDate)]
01063         >> ':' >> (bsc::ureal_p)[storeNbOfWLBkgs(self.
      _flightDate)]
01064         >> ':' >> (bsc::ureal_p)[storeClassETB(self._flightDate)]
01065         >> ':' >> !( class_nego )
01066         >> ':' >> (bsc::real_p)[storeClassAvailability(
      self._flightDate)]
01067         >> ':' >> (bsc::real_p)[storeSegmentAvailability
      (self._flightDate)]
01068         >> ':' >> (bsc::real_p)[storeRevenueAvailability
      (self._flightDate)]
01069         ;
01070
01071       family_cabin_details =
01072         (family_code_p)[storeFamilyCode(self.
      _flightDate)]
01073         >> ';'
01074         >> (class_code_list_p)[storeFClasses(self
      ._flightDate)]
01075         ;
01076
01077       // BOOST_SPIRIT_DEBUG_NODE (InventoryParser);
01078       BOOST_SPIRIT_DEBUG_NODE (flight_date_list);
01079       BOOST_SPIRIT_DEBUG_NODE (not_to_be_parsed);
01080       BOOST_SPIRIT_DEBUG_NODE (flight_date);
01081       BOOST_SPIRIT_DEBUG_NODE (flight_date_end);
01082       BOOST_SPIRIT_DEBUG_NODE (flight_key);
01083       BOOST_SPIRIT_DEBUG_NODE (airline_code);
01084       BOOST_SPIRIT_DEBUG_NODE (flight_number);
01085       BOOST_SPIRIT_DEBUG_NODE (flight_type_code);
01086       BOOST_SPIRIT_DEBUG_NODE (flight_visibility_code);
01087       BOOST_SPIRIT_DEBUG_NODE (date);
01088       BOOST_SPIRIT_DEBUG_NODE (leg_list);
01089       BOOST_SPIRIT_DEBUG_NODE (leg);
01090       BOOST_SPIRIT_DEBUG_NODE (operating_leg_details);
01091       BOOST_SPIRIT_DEBUG_NODE (leg_key);
01092       BOOST_SPIRIT_DEBUG_NODE (leg_details);
01093       BOOST_SPIRIT_DEBUG_NODE (leg_cabin_list);
01094       BOOST_SPIRIT_DEBUG_NODE (leg_cabin_details);
01095       BOOST_SPIRIT_DEBUG_NODE (bucket_list);
01096       BOOST_SPIRIT_DEBUG_NODE (bucket_details);
01097       BOOST_SPIRIT_DEBUG_NODE (time);
01098       BOOST_SPIRIT_DEBUG_NODE (segment_list);
01099       BOOST_SPIRIT_DEBUG_NODE (segment);
01100       BOOST_SPIRIT_DEBUG_NODE (segment_key);
01101       BOOST_SPIRIT_DEBUG_NODE (full_segment_cabin_details);
01102       BOOST_SPIRIT_DEBUG_NODE (segment_cabin_list);
01103       BOOST_SPIRIT_DEBUG_NODE (segment_cabin_key);
01104       BOOST_SPIRIT_DEBUG_NODE (segment_cabin_details);
01105       BOOST_SPIRIT_DEBUG_NODE (class_list);
01106       BOOST_SPIRIT_DEBUG_NODE (class_key);
01107       BOOST_SPIRIT_DEBUG_NODE (parent_subclass_code);
01108       BOOST_SPIRIT_DEBUG_NODE (class_protection);
01109       BOOST_SPIRIT_DEBUG_NODE (class_nego);
```

```
01110       BOOST_SPIRIT_DEBUG_NODE (class_details);
01111       BOOST_SPIRIT_DEBUG_NODE (family_cabin_list);
01112       BOOST_SPIRIT_DEBUG_NODE (family_cabin_details);
01113     }
01114
01115     // ////////////////////////////////////////////////////////////////////
01116     template<typename ScannerT>
01117     bsc::rule<ScannerT> const&
01118     InventoryParser::definition<ScannerT>::start
      () const {
01119       return flight_date_list;
01120     }
01121   }
01122
01123
01125   //
01126   //  Entry class for the file parser
01127   //
01129
01130   // ////////////////////////////////////////////////////////////////////////
01131   InventoryFileParser::
01132   InventoryFileParser (stdair::BomRoot& ioBomRoot, const
      std::string& iFilename)
01133     : _filename (iFilename), _bomRoot (ioBomRoot),
01134       _nbOfFlights (0) {
01135     init();
01136   }
01137
01138   // ////////////////////////////////////////////////////////////////////////
01139   void InventoryFileParser::init() {
01140     // Open the file
01141     _startIterator = iterator_t (_filename);
01142
01143     // Check the filename exists and can be open
01144     if (!_startIterator) {
01145       std::ostringstream oMessage;
01146       oMessage << "The file " << _filename << " can not be open.";
01147       STDAIR_LOG_ERROR (oMessage.str());
01148       throw InventoryInputFileNotFoundException
       (oMessage.str());
01149     }
01150
01151     // Create an EOF iterator
01152     _endIterator = _startIterator.make_end();
01153   }
01154
01155   // ////////////////////////////////////////////////////////////////////////
01156   bool InventoryFileParser::buildInventory (
      ) {
01157     bool oResult = false;
01158
01159     STDAIR_LOG_DEBUG ("Parsing inventory input file: " << _filename);
01160
01161     // Initialise the parser (grammar) with the helper/staging structure.
01162     InventoryParserHelper::InventoryParser
       lInventoryParser (_bomRoot,
01163                                                          _flightDate,
01164                                                          _nbOfFlights);
01165
01166     // Launch the parsing of the file and, thanks to the doEndFlightDate
01167     // call-back structure, the building of the whole Inventory BOM
01168     // (i.e., including Inventory, FlightDate, LegDate, SegmentDate, etc.)
01169     bsc::parse_info<iterator_t> info = bsc::parse (_startIterator, _endIterator
      ,
01170                                                    lInventoryParser,
01171                                                    bsc::space_p - bsc::eol_p);
01172
01173     // Retrieves whether or not the parsing was successful
01174     oResult = info.hit;
01175
01176     const std::string hasBeenFullyReadStr = (info.full == true)?"":"not ";
01177     if (oResult == true) {
01178       STDAIR_LOG_DEBUG ("Parsing of inventory input file: " << _filename
01179                         << " succeeded: read " << info.length
01180                         << " characters. The input file has "
01181                         << hasBeenFullyReadStr
01182                         << "been fully read. Stop point: " << info.stop);
01183
01184     } else {
01185       STDAIR_LOG_ERROR ("Parsing of inventory input file: " << _filename
01186                         << " failed: read " << info.length
01187                         << " characters. The input file has "
01188                         << hasBeenFullyReadStr
01189                         << "been fully read. Stop point: " << info.stop);
01190       throw InventoryFileParsingFailedException
      ("Parsing of inventory input file"
01191                                                ": " + _filename + " failed");
```

```
01192     }
01193
01194     return oResult;
01195   }
01196
01197 }
```

23.151 airinv/command/InventoryParserHelper.hpp File Reference

```
#include <string>
#include <stdair/command/CmdAbstract.hpp>
#include <airinv/AIRINV_Types.hpp>
#include <airinv/basic/BasParserTypes.hpp>
#include <airinv/bom/FlightDateStruct.hpp>
```

Classes

- struct AIRINV::InventoryParserHelper::ParserSemanticAction
- struct AIRINV::InventoryParserHelper::storeSnapshotDate
- struct AIRINV::InventoryParserHelper::storeAirlineCode
- struct AIRINV::InventoryParserHelper::storeFlightNumber
- struct AIRINV::InventoryParserHelper::storeFlightDate
- struct AIRINV::InventoryParserHelper::storeFlightTypeCode
- struct AIRINV::InventoryParserHelper::storeFlightVisibilityCode
- struct AIRINV::InventoryParserHelper::storeLegBoardingPoint
- struct AIRINV::InventoryParserHelper::storeLegOffPoint
- struct AIRINV::InventoryParserHelper::storeOperatingAirlineCode
- struct AIRINV::InventoryParserHelper::storeOperatingFlightNumber
- struct AIRINV::InventoryParserHelper::storeBoardingDate
- struct AIRINV::InventoryParserHelper::storeBoardingTime
- struct AIRINV::InventoryParserHelper::storeOffDate
- struct AIRINV::InventoryParserHelper::storeOffTime
- struct AIRINV::InventoryParserHelper::storeLegCabinCode
- struct AIRINV::InventoryParserHelper::storeSaleableCapacity
- struct AIRINV::InventoryParserHelper::storeAU
- struct AIRINV::InventoryParserHelper::storeUPR
- struct AIRINV::InventoryParserHelper::storeBookingCounter
- struct AIRINV::InventoryParserHelper::storeNAV
- struct AIRINV::InventoryParserHelper::storeGAV
- struct AIRINV::InventoryParserHelper::storeACP
- struct AIRINV::InventoryParserHelper::storeETB
- struct AIRINV::InventoryParserHelper::storeYieldUpperRange
- struct AIRINV::InventoryParserHelper::storeBucketAvaibality
- struct AIRINV::InventoryParserHelper::storeSeatIndex
- struct AIRINV::InventoryParserHelper::storeSegmentBoardingPoint
- struct AIRINV::InventoryParserHelper::storeSegmentOffPoint
- struct AIRINV::InventoryParserHelper::storeSegmentCabinCode
- struct AIRINV::InventoryParserHelper::storeSegmentCabinBookingCounter
- struct AIRINV::InventoryParserHelper::storeClassCode
- struct AIRINV::InventoryParserHelper::storeSubclassCode
- struct AIRINV::InventoryParserHelper::storeParentClassCode
- struct AIRINV::InventoryParserHelper::storeParentSubclassCode
- struct AIRINV::InventoryParserHelper::storeCumulatedProtection
- struct AIRINV::InventoryParserHelper::storeProtection
- struct AIRINV::InventoryParserHelper::storeNego

- struct AIRINV::InventoryParserHelper::storeNoShow
- struct AIRINV::InventoryParserHelper::storeOverbooking
- struct AIRINV::InventoryParserHelper::storeNbOfBkgs
- struct AIRINV::InventoryParserHelper::storeNbOfGroupBkgs
- struct AIRINV::InventoryParserHelper::storeNbOfPendingGroupBkgs
- struct AIRINV::InventoryParserHelper::storeNbOfStaffBkgs
- struct AIRINV::InventoryParserHelper::storeNbOfWLBkgs
- struct AIRINV::InventoryParserHelper::storeClassETB
- struct AIRINV::InventoryParserHelper::storeClassAvailability
- struct AIRINV::InventoryParserHelper::storeSegmentAvailability
- struct AIRINV::InventoryParserHelper::storeRevenueAvailability
- struct AIRINV::InventoryParserHelper::storeFamilyCode
- struct AIRINV::InventoryParserHelper::storeFClasses
- struct AIRINV::InventoryParserHelper::doEndFlightDate
- struct AIRINV::InventoryParserHelper::InventoryParser
- struct AIRINV::InventoryParserHelper::InventoryParser::definition< ScannerT >
- class AIRINV::InventoryFileParser

Namespaces

- namespace stdair

  Forward declarations.
- namespace AIRINV
- namespace AIRINV::InventoryParserHelper

23.152 InventoryParserHelper.hpp

```
00001 #ifndef __AIRINV_CMD_INVENTORYPARSERHELPER_HPP
00002 #define __AIRINV_CMD_INVENTORYPARSERHELPER_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/command/CmdAbstract.hpp>
00011 // Airinv
00012 #include <airinv/AIRINV_Types.hpp>
00013 #include <airinv/basic/BasParserTypes.hpp>
00014 #include <airinv/bom/FlightDateStruct.hpp>
00015
00016 // Forward declarations
00017 namespace stdair {
00018   class BomRoot;
00019 }
00020
00021 namespace AIRINV {
00022
00023   namespace InventoryParserHelper {
00024
00025     // //////////////////////////////////////////////////////////////////
00026     //  Semantic actions
00027     // //////////////////////////////////////////////////////////////////
00029     struct ParserSemanticAction {
00031       ParserSemanticAction (FlightDateStruct
      &);
00033       FlightDateStruct& _flightDate;
00034     };
00035
00037     struct storeSnapshotDate : public ParserSemanticAction
      {
00039       storeSnapshotDate (FlightDateStruct&);
00041       void operator() (iterator_t iStr, iterator_t
      iStrEnd) const;
00042     };
00043
00045     struct storeAirlineCode : public ParserSemanticAction
      {
```

```
00047       storeAirlineCode (FlightDateStruct&);
00049       void operator() (iterator_t iStr, iterator_t
      iStrEnd) const;
00050     };
00051
00053     struct storeFlightNumber : public ParserSemanticAction
      {
00055       storeFlightNumber (FlightDateStruct&);
00057       void operator() (unsigned int iNumber) const;
00058     };
00059
00061     struct storeFlightDate : public ParserSemanticAction
      {
00063       storeFlightDate (FlightDateStruct&);
00065       void operator() (iterator_t iStr, iterator_t
      iStrEnd) const;
00066     };
00067
00069     struct storeFlightTypeCode : public ParserSemanticAction
      {
00071       storeFlightTypeCode (FlightDateStruct&
      );
00073       void operator() (iterator_t iStr, iterator_t
      iStrEnd) const;
00074     };
00075
00077     struct storeFlightVisibilityCode : public
      ParserSemanticAction {
00079       storeFlightVisibilityCode (FlightDateStruct
      &);
00081       void operator() (iterator_t iStr, iterator_t
      iStrEnd) const;
00082     };
00083
00085     struct storeLegBoardingPoint : public
      ParserSemanticAction {
00087       storeLegBoardingPoint (FlightDateStruct
      &);
00089       void operator() (iterator_t iStr, iterator_t
      iStrEnd) const;
00090     };
00091
00093     struct storeLegOffPoint : public ParserSemanticAction
      {
00095       storeLegOffPoint (FlightDateStruct&);
00097       void operator() (iterator_t iStr, iterator_t
      iStrEnd) const;
00098     };
00099
00101     struct storeOperatingAirlineCode : public
      ParserSemanticAction {
00103       storeOperatingAirlineCode (FlightDateStruct
      &);
00105       void operator() (iterator_t iStr, iterator_t
      iStrEnd) const;
00106     };
00107
00109     struct storeOperatingFlightNumber : public
      ParserSemanticAction {
00111       storeOperatingFlightNumber (FlightDateStruct
      &);
00113       void operator() (unsigned int iNumber) const;
00114     };
00115
00117     struct storeBoardingDate : public ParserSemanticAction
      {
00119       storeBoardingDate (FlightDateStruct&);
00121       void operator() (iterator_t iStr, iterator_t
      iStrEnd) const;
00122     };
00123
00125     struct storeBoardingTime : public ParserSemanticAction
      {
00127       storeBoardingTime (FlightDateStruct&);
00129       void operator() (iterator_t iStr, iterator_t
      iStrEnd) const;
00130     };
00131
00133     struct storeOffDate : public ParserSemanticAction
      {
00135       storeOffDate (FlightDateStruct&);
00137       void operator() (iterator_t iStr, iterator_t
      iStrEnd) const;
00138     };
00139
00141     struct storeOffTime : public ParserSemanticAction
      {
```

```
00143       storeOffTime (FlightDateStruct&);
00145       void operator() (iterator_t iStr, iterator_t
     iStrEnd) const;
00146     };
00147
00149     struct storeLegCabinCode : public ParserSemanticAction
     {
00151       storeLegCabinCode (FlightDateStruct&);
00153       void operator() (char iChar) const;
00154     };
00155
00157     struct storeSaleableCapacity : public
    ParserSemanticAction {
00159       storeSaleableCapacity (FlightDateStruct
    &);
00161       void operator() (double iReal) const;
00162     };
00163
00165     struct storeAU : public ParserSemanticAction {
00167       storeAU (FlightDateStruct&);
00169       void operator() (double iReal) const;
00170     };
00171
00173     struct storeUPR : public ParserSemanticAction {
00175       storeUPR (FlightDateStruct&);
00177       void operator() (double iReal) const;
00178     };
00179
00181     struct storeBookingCounter : public ParserSemanticAction
     {
00183       storeBookingCounter (FlightDateStruct&
    );
00185       void operator() (double iReal) const;
00186     };
00187
00189     struct storeNAV : public ParserSemanticAction {
00191       storeNAV (FlightDateStruct&);
00193       void operator() (double iReal) const;
00194     };
00195
00197     struct storeGAV : public ParserSemanticAction {
00199       storeGAV (FlightDateStruct&);
00201       void operator() (double iReal) const;
00202     };
00203
00205     struct storeACP : public ParserSemanticAction {
00207       storeACP (FlightDateStruct&);
00209       void operator() (double iReal) const;
00210     };
00211
00213     struct storeETB : public ParserSemanticAction {
00215       storeETB (FlightDateStruct&);
00217       void operator() (double iReal) const;
00218     };
00219
00221     struct storeYieldUpperRange : public
    ParserSemanticAction {
00223       storeYieldUpperRange (FlightDateStruct
    &);
00225       void operator() (double iReal) const;
00226     };
00227
00229     struct storeBucketAvaibality : public
    ParserSemanticAction {
00231       storeBucketAvaibality (FlightDateStruct
    &);
00233       void operator() (double iReal) const;
00234     };
00235
00237     struct storeSeatIndex : public ParserSemanticAction
     {
00239       storeSeatIndex (FlightDateStruct&);
00241       void operator() (double iReal) const;
00242     };
00243
00245     struct storeSegmentBoardingPoint : public
    ParserSemanticAction {
00247       storeSegmentBoardingPoint (FlightDateStruct
    &);
00249       void operator() (iterator_t iStr, iterator_t
     iStrEnd) const;
00250     };
00251
00253     struct storeSegmentOffPoint : public
    ParserSemanticAction {
00255       storeSegmentOffPoint (FlightDateStruct
    &);
```

```
00257       void operator() (iterator_t iStr, iterator_t
      iStrEnd) const;
00258     };
00259
00261     struct storeSegmentCabinCode : public
      ParserSemanticAction {
00263       storeSegmentCabinCode (FlightDateStruct
      &);
00265       void operator() (char iChar) const;
00266     };
00267
00269     struct storeSegmentCabinBookingCounter :
      public ParserSemanticAction {
00271       storeSegmentCabinBookingCounter (
      FlightDateStruct&);
00273       void operator() (double iReal) const;
00274     };
00275
00277     struct storeClassCode : public ParserSemanticAction
      {
00279       storeClassCode (FlightDateStruct&);
00281       void operator() (char iChar) const;
00282     };
00283
00285     struct storeSubclassCode : public ParserSemanticAction
      {
00287       storeSubclassCode (FlightDateStruct&);
00289       void operator() (unsigned int iNumber) const;
00290     };
00291
00293     struct storeParentClassCode : public
      ParserSemanticAction {
00295       storeParentClassCode (FlightDateStruct
      &);
00297       void operator() (char iChar) const;
00298     };
00299
00301     struct storeParentSubclassCode : public
      ParserSemanticAction {
00303       storeParentSubclassCode (FlightDateStruct
      &);
00305       void operator() (unsigned int iNumber) const;
00306     };
00307
00309     struct storeCumulatedProtection : public
      ParserSemanticAction {
00311       storeCumulatedProtection (FlightDateStruct
      &);
00313       void operator() (double iReal) const;
00314     };
00315
00317     struct storeProtection : public ParserSemanticAction
      {
00319       storeProtection (FlightDateStruct&);
00321       void operator() (double iReal) const;
00322     };
00323
00325     struct storeNego : public ParserSemanticAction
      {
00327       storeNego (FlightDateStruct&);
00329       void operator() (double iReal) const;
00330     };
00331
00333     struct storeNoShow : public ParserSemanticAction
      {
00335       storeNoShow (FlightDateStruct&);
00337       void operator() (double iReal) const;
00338     };
00339
00341     struct storeOverbooking : public ParserSemanticAction
      {
00343       storeOverbooking (FlightDateStruct&);
00345       void operator() (double iReal) const;
00346     };
00347
00349     struct storeNbOfBkgs : public ParserSemanticAction
      {
00351       storeNbOfBkgs (FlightDateStruct&);
00353       void operator() (double iReal) const;
00354     };
00355
00357     struct storeNbOfGroupBkgs : public ParserSemanticAction
      {
00359       storeNbOfGroupBkgs (FlightDateStruct&);
00361       void operator() (double iReal) const;
00362     };
00363
```

```
00365     struct storeNbOfPendingGroupBkgs : public
      ParserSemanticAction {
00367       storeNbOfPendingGroupBkgs (FlightDateStruct
      &);
00369       void operator() (double iReal) const;
00370     };
00371
00373     struct storeNbOfStaffBkgs : public ParserSemanticAction
       {
00375       storeNbOfStaffBkgs (FlightDateStruct&);
00377       void operator() (double iReal) const;
00378     };
00379
00382     struct storeNbOfWLBkgs : public ParserSemanticAction
       {
00384       storeNbOfWLBkgs (FlightDateStruct&);
00386       void operator() (double iReal) const;
00387     };
00388
00390     struct storeClassETB : public ParserSemanticAction
       {
00392       storeClassETB (FlightDateStruct&);
00394       void operator() (double iReal) const;
00395     };
00396
00399     struct storeClassAvailability : public
      ParserSemanticAction {
00401       storeClassAvailability (FlightDateStruct
      &);
00403       void operator() (double iReal) const;
00404     };
00405
00408     struct storeSegmentAvailability : public
      ParserSemanticAction {
00410       storeSegmentAvailability (FlightDateStruct
      &);
00412       void operator() (double iReal) const;
00413     };
00414
00417     struct storeRevenueAvailability : public
      ParserSemanticAction {
00419       storeRevenueAvailability (FlightDateStruct
      &);
00421       void operator() (double iReal) const;
00422     };
00423
00425     struct storeFamilyCode : public ParserSemanticAction
       {
00427       storeFamilyCode (FlightDateStruct&);
00429       void operator() (int iCode) const;
00430     };
00431
00433     struct storeFClasses : public ParserSemanticAction
       {
00435       storeFClasses (FlightDateStruct&);
00437       void operator() (iterator_t iStr, iterator_t
       iStrEnd) const;
00438     };
00439
00441     struct doEndFlightDate : public ParserSemanticAction
       {
00443       doEndFlightDate (stdair::BomRoot&, FlightDateStruct
      &,
00444                        unsigned int&);
00446       void operator() (iterator_t iStr, iterator_t
       iStrEnd) const;
00448       stdair::BomRoot& _bomRoot;
00449       unsigned int& _nbOfFlights;
00450     };
00451
00452
00454     //
00455     //  (Boost Spirit) Grammar Definition
00456     //
00458
00470     struct InventoryParser :
00471       public boost::spirit::classic::grammar<InventoryParser> {
00472
00473       InventoryParser (stdair::BomRoot&, FlightDateStruct
      &, unsigned int&);
00474
00475       template <typename ScannerT>
00476       struct definition {
00477         definition (InventoryParser const& self);
00478
00479         // Instantiation of rules
00480         boost::spirit::classic::rule<ScannerT> flight_date_list
```

```
      ,
00481           not_to_be_parsed,
00482           flight_date, flight_date_end, flight_key
      , airline_code, flight_number,
00483           flight_type_code, flight_visibility_code
      ,
00484           date, leg_list, leg, operating_leg_details
      , leg_key, leg_details,
00485           leg_cabin_list, leg_cabin_details,
00486           bucket_list, bucket_details,
00487           time, segment_list, segment, segment_key
      , full_segment_cabin_details,
00488           segment_cabin_list, segment_cabin_key
      , segment_cabin_details,
00489           class_list, class_key, parent_subclass_code
      ,
00490           class_protection, class_nego, class_details
      ,
00491           family_cabin_list, family_cabin_details
      ;
00492
00494         boost::spirit::classic::rule<ScannerT> const& start() const;
00495       };
00496
00497       // Parser Context
00498       stdair::BomRoot& _bomRoot;
00499       FlightDateStruct& _flightDate;
00500       unsigned int& _nbOfFlights;
00501     };
00502
00503   }
00504
00505
00507   //
00508   //  Entry class for the file parser
00509   //
00511
00516   class InventoryFileParser : public stdair::CmdAbstract {
00517   public:
00519     InventoryFileParser (stdair::BomRoot&,
00520                          const stdair::Filename_T& iInventoryInputFilename);
00521
00523     bool buildInventory ();
00524
00525   private:
00527     void init();
00528
00529   private:
00530     // Attributes
00532     stdair::Filename_T _filename;
00533
00535     iterator_t _startIterator;
00536
00538     iterator_t _endIterator;
00539
00541     stdair::BomRoot& _bomRoot;
00542
00544     FlightDateStruct _flightDate;
00545
00547     unsigned int _nbOfFlights;
00548   };
00549
00550 }
00551 #endif // __AIRINV_CMD_INVENTORYPARSERHELPER_HPP
```

23.153 airinv/command/ScheduleParser.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/service/Logger.hpp>
#include <airinv/command/ScheduleParserHelper.hpp>
#include <airinv/command/ScheduleParser.hpp>
#include <airinv/command/InventoryManager.hpp>
```

Namespaces

- namespace AIRINV

23.154 ScheduleParser.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasFileMgr.hpp>
00009 #include <stdair/bom/BomRoot.hpp>
00010 #include <stdair/service/Logger.hpp>
00011 // Airinv
00012 #include <airinv/command/ScheduleParserHelper.hpp
      >
00013 #include <airinv/command/ScheduleParser.hpp>
00014 #include <airinv/command/InventoryManager.hpp
      >
00015
00016 namespace AIRINV {
00017
00018   // ////////////////////////////////////////////////////////////////////
00019   void ScheduleParser::
00020   generateInventories (const stdair::ScheduleFilePath&
      iScheduleFilename,
00021                        stdair::BomRoot& ioBomRoot) {
00022
00023     const stdair::Filename_T lFilename = iScheduleFilename.name();
00024
00025     // Check that the file path given as input corresponds to an actual file
00026     bool doesExistAndIsReadable =
00027       stdair::BasFileMgr::doesExistAndIsReadable (lFilename);
00028     if (doesExistAndIsReadable == false) {
00029       std::ostringstream oMessage;
00030       oMessage << "The schedule input file, '" << lFilename
00031                << "', can not be retrieved on the file-system";
00032       STDAIR_LOG_ERROR (oMessage.str());
00033       throw ScheduleInputFileNotFoundException
       (oMessage.str());
00034     }
00035
00036     // Initialise the Flight-Period file parser.
00037     FlightPeriodFileParser lFlightPeriodParser (ioBomRoot
      , lFilename);
00038
00039     // Parse the CSV-formatted schedule input file, and generate the
00040     // corresponding Inventories for the airlines.
00041     lFlightPeriodParser.generateInventories ();
00042
00043   }
00044
00045 }
```

23.155 airinv/command/ScheduleParser.hpp File Reference

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

Classes

- class AIRINV::ScheduleParser

  Class wrapping the parser entry point.

Namespaces

- namespace stdair

Forward declarations.

- namespace AIRINV

23.156 ScheduleParser.hpp

```
00001 #ifndef __AIRINV_CMD_SCHEDULEPARSER_HPP
00002 #define __AIRINV_CMD_SCHEDULEPARSER_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009 #include <stdair/stdair_file.hpp>
00010 #include <stdair/command/CmdAbstract.hpp>
00011
00013 namespace stdair {
00014   class BomRoot;
00015 }
00016
00017 namespace AIRINV {
00018
00022   class ScheduleParser : public stdair::CmdAbstract {
00023   public:
00032     static void generateInventories (const
      stdair::ScheduleFilePath& iScheduleFilename,
00033                                      stdair::BomRoot&);
00034   };
00035 }
00036 #endif // __AIRINV_CMD_SCHEDULEPARSER_HPP
```

23.157 airinv/command/ScheduleParserHelper.cpp File Reference

```
#include <cassert>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/stdair_types.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/service/Logger.hpp>
#include <airinv/command/InventoryGenerator.hpp>
#include <airinv/command/ScheduleParserHelper.hpp>
```

Namespaces

- namespace AIRINV
- namespace AIRINV::ScheduleParserHelper

Functions

- repeat_p_t AIRINV::ScheduleParserHelper::airline_code_p (chset_t("0-9A-Z").derived(), 2, 3)
- bounded1_4_p_t AIRINV::ScheduleParserHelper::flight_number_p (uint1_4_p.derived(), 0u, 9999u)
- bounded4_p_t AIRINV::ScheduleParserHelper::year_p (uint4_p.derived(), 2000u, 2099u)
- bounded2_p_t AIRINV::ScheduleParserHelper::month_p (uint2_p.derived(), 1u, 12u)
- bounded2_p_t AIRINV::ScheduleParserHelper::day_p (uint2_p.derived(), 1u, 31u)
- repeat_p_t AIRINV::ScheduleParserHelper::dow_p (chset_t("0-1").derived().derived(), 7, 7)
- repeat_p_t AIRINV::ScheduleParserHelper::airport_p (chset_t("0-9A-Z").derived(), 3, 3)
- bounded2_p_t AIRINV::ScheduleParserHelper::hours_p (uint2_p.derived(), 0u, 23u)
- bounded2_p_t AIRINV::ScheduleParserHelper::minutes_p (uint2_p.derived(), 0u, 59u)
- bounded2_p_t AIRINV::ScheduleParserHelper::seconds_p (uint2_p.derived(), 0u, 59u)
- chset_t AIRINV::ScheduleParserHelper::cabin_code_p ("A-Z")
- repeat_p_t AIRINV::ScheduleParserHelper::key_p (chset_t("0-9A-Z").derived(), 1, 10)
- repeat_p_t AIRINV::ScheduleParserHelper::class_code_list_p (chset_t("A-Z").derived(), 1, 26)

Variables

- int1_p_t AIRINV::ScheduleParserHelper::int1_p
- uint2_p_t AIRINV::ScheduleParserHelper::uint2_p
- uint4_p_t AIRINV::ScheduleParserHelper::uint4_p
- uint1_4_p_t AIRINV::ScheduleParserHelper::uint1_4_p
- int1_p_t AIRINV::ScheduleParserHelper::family_code_p

23.158 ScheduleParserHelper.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // StdAir
00007 #include <stdair/stdair_exceptions.hpp>
00008 #include <stdair/stdair_types.hpp>
00009 #include <stdair/bom/BomRoot.hpp>
00010 #include <stdair/service/Logger.hpp>
00011 // AirInv
00012 #include <airinv/command/InventoryGenerator.hpp
      >
00013 // #define BOOST_SPIRIT_DEBUG
00014 #include <airinv/command/ScheduleParserHelper.hpp
      >
00015
00016 //
00017 namespace bsc = boost::spirit::classic;
00018
00019 namespace AIRINV {
00020
00021   namespace ScheduleParserHelper {
00022
00023     // ////////////////////////////////////////////////////////////////
00024     //  Semantic actions
00025     // ////////////////////////////////////////////////////////////////
00026
00027     ParserSemanticAction::
00028     ParserSemanticAction (FlightPeriodStruct
      & ioFlightPeriod)
00029       : _flightPeriod (ioFlightPeriod) {
00030     }
00031
00032     // ////////////////////////////////////////////////////////////////
00033     storeAirlineCode::
00034     storeAirlineCode (FlightPeriodStruct&
       ioFlightPeriod)
00035       : ParserSemanticAction (ioFlightPeriod) {
00036     }
00037
00038     // ////////////////////////////////////////////////////////////////
00039     void storeAirlineCode::operator() (iterator_t
       iStr,
00040                                        iterator_t iStrEnd) const {
00041       const stdair::AirlineCode_T lAirlineCode (iStr, iStrEnd);
00042       _flightPeriod._airlineCode = lAirlineCode;
00043
00044       // As that's the beginning of a new flight, the list of legs
00045       // must be reset
00046       _flightPeriod._legList.clear();
00047     }
00048
00049     // ////////////////////////////////////////////////////////////////
00050     storeFlightNumber::
00051     storeFlightNumber (FlightPeriodStruct
      & ioFlightPeriod)
00052       : ParserSemanticAction (ioFlightPeriod) {
00053     }
00054
00055     // ////////////////////////////////////////////////////////////////
00056     void storeFlightNumber::operator() (unsigned
      int iNumber) const {
00057       _flightPeriod._flightNumber = iNumber;
00058     }
00059
00060     // ////////////////////////////////////////////////////////////////
00061     storeDateRangeStart::
00062     storeDateRangeStart (FlightPeriodStruct
      & ioFlightPeriod)
00063       : ParserSemanticAction (ioFlightPeriod) {
```

```
00064     }
00065
00066     // ////////////////////////////////////////////////////////////////////
00067     void storeDateRangeStart::operator() (
      iterator_t iStr,
00068                                           iterator_t iStrEnd) const {
00069       _flightPeriod._dateRangeStart = _flightPeriod
      .getDate();
00070
00071       // Reset the number of seconds
00072       _flightPeriod._itSeconds = 0;
00073     }
00074
00075     // ////////////////////////////////////////////////////////////////////
00076     storeDateRangeEnd::
00077     storeDateRangeEnd (FlightPeriodStruct
      & ioFlightPeriod)
00078       : ParserSemanticAction (ioFlightPeriod) {
00079     }
00080
00081     // ////////////////////////////////////////////////////////////////////
00082     void storeDateRangeEnd::operator() (
      iterator_t iStr,
00083                                         iterator_t iStrEnd) const {
00084       // As a Boost date period (DatePeriod_T) defines the last day of
00085       // the period to be end-date - one day, we have to add one day to that
00086       // end date before.
00087       const stdair::DateOffset_T oneDay (1);
00088       _flightPeriod._dateRangeEnd = _flightPeriod
      .getDate() + oneDay;
00089
00090       // Transform the date pair (i.e., the date range) into a date period
00091       _flightPeriod._dateRange =
00092         stdair::DatePeriod_T (_flightPeriod._dateRangeStart
      ,
00093                               _flightPeriod._dateRangeEnd
      );
00094
00095       // Reset the number of seconds
00096       _flightPeriod._itSeconds = 0;
00097
00098       // Set the (default) operating airline and flight number
00099       _flightPeriod._itLeg._airlineCode =
      _flightPeriod._airlineCode;
00100       _flightPeriod._itLeg._flightNumber =
      _flightPeriod._flightNumber;
00101
00102     }
00103
00104     // ////////////////////////////////////////////////////////////////////
00105     storeDow::storeDow (FlightPeriodStruct&
       ioFlightPeriod)
00106       : ParserSemanticAction (ioFlightPeriod) {
00107     }
00108
00109     // ////////////////////////////////////////////////////////////////////
00110     void storeDow::operator() (iterator_t iStr,
      iterator_t iStrEnd) const {
00111       stdair::DOW_String_T lDow (iStr, iStrEnd);
00112       _flightPeriod._dow = lDow;
00113     }
00114
00115     // ////////////////////////////////////////////////////////////////////
00116     storeLegBoardingPoint::
00117     storeLegBoardingPoint (FlightPeriodStruct
      & ioFlightPeriod)
00118       : ParserSemanticAction (ioFlightPeriod) {
00119     }
00120
00121     // ////////////////////////////////////////////////////////////////////
00122     void storeLegBoardingPoint::operator() (
      iterator_t iStr,
00123                                             iterator_t iStrEnd) const
       {
00124       stdair::AirportCode_T lBoardingPoint (iStr, iStrEnd);
00125
00126       // If a leg has already been parsed, add it to the FlightPeriod
00127       if (_flightPeriod._legAlreadyDefined ==
      true) {
00128         _flightPeriod._legList.push_back (_flightPeriod
      ._itLeg);
00129       } else {
00130         _flightPeriod._legAlreadyDefined = true;
00131       }
00132
00133       // Set the (new) boarding point
00134       _flightPeriod._itLeg._boardingPoint =
```

```
      lBoardingPoint;
00135
00136       // As that's the beginning of a new leg, the list of cabins
00137       // must be reset
00138       _flightPeriod._itLeg._cabinList.clear();
00139
00140       // Add the airport code if it is not already stored in the airport lists
00141       _flightPeriod.addAirport (lBoardingPoint);
00142     }
00143
00144     // ////////////////////////////////////////////////////////////////////
00145     storeLegOffPoint::
00146     storeLegOffPoint (FlightPeriodStruct&
       ioFlightPeriod)
00147       : ParserSemanticAction (ioFlightPeriod) {
00148     }
00149
00150     // ////////////////////////////////////////////////////////////////////
00151     void storeLegOffPoint::operator() (iterator_t
       iStr,
00152                                        iterator_t iStrEnd) const {
00153       stdair::AirportCode_T lOffPoint (iStr, iStrEnd);
00154       _flightPeriod._itLeg._offPoint = lOffPoint;
00155
00156       // Add the airport code if it is not already stored in the airport lists
00157       _flightPeriod.addAirport (lOffPoint);
00158     }
00159
00160     // ////////////////////////////////////////////////////////////////////
00161     storeOperatingAirlineCode::
00162     storeOperatingAirlineCode (FlightPeriodStruct
      & ioFlightPeriod)
00163       : ParserSemanticAction (ioFlightPeriod) {
00164     }
00165
00166     // ////////////////////////////////////////////////////////////////////
00167     void storeOperatingAirlineCode::operator()
      (iterator_t iStr,
00168                                               iterator_t iStrEnd)
       const {
00169       const stdair::AirlineCode_T lAirlineCode (iStr, iStrEnd);
00170       if (lAirlineCode.size() == 2) {
00171         _flightPeriod._itLeg._airlineCode =
      lAirlineCode;
00172       }
00173       //STDAIR_LOG_DEBUG ("Airline code: " << lAirlineCode);
00174     }
00175
00176     // ////////////////////////////////////////////////////////////////////
00177     storeOperatingFlightNumber::
00178     storeOperatingFlightNumber (
      FlightPeriodStruct& ioFlightPeriod)
00179       : ParserSemanticAction (ioFlightPeriod) {
00180     }
00181
00182     // ////////////////////////////////////////////////////////////////////
00183     void storeOperatingFlightNumber::operator()
      (unsigned int iNumber) const {
00184       _flightPeriod._itLeg._flightNumber =
      iNumber;
00185       //STDAIR_LOG_DEBUG ("Flight number: " << iNumber);
00186     }
00187
00188     // ////////////////////////////////////////////////////////////////////
00189     storeBoardingTime::
00190     storeBoardingTime (FlightPeriodStruct
      & ioFlightPeriod)
00191       : ParserSemanticAction (ioFlightPeriod) {
00192     }
00193
00194     // ////////////////////////////////////////////////////////////////////
00195     void storeBoardingTime::operator() (
      iterator_t iStr,
00196                                         iterator_t iStrEnd) const {
00197       _flightPeriod._itLeg._boardingTime =
      _flightPeriod.getTime();
00198
00199       // Reset the number of seconds
00200       _flightPeriod._itSeconds = 0;
00201
00202       // Reset the date off-set
00203       _flightPeriod._dateOffset = 0;
00204     }
00205
00206     // ////////////////////////////////////////////////////////////////////
00207     storeOffTime::
00208     storeOffTime (FlightPeriodStruct&
```

```
      ioFlightPeriod)
00209       : ParserSemanticAction (ioFlightPeriod) {
00210     }
00211
00212     // ////////////////////////////////////////////////////////////////////
00213     void storeOffTime::operator() (iterator_t
       iStr,
00214                                    iterator_t iStrEnd) const {
00215       _flightPeriod._itLeg._offTime = _flightPeriod
      .getTime();
00216
00217       // Reset the number of seconds
00218       _flightPeriod._itSeconds = 0;
00219
00220       // As the boarding date off set is optional, it can be set only
00221       // afterwards, based on the staging date off-set value
00222       // (_flightPeriod._dateOffset).
00223       const stdair::DateOffset_T lDateOffset (_flightPeriod.
      _dateOffset);
00224       _flightPeriod._itLeg._boardingDateOffset
       = lDateOffset;
00225     }
00226
00227     // ////////////////////////////////////////////////////////////////////
00228     storeElapsedTime::
00229     storeElapsedTime (FlightPeriodStruct&
       ioFlightPeriod)
00230       : ParserSemanticAction (ioFlightPeriod) {
00231     }
00232
00233     // ////////////////////////////////////////////////////////////////////
00234     void storeElapsedTime::operator() (iterator_t
       iStr,
00235                                        iterator_t iStrEnd) const {
00236       _flightPeriod._itLeg._elapsed = _flightPeriod
      .getTime();
00237
00238       // Reset the number of seconds
00239       _flightPeriod._itSeconds = 0;
00240
00241       // As the boarding date off set is optional, it can be set only
00242       // afterwards, based on the staging date off-set value
00243       // (_flightPeriod._dateOffset).
00244       const stdair::DateOffset_T lDateOffset (_flightPeriod.
      _dateOffset);
00245       _flightPeriod._itLeg._offDateOffset =
      lDateOffset;
00246     }
00247
00248     // ////////////////////////////////////////////////////////////////////
00249     storeLegCabinCode::
00250     storeLegCabinCode (FlightPeriodStruct
      & ioFlightPeriod)
00251       : ParserSemanticAction (ioFlightPeriod) {
00252     }
00253
00254     // ////////////////////////////////////////////////////////////////////
00255     void storeLegCabinCode::operator() (char
      iChar) const {
00256       _flightPeriod._itLegCabin._cabinCode =
      iChar;
00257       //std::cout << "Cabin code: " << iChar << std::endl;
00258     }
00259
00260     // ////////////////////////////////////////////////////////////////////
00261     storeCapacity::
00262     storeCapacity (FlightPeriodStruct&
      ioFlightPeriod)
00263       : ParserSemanticAction (ioFlightPeriod) {
00264     }
00265
00266     // ////////////////////////////////////////////////////////////////////
00267     void storeCapacity::operator() (double iReal)
       const {
00268       _flightPeriod._itLegCabin._saleableCapacity
       = iReal;
00269       //std::cout << "Capacity: " << iReal << std::endl;
00270
00271       // The capacity is the last (according to the arrival order
00272       // within the schedule input file) detail of the leg cabin. Hence,
00273       // when a capacity is parsed, it means that the full cabin
00274       // details have already been parsed as well: the cabin can
00275       // thus be added to the leg.
00276       _flightPeriod._itLeg._cabinList.push_back (
      _flightPeriod._itLegCabin);
00277     }
00278
```

```
00279     // ////////////////////////////////////////////////////////////////////
00280     storeSegmentSpecificity::
00281     storeSegmentSpecificity (FlightPeriodStruct
      & ioFlightPeriod)
00282       : ParserSemanticAction (ioFlightPeriod) {
00283     }
00284
00285     // ////////////////////////////////////////////////////////////////////
00286     void storeSegmentSpecificity::operator()
      (char iChar) const {
00287       if (iChar == '0') {
00288         _flightPeriod._areSegmentDefinitionsSpecific
       = false;
00289       } else {
00290         _flightPeriod._areSegmentDefinitionsSpecific
       = true;
00291       }
00292
00293       // Do a few sanity checks: the two lists should get exactly the same
00294       // content (in terms of airport codes). The only difference is that one
00295       // is a STL set, and the other a STL vector.
00296       assert (_flightPeriod._airportList.size()
00297               == _flightPeriod._airportOrderedList
      .size());
00298       assert (_flightPeriod._airportList.size() >= 2);
00299
00300       // Since all the legs have now been parsed, we get all the airports
00301       // and the segments may be built.
00302       _flightPeriod.buildSegments();
00303     }
00304
00305     // ////////////////////////////////////////////////////////////////////
00306     storeSegmentBoardingPoint::
00307     storeSegmentBoardingPoint (FlightPeriodStruct
      & ioFlightPeriod)
00308       : ParserSemanticAction (ioFlightPeriod) {
00309     }
00310
00311     // ////////////////////////////////////////////////////////////////////
00312     void storeSegmentBoardingPoint::operator()
      (iterator_t iStr,
00313                                             iterator_t iStrEnd)
       const {
00314       stdair::AirportCode_T lBoardingPoint (iStr, iStrEnd);
00315       _flightPeriod._itSegment._boardingPoint
       = lBoardingPoint;
00316     }
00317
00318     // ////////////////////////////////////////////////////////////////////
00319     storeSegmentOffPoint::
00320     storeSegmentOffPoint (FlightPeriodStruct
      & ioFlightPeriod)
00321       : ParserSemanticAction (ioFlightPeriod) {
00322     }
00323
00324     // ////////////////////////////////////////////////////////////////////
00325     void storeSegmentOffPoint::operator() (
      iterator_t iStr,
00326                                            iterator_t iStrEnd) const
      {
00327       stdair::AirportCode_T lOffPoint (iStr, iStrEnd);
00328       _flightPeriod._itSegment._offPoint =
      lOffPoint;
00329     }
00330
00331     // ////////////////////////////////////////////////////////////////////
00332     storeSegmentCabinCode::
00333     storeSegmentCabinCode (FlightPeriodStruct
      & ioFlightPeriod)
00334       : ParserSemanticAction (ioFlightPeriod) {
00335     }
00336
00337     // ////////////////////////////////////////////////////////////////////
00338     void storeSegmentCabinCode::operator() (
      char iChar) const {
00339       _flightPeriod._itSegmentCabin._cabinCode
       = iChar;
00340     }
00341
00342     // ////////////////////////////////////////////////////////////////////
00343     storeClasses::
00344     storeClasses (FlightPeriodStruct&
      ioFlightPeriod)
00345       : ParserSemanticAction (ioFlightPeriod) {
00346     }
00347
00348     // ////////////////////////////////////////////////////////////////////
```

```
00349     void storeClasses::operator() (iterator_t
      iStr,
00350                                    iterator_t iStrEnd) const {
00351       std::string lClasses (iStr, iStrEnd);
00352       _flightPeriod._itSegmentCabin._itFareFamily
      ._classes = lClasses;
00353
00354       // The list of classes is the last (according to the arrival order
00355       // within the schedule input file) detail of the segment cabin. Hence,
00356       // when a list of classes is parsed, it means that the full segment
00357       // cabin details have already been parsed as well: the segment cabin
00358       // can thus be added to the segment.
00359       if (_flightPeriod._areSegmentDefinitionsSpecific
      == true) {
00360         _flightPeriod.addSegmentCabin (
      _flightPeriod._itSegment,
00361                                        _flightPeriod.
      _itSegmentCabin);
00362       } else {
00363         _flightPeriod.addSegmentCabin (
      _flightPeriod._itSegmentCabin);
00364       }
00365     }
00366
00367     // ////////////////////////////////////////////////////////////////////
00368     storeFamilyCode::
00369     storeFamilyCode (FlightPeriodStruct&
      ioFlightPeriod)
00370       : ParserSemanticAction (ioFlightPeriod) {
00371     }
00372
00373     // ////////////////////////////////////////////////////////////////////
00374     void storeFamilyCode::operator() (int iCode)
      const {
00375       std::ostringstream ostr;
00376       ostr << iCode;
00377       _flightPeriod._itSegmentCabin._itFareFamily
      ._familyCode = ostr.str();
00378     }
00379
00380     // ////////////////////////////////////////////////////////////////////
00381     storeFRAT5CurveKey::
00382     storeFRAT5CurveKey (FlightPeriodStruct
      & ioFlightPeriod)
00383       : ParserSemanticAction (ioFlightPeriod) {
00384     }
00385
00386     // ////////////////////////////////////////////////////////////////////
00387     void storeFRAT5CurveKey::operator() (
      iterator_t iStr,
00388                                         iterator_t iStrEnd) const {
00389       const std::string lKey (iStr, iStrEnd);
00390       _flightPeriod._itSegmentCabin._itFareFamily
      ._frat5CurveKey = lKey;
00391       //STDAIR_LOG_DEBUG ("FRAT5 key: " << lKey);
00392     }
00393
00394     // ////////////////////////////////////////////////////////////////////
00395     storeFFDisutilityCurveKey::
00396     storeFFDisutilityCurveKey (FlightPeriodStruct
      & ioFlightPeriod)
00397       : ParserSemanticAction (ioFlightPeriod) {
00398     }
00399
00400     // ////////////////////////////////////////////////////////////////////
00401     void storeFFDisutilityCurveKey::operator()
      (iterator_t iStr,
00402                                                 iterator_t iStrEnd)
      const {
00403       const std::string lKey (iStr, iStrEnd);
00404       _flightPeriod._itSegmentCabin._itFareFamily
      ._ffDisutilityCurveKey = lKey;
00405     }
00406
00407     // ////////////////////////////////////////////////////////////////////
00408     storeFClasses::
00409     storeFClasses (FlightPeriodStruct&
      ioFlightPeriod)
00410       : ParserSemanticAction (ioFlightPeriod) {
00411     }
00412
00413     // ////////////////////////////////////////////////////////////////////
00414     void storeFClasses::operator() (iterator_t
      iStr,
00415                                     iterator_t iStrEnd) const {
00416       std::string lClasses (iStr, iStrEnd);
00417       FareFamilyStruct lFareFamily (_flightPeriod.
```

```
      _itSegmentCabin._itFareFamily._familyCode
      , _flightPeriod._itSegmentCabin._itFareFamily
      ._frat5CurveKey, _flightPeriod._itSegmentCabin
      ._itFareFamily._ffDisutilityCurveKey, lClasses
      );
00418
00419       // The list of classes is the last (according to the arrival order
00420       // within the schedule input file) detail of the segment cabin. Hence,
00421       // when a list of classes is parsed, it means that the full segment
00422       // cabin details have already been parsed as well: the segment cabin
00423       // can thus be added to the segment.
00424       if (_flightPeriod._areSegmentDefinitionsSpecific
       == true) {
00425         _flightPeriod.addFareFamily (_flightPeriod
      ._itSegment,
00426                                      _flightPeriod._itSegmentCabin
      ,
00427                                      lFareFamily);
00428       } else {
00429         _flightPeriod.addFareFamily (_flightPeriod
      ._itSegmentCabin,
00430                                      lFareFamily);
00431       }
00432     }
00433
00434     // ////////////////////////////////////////////////////////////////////
00435     doEndFlight::
00436     doEndFlight (stdair::BomRoot& ioBomRoot,
00437                  FlightPeriodStruct& ioFlightPeriod)
00438       : ParserSemanticAction (ioFlightPeriod),
00439         _bomRoot (ioBomRoot) {
00440     }
00441
00442     // ////////////////////////////////////////////////////////////////////
00443     // void doEndFlight::operator() (char iChar) const {
00444     void doEndFlight::operator() (iterator_t
      iStr,
00445                                   iterator_t iStrEnd) const {
00446
00447       assert (_flightPeriod._legAlreadyDefined
      == true);
00448       _flightPeriod._legList.push_back (_flightPeriod
      ._itLeg);
00449
00450       // The lists of legs and cabins must be reset
00451       _flightPeriod._legAlreadyDefined = false;
00452       _flightPeriod._itLeg._cabinList.clear();
00453
00454       // DEBUG: Display the result
00455       STDAIR_LOG_DEBUG ("FlightPeriod: " << _flightPeriod.describe
      ());
00456
00457       // Create the FlightDate BOM objects, and potentially the intermediary
00458       // objects (e.g., Inventory).
00459       InventoryGenerator::createFlightDate (_bomRoot, _flightPeriod
      );
00460     }
00461
00462
00463     // ////////////////////////////////////////////////////////////////////
00464     //
00465     //  Utility Parsers
00466     //
00467     // ////////////////////////////////////////////////////////////////////
00469     int1_p_t int1_p;
00470
00472     uint2_p_t uint2_p;
00473
00475     uint4_p_t uint4_p;
00476
00478     uint1_4_p_t uint1_4_p;
00479
00481     repeat_p_t airline_code_p (chset_t("0-9A-Z")
      .derived(), 2, 3);
00482
00484     bounded1_4_p_t flight_number_p (uint1_4_p
      .derived(), 0u, 9999u);
00485
00487     bounded4_p_t year_p (uint4_p.derived(), 2000u,
      2099u);
00488
00490     bounded2_p_t month_p (uint2_p.derived(), 1u, 12u)
      ;
00491
00493     bounded2_p_t day_p (uint2_p.derived(), 1u, 31u);
00494
00496     repeat_p_t dow_p (chset_t("0-1").derived().derived(),
```

```
      7, 7);
00497
00499     repeat_p_t airport_p (chset_t("0-9A-Z").derived()
      , 3, 3);
00500
00502     bounded2_p_t hours_p (uint2_p.derived(), 0u, 23u)
      ;
00503
00505     bounded2_p_t minutes_p (uint2_p.derived(), 0u,
      59u);
00506
00508     bounded2_p_t seconds_p (uint2_p.derived(), 0u,
      59u);
00509
00511     chset_t cabin_code_p ("A-Z");
00512
00514     int1_p_t family_code_p;
00515
00517     repeat_p_t key_p (chset_t("0-9A-Z").derived(), 1, 10)
      ;
00518
00520     repeat_p_t class_code_list_p (chset_t("
      A-Z").derived(), 1, 26);
00521
00522
00523     // ////////////////////////////////////////////////////////////////
00524     //  (Boost Spirit) Grammar Definition
00525     // ////////////////////////////////////////////////////////////////
00526
00527     // ////////////////////////////////////////////////////////////////
00528     FlightPeriodParser::
00529     FlightPeriodParser (stdair::BomRoot& ioBomRoot,
00530                         FlightPeriodStruct& ioFlightPeriod)
00531       : _bomRoot (ioBomRoot),
00532         _flightPeriod (ioFlightPeriod) {
00533     }
00534
00535     // ////////////////////////////////////////////////////////////////
00536     template<typename ScannerT>
00537     FlightPeriodParser::definition<ScannerT>::
00538     definition (FlightPeriodParser const& self)
       {
00539
00540       flight_period_list = *( not_to_be_parsed | flight_period )
00541         ;
00542
00543       not_to_be_parsed =
00544         bsc::lexeme_d[ bsc::comment_p("//") | bsc::comment_p("/*", "*/")
00545                        | bsc::space_p ]
00546         ;
00547
00548       flight_period = flight_key
00549         >> +( ';' >> leg )
00550         >> ';' >> segment_section
00551         >> flight_period_end[doEndFlight (self._bomRoot, self.
      _flightPeriod)]
00552         ;
00553
00554       flight_period_end = bsc::ch_p(';')
00555         ;
00556
00557       flight_key = airline_code
00558         >> ';' >> flight_number
00559         >> ';' >> date[storeDateRangeStart(self.
      _flightPeriod)]
00560         >> ';' >> date[storeDateRangeEnd(self._flightPeriod)]
00561         >> ';' >> dow[storeDow(self._flightPeriod)]
00562         ;
00563
00564       airline_code =
00565         bsc::lexeme_d[(airline_code_p)[storeAirlineCode
      (self._flightPeriod)]]
00566         ;
00567
00568       flight_number =
00569         bsc::lexeme_d[(flight_number_p)[storeFlightNumber
      (self._flightPeriod)]]
00570         ;
00571
00572       date =
00573         bsc::lexeme_d[(year_p)[bsc::assign_a(self._flightPeriod._itYear)]
00574           >> '-' >> (month_p)[bsc::assign_a(self._flightPeriod._itMonth)
      ]
00575           >> '-' >> (day_p)[bsc::assign_a(self._flightPeriod._itDay)]]
00576         ;
00577
00578       dow = bsc::lexeme_d[ dow_p ]
```

```
00579         ;
00580
00581       leg = !( operating_leg_details >> ';' )
00582         >> leg_key
00583         >> ';' >> leg_details
00584         >> +( ';' >> leg_cabin_details )
00585         ;
00586
00587       leg_key = (airport_p)[storeLegBoardingPoint
      (self._flightPeriod)]
00588          >> ';'
00589          >> (airport_p)[storeLegOffPoint(self.
      _flightPeriod)]
00590          ;
00591
00592       operating_leg_details =
00593         bsc::lexeme_d[(airline_code_p)[storeOperatingAirlineCode
      (self._flightPeriod)] ]
00594         >> ";"
00595         >> bsc::lexeme_d[(flight_number_p)[
      storeOperatingFlightNumber(self._flightPeriod)] ]
00596         ;
00597
00598       leg_details =
00599         time[storeBoardingTime(self._flightPeriod)]
00600         >> !(date_offset)
00601         >> ';'
00602         >> time[storeOffTime(self._flightPeriod)]
00603         >> !(date_offset)
00604         >> ';'
00605         >> time[storeElapsedTime(self._flightPeriod)]
00606         ;
00607
00608       time =
00609         bsc::lexeme_d[(hours_p)[bsc::assign_a(self._flightPeriod.
      _itHours)]
00610           >> ':' >> (minutes_p)[bsc::assign_a(self._flightPeriod.
      _itMinutes)]
00611           >> !(':' >> (seconds_p)[bsc::assign_a(self._flightPeriod.
      _itSeconds)])]
00612         ;
00613
00614       date_offset =
00615         bsc::ch_p('/')
00616         >> (int1_p)[bsc::assign_a(self._flightPeriod._dateOffset)]
00617         ;
00618
00619       leg_cabin_details =
00620         (cabin_code_p)[storeLegCabinCode(self.
      _flightPeriod)]
00621         >> ';' >> (bsc::ureal_p)[storeCapacity(self._flightPeriod)
      ]
00622         ;
00623
00624       segment_key =
00625         (airport_p)[storeSegmentBoardingPoint
      (self._flightPeriod)]
00626         >> ';'
00627         >> (airport_p)[storeSegmentOffPoint(self.
      _flightPeriod)]
00628         ;
00629
00630       segment_section =
00631         generic_segment | specific_segment_list
00632         ;
00633
00634       generic_segment =
00635         bsc::ch_p('0')[storeSegmentSpecificity(self.
      _flightPeriod)]
00636         >> +(';' >> segment_cabin_details)
00637         ;
00638
00639       specific_segment_list =
00640         bsc::ch_p('1')[storeSegmentSpecificity(self.
      _flightPeriod)]
00641         >> +(';' >> segment_key >> full_segment_cabin_details)
00642         ;
00643
00644       full_segment_cabin_details =
00645         +(';' >> segment_cabin_details)
00646         ;
00647
00648       segment_cabin_details =
00649         (cabin_code_p)[storeSegmentCabinCode(
      self._flightPeriod)]
00650         >> ';' >> (class_code_list_p)[storeClasses
      (self._flightPeriod)]
```

```
00651         >> +(';' >> family_cabin_details)
00652         ;
00653
00654       family_cabin_details =
00655         (family_code_p)[storeFamilyCode(self.
      _flightPeriod)]
00656         >> ';'
00657         >> (key_p)[storeFRAT5CurveKey(self._flightPeriod
      )]
00658         >> ';'
00659         >> (key_p)[storeFFDisutilityCurveKey(self
      ._flightPeriod)]
00660         >> ';'
00661         >> (class_code_list_p)[storeFClasses(self
      ._flightPeriod)]
00662         ;
00663
00664       // BOOST_SPIRIT_DEBUG_NODE (FlightPeriodParser);
00665       BOOST_SPIRIT_DEBUG_NODE (flight_period_list);
00666       BOOST_SPIRIT_DEBUG_NODE (not_to_be_parsed);
00667       BOOST_SPIRIT_DEBUG_NODE (flight_period);
00668       BOOST_SPIRIT_DEBUG_NODE (flight_period_end);
00669       BOOST_SPIRIT_DEBUG_NODE (flight_key);
00670       BOOST_SPIRIT_DEBUG_NODE (airline_code);
00671       BOOST_SPIRIT_DEBUG_NODE (flight_number);
00672       BOOST_SPIRIT_DEBUG_NODE (date);
00673       BOOST_SPIRIT_DEBUG_NODE (dow);
00674       BOOST_SPIRIT_DEBUG_NODE (leg);
00675       BOOST_SPIRIT_DEBUG_NODE (leg_key);
00676       BOOST_SPIRIT_DEBUG_NODE (leg_details);
00677       BOOST_SPIRIT_DEBUG_NODE (time);
00678       BOOST_SPIRIT_DEBUG_NODE (date_offset);
00679       BOOST_SPIRIT_DEBUG_NODE (leg_cabin_details);
00680       BOOST_SPIRIT_DEBUG_NODE (segment_section);
00681       BOOST_SPIRIT_DEBUG_NODE (segment_key);
00682       BOOST_SPIRIT_DEBUG_NODE (generic_segment);
00683       BOOST_SPIRIT_DEBUG_NODE (specific_segment_list);
00684       BOOST_SPIRIT_DEBUG_NODE (full_segment_cabin_details);
00685       BOOST_SPIRIT_DEBUG_NODE (segment_cabin_details);
00686       BOOST_SPIRIT_DEBUG_NODE (family_cabin_details);
00687     }
00688
00689     // ////////////////////////////////////////////////////////////////////
00690     template<typename ScannerT>
00691     bsc::rule<ScannerT> const&
00692     FlightPeriodParser::definition<ScannerT>::start
      () const {
00693       return flight_period_list;
00694     }
00695   }
00696
00697
00699   //
00700   //  Entry class for the file parser
00701   //
00703
00704   // ////////////////////////////////////////////////////////////////////////
00705   FlightPeriodFileParser::
00706   FlightPeriodFileParser (stdair::BomRoot& ioBomRoot,
00707                           const stdair::Filename_T& iFilename)
00708     : _filename (iFilename), _bomRoot (ioBomRoot) {
00709     init();
00710   }
00711
00712   // ////////////////////////////////////////////////////////////////////////
00713   void FlightPeriodFileParser::init() {
00714     // Open the file
00715     _startIterator = iterator_t (_filename);
00716
00717     // Check the filename exists and can be open
00718     if (!_startIterator) {
00719       std::ostringstream oMessage;
00720       oMessage << "The file " << _filename << " can not be open." << std::endl;
00721       STDAIR_LOG_ERROR (oMessage.str());
00722       throw ScheduleInputFileNotFoundException
       (oMessage.str());
00723     }
00724
00725     // Create an EOF iterator
00726     _endIterator = _startIterator.make_end();
00727   }
00728
00729   // ////////////////////////////////////////////////////////////////////////
00730   bool FlightPeriodFileParser::generateInventories
      () {
00731     bool oResult = false;
00732
```

```
00733     STDAIR_LOG_DEBUG ("Parsing schedule input file: " << _filename);
00734
00735     // Initialise the parser (grammar) with the helper/staging structure.
00736     ScheduleParserHelper::FlightPeriodParser
      lFPParser (_bomRoot, _flightPeriod);
00737
00738     // Launch the parsing of the file and, thanks to the doEndFlight
00739     // call-back structure, the building of the whole BomRoot BOM
00740     // (i.e., including Inventory, FlightDate, LegDate, SegmentDate, etc.)
00741     bsc::parse_info<iterator_t> info = bsc::parse (_startIterator, _endIterator
     ,
00742                                                    lFPParser,
00743                                                    bsc::space_p - bsc::eol_p);
00744
00745     // Retrieves whether or not the parsing was successful
00746     oResult = info.hit;
00747
00748     const bool isFull = info.full;
00749
00750     const std::string hasBeenFullyReadStr = (isFull  == true)?"":"not ";
00751     if (oResult == true && isFull == true) {
00752       STDAIR_LOG_DEBUG ("Parsing of schedule input file: " << _filename
00753                         << " succeeded: read " << info.length
00754                         << " characters. The input file has "
00755                         << hasBeenFullyReadStr
00756                         << "been fully read. Stop point: " << info.stop);
00757
00758     } else {
00759       STDAIR_LOG_ERROR ("Parsing of schedule input file: " << _filename
00760                         << " failed: read " << info.length
00761                         << " characters. The input file has "
00762                         << hasBeenFullyReadStr
00763                         << "been fully read. Stop point: " << info.stop);
00764       throw ScheduleFileParsingFailedException
      ("Parsing of schedule input file: "
00765                                                + _filename + " failed.");
00766     }
00767
00768     return oResult;
00769   }
00770
00771 }
```

23.159 airinv/command/ScheduleParserHelper.hpp File Reference

```
#include <string>
#include <stdair/command/CmdAbstract.hpp>
#include <airinv/AIRINV_Types.hpp>
#include <airinv/basic/BasParserTypes.hpp>
#include <airinv/bom/FlightPeriodStruct.hpp>
```

Classes

- struct AIRINV::ScheduleParserHelper::ParserSemanticAction
- struct AIRINV::ScheduleParserHelper::storeAirlineCode
- struct AIRINV::ScheduleParserHelper::storeFlightNumber
- struct AIRINV::ScheduleParserHelper::storeDateRangeStart
- struct AIRINV::ScheduleParserHelper::storeDateRangeEnd
- struct AIRINV::ScheduleParserHelper::storeDow
- struct AIRINV::ScheduleParserHelper::storeLegBoardingPoint
- struct AIRINV::ScheduleParserHelper::storeLegOffPoint
- struct AIRINV::ScheduleParserHelper::storeOperatingAirlineCode
- struct AIRINV::ScheduleParserHelper::storeOperatingFlightNumber
- struct AIRINV::ScheduleParserHelper::storeBoardingTime
- struct AIRINV::ScheduleParserHelper::storeOffTime
- struct AIRINV::ScheduleParserHelper::storeElapsedTime
- struct AIRINV::ScheduleParserHelper::storeLegCabinCode
- struct AIRINV::ScheduleParserHelper::storeCapacity

- struct AIRINV::ScheduleParserHelper::storeSegmentSpecificity
- struct AIRINV::ScheduleParserHelper::storeSegmentBoardingPoint
- struct AIRINV::ScheduleParserHelper::storeSegmentOffPoint
- struct AIRINV::ScheduleParserHelper::storeSegmentCabinCode
- struct AIRINV::ScheduleParserHelper::storeClasses
- struct AIRINV::ScheduleParserHelper::storeFamilyCode
- struct AIRINV::ScheduleParserHelper::storeFRAT5CurveKey
- struct AIRINV::ScheduleParserHelper::storeFFDisutilityCurveKey
- struct AIRINV::ScheduleParserHelper::storeFClasses
- struct AIRINV::ScheduleParserHelper::doEndFlight
- struct AIRINV::ScheduleParserHelper::FlightPeriodParser
- struct AIRINV::ScheduleParserHelper::FlightPeriodParser::definition< ScannerT >
- class AIRINV::FlightPeriodFileParser

Namespaces

- namespace stdair

  Forward declarations.
- namespace AIRINV
- namespace AIRINV::ScheduleParserHelper

23.160 ScheduleParserHelper.hpp

```
00001 #ifndef __AIRINV_CMD_SCHEDULEPARSERHELPER_HPP
00002 #define __AIRINV_CMD_SCHEDULEPARSERHELPER_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/command/CmdAbstract.hpp>
00011 // Airinv
00012 #include <airinv/AIRINV_Types.hpp>
00013 #include <airinv/basic/BasParserTypes.hpp>
00014 #include <airinv/bom/FlightPeriodStruct.hpp>
00015
00016 // Forward declarations
00017 namespace stdair {
00018   class BomRoot;
00019 }
00020
00021 namespace AIRINV {
00022
00023   namespace ScheduleParserHelper {
00024
00025     // //////////////////////////////////////////////////////////////////
00026     //  Semantic actions
00027     // //////////////////////////////////////////////////////////////////
00029     struct ParserSemanticAction {
00031       ParserSemanticAction (FlightPeriodStruct
      &);
00033       FlightPeriodStruct& _flightPeriod;
00034     };
00035
00037     struct storeAirlineCode : public ParserSemanticAction
      {
00039       storeAirlineCode (FlightPeriodStruct&);
00041       void operator() (iterator_t iStr, iterator_t
      iStrEnd) const;
00042     };
00043
00045     struct storeFlightNumber : public ParserSemanticAction
      {
00047       storeFlightNumber (FlightPeriodStruct&
      );
00049       void operator() (unsigned int iNumber) const;
00050     };
00051
00053     struct storeDateRangeStart : public ParserSemanticAction
      {
```

```
00055       storeDateRangeStart (FlightPeriodStruct
      &);
00057       void operator() (iterator_t iStr, iterator_t
       iStrEnd) const;
00058     };
00059
00061     struct storeDateRangeEnd : public ParserSemanticAction
       {
00063       storeDateRangeEnd (FlightPeriodStruct&
      );
00065       void operator() (iterator_t iStr, iterator_t
       iStrEnd) const;
00066     };
00067
00069     struct storeDow : public ParserSemanticAction {
00071       storeDow (FlightPeriodStruct&);
00073       void operator() (iterator_t iStr, iterator_t
       iStrEnd) const;
00074     };
00075
00077     struct storeLegBoardingPoint : public
      ParserSemanticAction {
00079       storeLegBoardingPoint (FlightPeriodStruct
      &);
00081       void operator() (iterator_t iStr, iterator_t
       iStrEnd) const;
00082     };
00083
00085     struct storeLegOffPoint : public ParserSemanticAction
       {
00087       storeLegOffPoint (FlightPeriodStruct&);
00089       void operator() (iterator_t iStr, iterator_t
       iStrEnd) const;
00090     };
00091
00093     struct storeOperatingAirlineCode : public
      ParserSemanticAction {
00095       storeOperatingAirlineCode (FlightPeriodStruct
      &);
00097       void operator() (iterator_t iStr, iterator_t
       iStrEnd) const;
00098     };
00099
00101     struct storeOperatingFlightNumber : public
      ParserSemanticAction {
00103       storeOperatingFlightNumber (FlightPeriodStruct
      &);
00105       void operator() (unsigned int iNumber) const;
00106     };
00107
00109     struct storeBoardingTime : public ParserSemanticAction
       {
00111       storeBoardingTime (FlightPeriodStruct&
      );
00113       void operator() (iterator_t iStr, iterator_t
       iStrEnd) const;
00114     };
00115
00117     struct storeOffTime : public ParserSemanticAction
       {
00119       storeOffTime (FlightPeriodStruct&);
00121       void operator() (iterator_t iStr, iterator_t
       iStrEnd) const;
00122     };
00123
00125     struct storeElapsedTime : public ParserSemanticAction
       {
00127       storeElapsedTime (FlightPeriodStruct&);
00129       void operator() (iterator_t iStr, iterator_t
       iStrEnd) const;
00130     };
00131
00133     struct storeLegCabinCode : public ParserSemanticAction
       {
00135       storeLegCabinCode (FlightPeriodStruct&
      );
00137       void operator() (char iChar) const;
00138     };
00139
00141     struct storeCapacity : public ParserSemanticAction
       {
00143       storeCapacity (FlightPeriodStruct&);
00145       void operator() (double iReal) const;
00146     };
00147
00152     struct storeSegmentSpecificity : public
      ParserSemanticAction {
```

```
00154       storeSegmentSpecificity (FlightPeriodStruct
      &);
00156       void operator() (char iChar) const;
00157     };
00158
00160     struct storeSegmentBoardingPoint : public
      ParserSemanticAction {
00162       storeSegmentBoardingPoint (FlightPeriodStruct
      &);
00164       void operator() (iterator_t iStr, iterator_t
       iStrEnd) const;
00165     };
00166
00168     struct storeSegmentOffPoint : public
      ParserSemanticAction {
00170       storeSegmentOffPoint (FlightPeriodStruct
      &);
00172       void operator() (iterator_t iStr, iterator_t
       iStrEnd) const;
00173     };
00174
00176     struct storeSegmentCabinCode : public
      ParserSemanticAction {
00178       storeSegmentCabinCode (FlightPeriodStruct
      &);
00180       void operator() (char iChar) const;
00181     };
00182
00184     struct storeClasses : public ParserSemanticAction
      {
00186       storeClasses (FlightPeriodStruct&);
00188       void operator() (iterator_t iStr, iterator_t
       iStrEnd) const;
00189     };
00190
00192     struct storeFamilyCode : public ParserSemanticAction
      {
00194       storeFamilyCode (FlightPeriodStruct&);
00196       void operator() (int iCode) const;
00197     };
00198
00200     struct storeFRAT5CurveKey : public ParserSemanticAction
      {
00202       storeFRAT5CurveKey (FlightPeriodStruct
      &);
00204       void operator() (iterator_t iStr, iterator_t
       iStrEnd) const;
00205     };
00206
00208     struct storeFFDisutilityCurveKey : public
      ParserSemanticAction {
00210       storeFFDisutilityCurveKey (FlightPeriodStruct
      &);
00212       void operator() (iterator_t iStr, iterator_t
       iStrEnd) const;
00213     };
00214
00216     struct storeFClasses : public ParserSemanticAction
      {
00218       storeFClasses (FlightPeriodStruct&);
00220       void operator() (iterator_t iStr, iterator_t
       iStrEnd) const;
00221     };
00222
00224     struct doEndFlight : public ParserSemanticAction
      {
00226       doEndFlight (stdair::BomRoot&, FlightPeriodStruct
      &);
00228       void operator() (iterator_t iStr, iterator_t
       iStrEnd) const;
00230       stdair::BomRoot& _bomRoot;
00231     };
00232
00233
00235     //
00236     //  (Boost Spirit) Grammar Definition
00237     //
00239
00282     struct FlightPeriodParser :
00283       public boost::spirit::classic::grammar<FlightPeriodParser> {
00284
00285       FlightPeriodParser (stdair::BomRoot&,
      FlightPeriodStruct&);
00286
00287       template <typename ScannerT>
00288       struct definition {
00289         definition (FlightPeriodParser const& self)
```

```
    ;
00290
00291         // Instantiation of rules
00292         boost::spirit::classic::rule<ScannerT> flight_period_list
    ,
00293           not_to_be_parsed, flight_period,
    flight_period_end,
00294           flight_key, airline_code, flight_number
    ,
00295           date, dow, time, date_offset,
00296           leg, leg_key, operating_leg_details,
    leg_details, leg_cabin_details,
00297           segment_section, segment_key,
    full_segment_cabin_details,
00298           segment_cabin_details, full_family_cabin_details
    ,
00299           family_cabin_details, generic_segment
    , specific_segment_list;
00300
00302         boost::spirit::classic::rule<ScannerT> const& start() const;
00303       };
00304
00305       // Parser Context
00306       stdair::BomRoot& _bomRoot;
00307       FlightPeriodStruct& _flightPeriod;
00308     };
00309
00310   }
00315
00316   //
00317   //  Entry class for the file parser
00318   //
00320
00325   class FlightPeriodFileParser : public
    stdair::CmdAbstract {
00326   public:
00328     FlightPeriodFileParser (stdair::BomRoot& ioBomRoot,
00329                             const stdair::Filename_T& iFilename);
00330
00332     bool generateInventories ();
00333
00334   private:
00336     void init();
00337
00338   private:
00339     // Attributes
00341     stdair::Filename_T _filename;
00342
00344     iterator_t _startIterator;
00345
00347     iterator_t _endIterator;
00348
00350     stdair::BomRoot& _bomRoot;
00351
00353     FlightPeriodStruct _flightPeriod;
00354   };
00355
00356 }
00357 #endif // __AIRINV_CMD_SCHEDULEPARSERHELPER_HPP
```

23.161 airinv/command/vault/DCPEventGenerator.cpp File Reference

```
#include <cassert>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/factory/FacBomManager.hpp>
#include <stdair/service/Logger.hpp>
#include <airinv/bom/DCPEventStruct.hpp>
#include <airinv/command/DCPEventGenerator.hpp>
```

Namespaces

- namespace AIRINV

23.162 DCPEventGenerator.cpp

```
00001 // //////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // StdAir
00007 #include <stdair/bom/BomManager.hpp>
00008 #include <stdair/bom/BomRoot.hpp>
00009 #include <stdair/factory/FacBomManager.hpp>
00010 #include <stdair/service/Logger.hpp>
00011 // AirInv
00012 #include <airinv/bom/DCPEventStruct.hpp>
00013 #include <airinv/command/DCPEventGenerator.hpp>
00014
00015 namespace AIRINV {
00016
00017   // //////////////////////////////////////////////////////////////////////
00018   void DCPEventGenerator::
00019   createDCPEvent (stdair::BomRoot& ioBomRoot,
00020                   DCPEventStruct& iDCPEventStruct) {
00021
00022     // Set the airport-pair primary key.
00023     /*
00024     const stdair::AirportCode_T& lBoardPoint = iDCPEventStruct._origin;
00025     const stdair::AirportCode_T& lOffPoint = iDCPEventStruct._destination;
00026     */
00027
00028     // Set the DCP date-period primary key.
00029     const stdair::Date_T& lDateRangeStart = iDCPEventStruct._dateRangeStart;
00030     const stdair::Date_T& lDateRangeEnd = iDCPEventStruct._dateRangeEnd;
00031     const stdair::DatePeriod_T lDatePeriod (lDateRangeStart, lDateRangeEnd);
00032
00033     // Set the DCP time-period primary key.
00034     /*
00035     const stdair::Time_T& lTimeRangeStart = iDCPEventStruct._timeRangeStart;
00036     const stdair::Time_T& lTimeRangeEnd = iDCPEventStruct._timeRangeEnd;
00037     */
00038
00039     // Generate the DCPEvent
00040     const stdair::DayDuration_T& lAdvancePurchase =
00041       iDCPEventStruct._advancePurchase;
00042     const stdair::SaturdayStay_T& lSaturdayStay = iDCPEventStruct._saturdayStay
     ;
00043     const stdair::ChangeFees_T& lChangeFees = iDCPEventStruct._changeFees;
00044     const stdair::NonRefundable_T& lNonRefundable =
00045       iDCPEventStruct._nonRefundable;
00046     const stdair::DayDuration_T& lMinimumStay = iDCPEventStruct._minimumStay;
00047     const stdair::Fare_T& lDCP = iDCPEventStruct._DCP;
00048
00049     // Generate Segment Features and link them to their DCPEvent
00050     stdair::ClassList_StringList_T::const_iterator lItCurrentClassCodeList =
00051       iDCPEventStruct._classCodeList.begin();
00052
00053     const unsigned int lAirlineListSize = iDCPEventStruct.getAirlineListSize();
00054     const unsigned int lClassCodeListSize =
00055       iDCPEventStruct.getClassCodeListSize();
00056     assert (lAirlineListSize == lClassCodeListSize);
00057
00058     iDCPEventStruct.beginClassCode();
00059     for (iDCPEventStruct.beginAirline();
00060          iDCPEventStruct.hasNotReachedEndAirline();
00061          iDCPEventStruct.iterateAirline()) {
00062       /*
00063       const stdair::AirlineCode_T& lAirlineCode =
00064         iDCPEventStruct.getCurrentAirlineCode();
00065       const std::string& lClassCodeList =
       iDCPEventStruct.getCurrentClassCode();
00066       iDCPEventStruct.iterateClassCode();
00067       */
00068     }
00069   }
00070
00071 }
00072
```

23.163 airinv/command/vault/DCPEventGenerator.hpp File Reference

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

Classes

- class AIRINV::DCPEventGenerator

Namespaces

- namespace stdair

  Forward declarations.
- namespace AIRINV
- namespace AIRINV::DCPParserHelper

23.164 DCPEventGenerator.hpp

```
00001 #ifndef __AIRINV_CMD_DCPEVENTGENERATOR_HPP
00002 #define __AIRINV_CMD_DCPEVENTGENERATOR_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // StdAir
00008 #include <stdair/command/CmdAbstract.hpp>
00009 // AirInv
00010 #include <airinv/AIRINV_Types.hpp>
00011
00012 // Forward declarations
00013 namespace stdair {
00014   class BomRoot;
00015   class DCPEvent;
00016 }
00017
00018 namespace AIRINV {
00019
00020   // Forward declarations
00021   struct DCPEventStruct;
00022   namespace DCPParserHelper {
00023     struct doEndDCP;
00024   }
00025
00027   class DCPEventGenerator : public stdair::CmdAbstract {
00028     // Only the following class may use methods of DCPGenerator.
00029     // Indeed, as those methods build the BOM, it is not good to expose
00030     // them public.
00031     friend class DCPFileParser;
00032     friend struct DCPParserHelper::doEndDCP;
00033     friend class DCPParser;
00034   private:
00037     static void createDCPEvent (stdair::BomRoot&, DCPEventStruct&
      );
00038   };
00039
00040 }
00041 #endif // __AIRINV_CMD_DCPEVENTGENERATOR_HPP
```

23.165 airinv/command/vault/DCPParser.cpp File Reference

```
#include <cassert>
#include <string>
#include <stdair/service/Logger.hpp>
#include <airinv/command/DCPParserHelper.hpp>
#include <airinv/command/DCPParser.hpp>
```

Namespaces

- namespace AIRINV

23.166 DCPParser.cpp

```
00001 // //////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <string>
00007 // StdAir
00008 #include <stdair/service/Logger.hpp>
00009 // AirSched
00010 #include <airinv/command/DCPParserHelper.hpp>
00011 #include <airinv/command/DCPParser.hpp>
00012
00013 namespace AIRINV {
00014
00015   // ////////////////////////////////////////////////////////////////////
00016   void DCPParser::DCPRuleGeneration (const
      stdair::Filename_T& iFilename,
00017                                      stdair::BomRoot& ioBomRoot) {
00018
00019     // Initialise the DCP file parser
00020     DCPRuleFileParser lDCPRuleFileParser (ioBomRoot, iFilename
      );
00021
00022     // Parse the CSV-formatted DCP input file and generate the
00023     // corresponding DCP events
00024     lDCPRuleFileParser.generateDCPRules();
00025   }
00026
00027 }
```

23.167 airinv/command/vault/DCPParser.hpp File Reference

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

Classes

- class AIRINV::DCPParser

Namespaces

- namespace stdair

  Forward declarations.
- namespace AIRINV

23.168 DCPParser.hpp

```
00001 #ifndef __AIRINV_CMD_DCPPARSER_HPP
00002 #define __AIRINV_CMD_DCPPARSER_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009 #include <stdair/command/CmdAbstract.hpp>
00010
00011 // Forward declarations.
00012 namespace stdair {
00013   class BomRoot;
00014 }
00015
00016 namespace AIRINV {
00017
00019   class DCPParser : public stdair::CmdAbstract {
00020   public:
00028     static void DCPRuleGeneration (const stdair::Filename_T&,
00029                                    stdair::BomRoot&);
00030   };
```

```
00031 }
00032 #endif // __AIRINV_CMD_DCPPARSER_HPP
```

23.169 airinv/command/vault/DCPParserHelper.cpp File Reference

```
#include <cassert>
#include <string>
#include <vector>
#include <fstream>
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/service/Logger.hpp>
#include <airinv/command/DCPParserHelper.hpp>
#include <airinv/command/DCPRuleGenerator.hpp>
```

Namespaces

- namespace AIRINV
- namespace AIRINV::DCPParserHelper

Variables

- stdair::int1_p_t AIRINV::DCPParserHelper::int1_p
- stdair::uint2_p_t AIRINV::DCPParserHelper::uint2_p
- stdair::uint4_p_t AIRINV::DCPParserHelper::uint4_p
- stdair::uint1_4_p_t AIRINV::DCPParserHelper::uint1_4_p
- stdair::hour_p_t AIRINV::DCPParserHelper::hour_p
- stdair::minute_p_t AIRINV::DCPParserHelper::minute_p
- stdair::second_p_t AIRINV::DCPParserHelper::second_p
- stdair::year_p_t AIRINV::DCPParserHelper::year_p
- stdair::month_p_t AIRINV::DCPParserHelper::month_p
- stdair::day_p_t AIRINV::DCPParserHelper::day_p

23.170 DCPParserHelper.cpp

```
00001  // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <string>
00007 #include <vector>
00008 #include <fstream>
00009 // StdAir
00010 #include <stdair/basic/BasFileMgr.hpp>
00011 #include <stdair/bom/BomRoot.hpp>
00012 #include <stdair/service/Logger.hpp>
00013 // AirInv
00014 #include <airinv/command/DCPParserHelper.hpp>
00015 #include <airinv/command/DCPRuleGenerator.hpp>
00016
00017 namespace AIRINV {
00018
00019   namespace DCPParserHelper {
00020
00021     // ////////////////////////////////////////////////////////////////
00022     //  Semantic actions
00023     // ////////////////////////////////////////////////////////////////
00024
00025     ParserSemanticAction::ParserSemanticAction
       (DCPRuleStruct& ioDCPRule)
00026       : _DCPRule (ioDCPRule) {
00027     }
```

```
00028
00029     // ////////////////////////////////////////////////////////////////////
00030     storeDCPId::storeDCPId (DCPRuleStruct& ioDCPRule)
00031       : ParserSemanticAction (ioDCPRule) {
00032     }
00033
00034     // ////////////////////////////////////////////////////////////////////
00035     void storeDCPId::operator() (unsigned int iDCPId,
00036                                  boost::spirit::qi::unused_type,
00037                                  boost::spirit::qi::unused_type) const {
00038       _DCPRule._DCPId = iDCPId;
00039
00040       // DEBUG
00041       //STDAIR_LOG_DEBUG ( "DCP Id: " << _DCPRule._DCPId);
00042
00043       _DCPRule._nbOfAirlines = 0;
00044       _DCPRule._airlineCode = "";
00045       _DCPRule._classCode = "";
00046       _DCPRule._airlineCodeList.clear();
00047       _DCPRule._classCodeList.clear();
00048       _DCPRule._classCodeListOfList.clear();
00049       _DCPRule._itSeconds = 0;
00050     }
00051
00052     // ////////////////////////////////////////////////////////////////////
00053     storeOrigin ::
00054     storeOrigin (DCPRuleStruct& ioDCPRule)
00055       : ParserSemanticAction (ioDCPRule) {
00056     }
00057
00058     // ////////////////////////////////////////////////////////////////////
00059     void storeOrigin::operator() (std::vector<char>
      iChar,
00060                                  boost::spirit::qi::unused_type,
00061                                  boost::spirit::qi::unused_type) const {
00062       stdair::AirportCode_T lOrigin (iChar.begin(), iChar.end());
00063       // DEBUG
00064       //STDAIR_LOG_DEBUG ( "Origin: " << lOrigin);
00065       _DCPRule._origin = lOrigin;
00066     }
00067
00068     // ////////////////////////////////////////////////////////////////////
00069     storeDestination ::
00070     storeDestination (DCPRuleStruct& ioDCPRule)
00071       : ParserSemanticAction (ioDCPRule) {
00072     }
00073
00074     // ////////////////////////////////////////////////////////////////////
00075     void storeDestination::operator() (
      std::vector<char> iChar,
00076                                        boost::spirit::qi::unused_type,
00077                                        boost::spirit::qi::unused_type) const {
00078       stdair::AirportCode_T lDestination (iChar.begin(), iChar.end());
00079       // DEBUG
00080       //STDAIR_LOG_DEBUG ( "Destination: " << lDestination);
00081       _DCPRule._destination = lDestination;
00082     }
00083
00084     // ////////////////////////////////////////////////////////////////////
00085     storeDateRangeStart::
00086     storeDateRangeStart (DCPRuleStruct& ioDCPRule)
00087       : ParserSemanticAction (ioDCPRule) {
00088     }
00089
00090     // ////////////////////////////////////////////////////////////////////
00091     void storeDateRangeStart::operator() (
      boost::spirit::qi::unused_type,
00092                                           boost::spirit::qi::unused_type,
00093                                           boost::spirit::qi::unused_type) const
       {
00094       _DCPRule._dateRangeStart = _DCPRule.getDate();
00095       // DEBUG
00096       //STDAIR_LOG_DEBUG ("Date Range Start: "<< _DCPRule._dateRangeStart);
00097     }
00098
00099     // ////////////////////////////////////////////////////////////////////
00100     storeDateRangeEnd::
00101     storeDateRangeEnd(DCPRuleStruct& ioDCPRule)
00102       : ParserSemanticAction (ioDCPRule) {
00103     }
00104
00105     // ////////////////////////////////////////////////////////////////////
00106     void storeDateRangeEnd::operator() (
      boost::spirit::qi::unused_type,
00107                                         boost::spirit::qi::unused_type,
00108                                         boost::spirit::qi::unused_type) const {
00109       _DCPRule._dateRangeEnd = _DCPRule.getDate();
```

```
00110       // DEBUG
00111       //STDAIR_LOG_DEBUG ("Date Range End: " << _DCPRule._dateRangeEnd);
00112     }
00113
00114     // ////////////////////////////////////////////////////////////////////
00115     storeStartRangeTime::
00116     storeStartRangeTime (DCPRuleStruct& ioDCPRule)
00117       : ParserSemanticAction (ioDCPRule) {
00118     }
00119
00120     // ////////////////////////////////////////////////////////////////////
00121     void storeStartRangeTime::operator() (
      boost::spirit::qi::unused_type,
00122                                           boost::spirit::qi::unused_type,
00123                                           boost::spirit::qi::unused_type) const
       {
00124       _DCPRule._timeRangeStart = _DCPRule.getTime();
00125       // DEBUG
00126       //STDAIR_LOG_DEBUG ("Time Range Start: " << _DCPRule._timeRangeStart);
00127       // Reset the number of seconds
00128       _DCPRule._itSeconds = 0;
00129     }
00130
00131     // ////////////////////////////////////////////////////////////////////
00132     storeEndRangeTime::
00133     storeEndRangeTime (DCPRuleStruct& ioDCPRule)
00134       : ParserSemanticAction (ioDCPRule) {
00135     }
00136
00137     // ////////////////////////////////////////////////////////////////////
00138     void storeEndRangeTime::operator() (
      boost::spirit::qi::unused_type,
00139                                         boost::spirit::qi::unused_type,
00140                                         boost::spirit::qi::unused_type) const {
00141       _DCPRule._timeRangeEnd = _DCPRule.getTime();
00142       // DEBUG
00143       //STDAIR_LOG_DEBUG ("Time Range End: " << _DCPRule._timeRangeEnd);
00144       // Reset the number of seconds
00145       _DCPRule._itSeconds = 0;
00146     }
00147
00148     // ////////////////////////////////////////////////////////////////////
00149     storePOS ::
00150     storePOS (DCPRuleStruct& ioDCPRule)
00151       : ParserSemanticAction (ioDCPRule) {
00152     }
00153
00154     // ////////////////////////////////////////////////////////////////////
00155     void storePOS::operator() (std::vector<char> iChar,
00156                                boost::spirit::qi::unused_type,
00157                                boost::spirit::qi::unused_type) const {
00158       stdair::AirlineCode_T lPOS (iChar.begin(), iChar.end());
00159       _DCPRule._pos = lPOS;
00160       // DEBUG
00161       //STDAIR_LOG_DEBUG ("POS: " << _DCPRule._pos);
00162     }
00163
00164     // ////////////////////////////////////////////////////////////////////
00165     storeCabinCode ::
00166     storeCabinCode (DCPRuleStruct& ioDCPRule)
00167       : ParserSemanticAction (ioDCPRule) {
00168     }
00169
00170     // ////////////////////////////////////////////////////////////////////
00171     void storeCabinCode::operator() (char iChar,
00172                                      boost::spirit::qi::unused_type,
00173                                      boost::spirit::qi::unused_type) const {
00174       std::ostringstream ostr;
00175       ostr << iChar;
00176       std::string cabinCodeStr = ostr.str();
00177       const stdair::CabinCode_T lCabinCode (cabinCodeStr);
00178       _DCPRule._cabinCode = lCabinCode;
00179
00180       // DEBUG
00181       //STDAIR_LOG_DEBUG ("Cabin Code: " << lCabinCode);
00182
00183     }
00184
00185     // ////////////////////////////////////////////////////////////////////
00186     storeChannel ::
00187     storeChannel (DCPRuleStruct& ioDCPRule)
00188       : ParserSemanticAction (ioDCPRule) {
00189     }
00190
00191     // ////////////////////////////////////////////////////////////////////
00192     void storeChannel::operator() (std::vector<char>
      iChar,
```

```
00193                                   boost::spirit::qi::unused_type,
00194                                   boost::spirit::qi::unused_type) const {
00195       stdair::ChannelLabel_T lChannel (iChar.begin(), iChar.end());
00196       if (lChannel != "IN" && lChannel != "IF"
00197           && lChannel != "DN" && lChannel != "DF") {
00198         // DEBUG
00199         STDAIR_LOG_DEBUG ("Invalid channel " << lChannel);
00200       }
00201       _DCPRule._channel = lChannel;
00202       // DEBUG
00203       //STDAIR_LOG_DEBUG ("Channel: " << _DCPRule._channel);
00204     }
00205
00206     // ////////////////////////////////////////////////////////////////////
00207     storeAdvancePurchase ::
00208     storeAdvancePurchase (DCPRuleStruct& ioDCPRule)
00209       : ParserSemanticAction (ioDCPRule) {
00210     }
00211
00212     // ////////////////////////////////////////////////////////////////////
00213     void storeAdvancePurchase::operator() (
    unsigned int iAdancePurchase,
00214                                            boost::spirit::qi::unused_type,
00215                                            boost::spirit::qi::unused_type)
     const {
00216       _DCPRule._advancePurchase = iAdancePurchase;
00217       // DEBUG
00218       //STDAIR_LOG_DEBUG ( "Advance Purchase: " << _DCPRule._advancePurchase);
00219     }
00220
00221     // ////////////////////////////////////////////////////////////////////
00222     storeSaturdayStay ::
00223     storeSaturdayStay (DCPRuleStruct& ioDCPRule)
00224       : ParserSemanticAction (ioDCPRule) {
00225     }
00226
00227     // ////////////////////////////////////////////////////////////////////
00228     void storeSaturdayStay::operator() (char
    iSaturdayStay,
00229                                         boost::spirit::qi::unused_type,
00230                                         boost::spirit::qi::unused_type) const {
00231       bool lBool = false;
00232       if (iSaturdayStay == 'T') {
00233         lBool = true;
00234       } else {
00235         if (iSaturdayStay != 'F') {
00236           // DEBUG
00237           STDAIR_LOG_DEBUG ("Invalid saturdayStay char " << iSaturdayStay);
00238         }
00239       }
00240       stdair::SaturdayStay_T lSaturdayStay (lBool);
00241       _DCPRule._saturdayStay = lSaturdayStay;
00242       // DEBUG
00243       //STDAIR_LOG_DEBUG ("Saturday Stay: " << _DCPRule._saturdayStay);
00244     }
00245
00246     // ////////////////////////////////////////////////////////////////////
00247     storeChangeFees ::
00248     storeChangeFees (DCPRuleStruct& ioDCPRule)
00249       : ParserSemanticAction (ioDCPRule) {
00250     }
00251
00252     // ////////////////////////////////////////////////////////////////////
00253     void storeChangeFees::operator() (char
    iChangefees,
00254                                       boost::spirit::qi::unused_type,
00255                                       boost::spirit::qi::unused_type) const {
00256
00257       bool lBool = false;
00258       if (iChangefees == 'T') {
00259         lBool = true;
00260       } else {
00261         if (iChangefees != 'F') {
00262           // DEBUG
00263           STDAIR_LOG_DEBUG ("Invalid change fees char " << iChangefees);
00264         }
00265       }
00266       stdair::ChangeFees_T lChangefees (lBool);
00267       _DCPRule._changeFees = lChangefees;
00268       // DEBUG
00269       //STDAIR_LOG_DEBUG ("Change fees: " << _DCPRule._changeFees);
00270     }
00271
00272     // ////////////////////////////////////////////////////////////////////
00273     storeNonRefundable ::
00274     storeNonRefundable (DCPRuleStruct& ioDCPRule)
00275       : ParserSemanticAction (ioDCPRule) {
```

```
00276     }
00277
00278     // ////////////////////////////////////////////////////////////////////
00279     void storeNonRefundable::operator() (char
    iNonRefundable,
00280                                          boost::spirit::qi::unused_type,
00281                                          boost::spirit::qi::unused_type) const
    {
00282       bool lBool = false;
00283       if (iNonRefundable == 'T') {
00284         lBool = true;
00285       } else {
00286         if (iNonRefundable != 'F') {
00287           // DEBUG
00288           STDAIR_LOG_DEBUG ("Invalid non refundable char " << iNonRefundable);
00289         }
00290       }
00291       stdair::NonRefundable_T lNonRefundable (lBool);
00292       _DCPRule._nonRefundable = lNonRefundable;
00293       // DEBUG
00294       //STDAIR_LOG_DEBUG ("Non refundable: " << _DCPRule._nonRefundable);
00295     }
00296
00297     // ////////////////////////////////////////////////////////////////////
00298     storeMinimumStay ::
00299     storeMinimumStay (DCPRuleStruct& ioDCPRule)
00300       : ParserSemanticAction (ioDCPRule) {
00301     }
00302
00303     // ////////////////////////////////////////////////////////////////////
00304     void storeMinimumStay::operator() (unsigned
    int iMinStay,
00305                                        boost::spirit::qi::unused_type,
00306                                        boost::spirit::qi::unused_type) const {
00307       _DCPRule._minimumStay = iMinStay;
00308       // DEBUG
00309       //STDAIR_LOG_DEBUG ("Minimum Stay: " << _DCPRule._minimumStay );
00310     }
00311
00312     // ////////////////////////////////////////////////////////////////////
00313     storeDCP ::
00314     storeDCP (DCPRuleStruct& ioDCPRule)
00315       : ParserSemanticAction (ioDCPRule) {
00316     }
00317
00318     // ////////////////////////////////////////////////////////////////////
00319     void storeDCP::operator() (double iDCP,
00320                                boost::spirit::qi::unused_type,
00321                                boost::spirit::qi::unused_type) const {
00322       _DCPRule._DCP = iDCP;
00323       // DEBUG
00324       //STDAIR_LOG_DEBUG ("DCP: " << _DCPRule._DCP);
00325     }
00326
00327     // ////////////////////////////////////////////////////////////////////
00328     storeAirlineCode ::
00329     storeAirlineCode (DCPRuleStruct& ioDCPRule)
00330       : ParserSemanticAction (ioDCPRule) {
00331     }
00332
00333     // ////////////////////////////////////////////////////////////////////
00334     void storeAirlineCode::operator() (
    std::vector<char> iChar,
00335                                        boost::spirit::qi::unused_type,
00336                                        boost::spirit::qi::unused_type) const {
00337
00338       bool lAlreadyInTheList = false;
00339       stdair::AirlineCode_T lAirlineCode (iChar.begin(), iChar.end());
00340       // Update the airline code
00341       _DCPRule._airlineCode = lAirlineCode;
00342       // Test if the DCPRule Struct stands for interline products
00343       if (_DCPRule._airlineCodeList.size() > 0) {
00344         _DCPRule._classCodeListOfList.push_back(_DCPRule.
    _classCodeList);
00345         _DCPRule._classCodeList.clear();
00346         // Update the number of airlines if necessary
00347         std::vector<stdair::AirlineCode_T>::iterator Airline_iterator;
00348         for (Airline_iterator = _DCPRule._airlineCodeList.begin();
00349              Airline_iterator != _DCPRule._airlineCodeList.end();
00350              ++Airline_iterator) {
00351           stdair::AirlineCode_T lPreviousAirlineCode =
00352             *Airline_iterator;
00353           if (lPreviousAirlineCode == lAirlineCode) {
00354             lAlreadyInTheList = true;
00355             /*STDAIR_LOG_DEBUG ("Airline Code Already Existing: "
00356               << lAirlineCode);*/
00357           }
```

```
00358         }
00359         if (lAlreadyInTheList == false) {
00360           /*STDAIR_LOG_DEBUG ("New Airline Code: "
00361           << lAirlineCode);*/
00362           _DCPRule._airlineCodeList.push_back(lAirlineCode);
00363           _DCPRule._classCodeList.clear();
00364         }
00365       } else {
00366         /*STDAIR_LOG_DEBUG ("First Airline Code: "
00367           << lAirlineCode);*/
00368         _DCPRule._airlineCodeList.push_back (lAirlineCode);
00369       }
00370       // DEBUG
00371       //STDAIR_LOG_DEBUG ( "Airline code: " << lAirlineCode);
00372     }
00373
00374     // ////////////////////////////////////////////////////////////////////
00375     storeClass ::
00376     storeClass (DCPRuleStruct& ioDCPRule)
00377       : ParserSemanticAction (ioDCPRule) {
00378     }
00379
00380     // ////////////////////////////////////////////////////////////////////
00381     void storeClass::operator() (std::vector<char> iChar
      ,
00382                                  boost::spirit::qi::unused_type,
00383                                  boost::spirit::qi::unused_type) const {
00384       std::ostringstream ostr;
00385       for (std::vector<char>::const_iterator lItVector = iChar.begin();
00386          lItVector != iChar.end();
00387          lItVector++) {
00388         ostr << *lItVector;
00389       }
00390       std::string classCodeStr = ostr.str();
00391       // Insertion of this class Code list in the whole classCode name
00392       _DCPRule._classCodeList.push_back(classCodeStr);
00393       // DEBUG
00394       // STDAIR_LOG_DEBUG ("Class Code: " << classCodeStr);
00395     }
00396
00397     // ////////////////////////////////////////////////////////////////////
00398     doEndDCP::
00399     doEndDCP (stdair::BomRoot& ioBomRoot,
00400               DCPRuleStruct& ioDCPRule)
00401       : ParserSemanticAction (ioDCPRule),
00402         _bomRoot (ioBomRoot) {
00403     }
00404
00405     // ////////////////////////////////////////////////////////////////////
00406     void doEndDCP::operator() (
      boost::spirit::qi::unused_type,
00407                                boost::spirit::qi::unused_type,
00408                                boost::spirit::qi::unused_type) const {
00409       // DEBUG
00410       // STDAIR_LOG_DEBUG ("Do End");
00411       // Generation of the DCP rule object.
00412       _DCPRule._classCodeListOfList.push_back(_DCPRule.
      _classCodeList);
00413       DCPRuleGenerator::createDCPRule (_bomRoot, _DCPRule);
00414       STDAIR_LOG_DEBUG(_DCPRule.describe());
00415     }
00416
00417     // ////////////////////////////////////////////////////////////////////
00418     //
00419     //  Utility Parsers
00420     //
00421     // ////////////////////////////////////////////////////////////////////
00423     namespace bsq = boost::spirit::qi;
00424     namespace bsa = boost::spirit::ascii;
00425
00427     stdair::int1_p_t int1_p;
00428
00430     stdair::uint2_p_t uint2_p;
00431
00433     stdair::uint4_p_t uint4_p;
00434
00436     stdair::uint1_4_p_t uint1_4_p;
00437
00439     stdair::hour_p_t hour_p;
00440     stdair::minute_p_t minute_p;
00441     stdair::second_p_t second_p;
00442
00444     stdair::year_p_t year_p;
00445     stdair::month_p_t month_p;
00446     stdair::day_p_t day_p;
00447
00448     // ////////////////////////////////////////////////////////////////////
```

```
00449     //  (Boost Spirit) Grammar Definition
00450     // ////////////////////////////////////////////////////////////////////
00451
00452     // ////////////////////////////////////////////////////////////////////
00453     DCPRuleParser::DCPRuleParser (stdair::BomRoot&
      ioBomRoot,
00454                                   DCPRuleStruct& ioDCPRule) :
00455       DCPRuleParser::base_type(start),
00456       _bomRoot(ioBomRoot), _DCPRule(ioDCPRule) {
00457
00458       start = *(comments | DCP_rule);
00459
00460       comments = (bsq::lexeme[bsq::repeat(2)[bsa::char_('/')]
00461                               >> +(bsa::char_ - bsq::eol)
00462                               >> bsq::eol]
00463                   | bsq::lexeme[bsa::char_('/') >>bsa::char_('*')
00464                                 >> +(bsa::char_ - bsa::char_('*'))
00465                                 >> bsa::char_('*') >> bsa::char_('/')]);
00466
00467       DCP_rule = DCP_key
00468         >> +( ';' >> segment )
00469         >> DCP_rule_end[doEndDCP(_bomRoot, _DCPRule
      )];
00470
00471       DCP_rule_end = bsa::char_(';');
00472
00473       DCP_key = DCP_id
00474         >> ';' >> origin >> ';' >> destination
00475         >> ';' >> dateRangeStart >> ';' >> dateRangeEnd
00476         >> ';' >> timeRangeStart >> ';' >> timeRangeEnd
00477         >> ';' >> position >>  ';' >> cabinCode >> ';' >>
      channel
00478         >> ';' >> advancePurchase >> ';' >> saturdayStay
00479         >> ';' >> changeFees >> ';' >> nonRefundable
00480         >> ';' >> minimumStay >> ';' >> DCP;
00481
00482       DCP_id = uint1_4_p[storeDCPId(_DCPRule)]
      ;
00483
00484       origin = bsq::repeat(3)[bsa::char_("A-Z")][storeOrigin(
      _DCPRule)];
00485
00486       destination =
00487         bsq::repeat(3)[bsa::char_("A-Z")][storeDestination(
      _DCPRule)];
00488
00489       dateRangeStart = date[storeDateRangeStart
      (_DCPRule)];
00490
00491       dateRangeEnd = date[storeDateRangeEnd(
      _DCPRule)];
00492
00493       date = bsq::lexeme
00494         [year_p[boost::phoenix::ref(_DCPRule._itYear) =
      bsq::labels::_1]
00495         >> '-'
00496         >> month_p[boost::phoenix::ref(_DCPRule._itMonth) =
      bsq::labels::_1]
00497         >> '-'
00498         >> day_p[boost::phoenix::ref(_DCPRule._itDay) =
      bsq::labels::_1] ];
00499
00500       timeRangeStart = time[storeStartRangeTime
      (_DCPRule)];
00501
00502       timeRangeEnd = time[storeEndRangeTime(
      _DCPRule)];
00503
00504       time =  bsq::lexeme
00505         [hour_p[boost::phoenix::ref(_DCPRule._itHours) =
      bsq::labels::_1]
00506         >> ':'
00507         >> minute_p[boost::phoenix::ref(_DCPRule._itMinutes) =
      bsq::labels::_1]
00508         >> - (':' >> second_p[boost::phoenix::ref(_DCPRule.
      _itSeconds) = bsq::labels::_1]) ];
00509
00510       position = bsq::repeat(3)[bsa::char_("A-Z")][storePOS(
      _DCPRule)];
00511
00512       cabinCode = bsa::char_("A-Z")[storeCabinCode(
      _DCPRule)];
00513
00514       channel = bsq::repeat(2)[bsa::char_("A-Z")][storeChannel
      (_DCPRule)];
00515
00516       advancePurchase = uint1_4_p[storeAdvancePurchase
```

```
      (_DCPRule)];
00517
00518       saturdayStay = bsa::char_("A-Z")[storeSaturdayStay
      (_DCPRule)];
00519
00520       changeFees = bsa::char_("A-Z")[storeChangeFees(
     _DCPRule)];
00521
00522       nonRefundable = bsa::char_("A-Z")[storeNonRefundable
      (_DCPRule)];
00523
00524       minimumStay = uint1_4_p[storeMinimumStay
      (_DCPRule)];
00525
00526       DCP = bsq::double_[storeDCP(_DCPRule)];
00527
00528       segment = bsq::repeat(2)[bsa::char_("A-Z")][storeAirlineCode
      (_DCPRule)]
00529         //>> ';'
00530         //>> bsa::char_("A-Z")[storeClass(_DCPRule)]
00531         >> +(';' >> list_class);
00532
00533       list_class = bsq::repeat(1,bsq::inf)[bsa::char_("A-Z")][
     storeClass(_DCPRule)];
00534
00535       //BOOST_SPIRIT_DEBUG_NODE (DCPRuleParser);
00536       BOOST_SPIRIT_DEBUG_NODE (start);
00537       BOOST_SPIRIT_DEBUG_NODE (comments);
00538       BOOST_SPIRIT_DEBUG_NODE (DCP_rule);
00539       BOOST_SPIRIT_DEBUG_NODE (DCP_rule_end);
00540       BOOST_SPIRIT_DEBUG_NODE (DCP_key);
00541       BOOST_SPIRIT_DEBUG_NODE (DCP_id);
00542       BOOST_SPIRIT_DEBUG_NODE (origin);
00543       BOOST_SPIRIT_DEBUG_NODE (destination);
00544       BOOST_SPIRIT_DEBUG_NODE (dateRangeStart);
00545       BOOST_SPIRIT_DEBUG_NODE (dateRangeEnd);
00546       BOOST_SPIRIT_DEBUG_NODE (date);
00547       BOOST_SPIRIT_DEBUG_NODE (timeRangeStart);
00548       BOOST_SPIRIT_DEBUG_NODE (timeRangeEnd);
00549       BOOST_SPIRIT_DEBUG_NODE (time);
00550       BOOST_SPIRIT_DEBUG_NODE (position);
00551       BOOST_SPIRIT_DEBUG_NODE (cabinCode);
00552       BOOST_SPIRIT_DEBUG_NODE (channel);
00553       BOOST_SPIRIT_DEBUG_NODE (advancePurchase);
00554       BOOST_SPIRIT_DEBUG_NODE (saturdayStay);
00555       BOOST_SPIRIT_DEBUG_NODE (changeFees);
00556       BOOST_SPIRIT_DEBUG_NODE (nonRefundable);
00557       BOOST_SPIRIT_DEBUG_NODE (minimumStay);
00558       BOOST_SPIRIT_DEBUG_NODE (DCP);
00559       BOOST_SPIRIT_DEBUG_NODE (segment);
00560       BOOST_SPIRIT_DEBUG_NODE (list_class);
00561     }
00562   }
00563
00565   //
00566   //  Entry class for the file parser
00567   //
00569
00570   // ////////////////////////////////////////////////////////////////////
00571   DCPRuleFileParser::
00572   DCPRuleFileParser (stdair::BomRoot& ioBomRoot,
00573                      const stdair::Filename_T& iFilename)
00574     : _filename (iFilename), _bomRoot (ioBomRoot) {
00575     init();
00576   }
00577
00578   // ////////////////////////////////////////////////////////////////////
00579   void DCPRuleFileParser::init() {
00580     // Check that the file exists and is readable
00581     const bool doesExistAndIsReadable =
00582       stdair::BasFileMgr::doesExistAndIsReadable (_filename);
00583
00584     if (doesExistAndIsReadable == false) {
00585       STDAIR_LOG_ERROR ("The DCP schedule file " << _filename
00586                         << " does not exist or can not be  read.");
00587
00588       throw DCPInputFileNotFoundException ("The DCP file " + _filename + " does
      not exist or can not be read");
00589     }
00590   }
00591
00592   // ////////////////////////////////////////////////////////////////////
00593   bool DCPRuleFileParser::generateDCPRules (
     ) {
00594
00595     STDAIR_LOG_DEBUG ("Parsing DCP input file: " << _filename);
00596
```

```
00597     // File to be parsed
00598     const std::string* lFileName = &_filename;
00599     const char *lChar = (*lFileName).c_str();
00600     std::ifstream fileToBeParsed(lChar, std::ios_base::in);
00601
00602     // Check the filename exists and can be open
00603     if (fileToBeParsed == false) {
00604       STDAIR_LOG_ERROR ("The DCP file " << _filename << " can not be open."
00605                         << std::endl);
00606
00607       throw DCPInputFileNotFoundException ("The file " + _filename + " does not
       exist or can not be read");
00608     }
00609
00610     // Create an input iterator
00611     stdair::base_iterator_t inputBegin (fileToBeParsed);
00612
00613     // Convert input iterator to an iterator usable by spirit parser
00614     stdair::iterator_t
00615       start (boost::spirit::make_default_multi_pass (inputBegin));
00616     stdair::iterator_t end;
00617
00618     // Initialise the parser (grammar) with the helper/staging structure.
00619     DCPParserHelper::DCPRuleParser lFPParser(
      _bomRoot, _DCPRule);
00620
00621     // Launch the parsing of the file and, thanks to the doEndDCP
00622     // call-back structure, the building of the whole BomRoot BOM
00623
00624     const bool hasParsingBeenSuccesful =
00625        boost::spirit::qi::phrase_parse (start, end, lFPParser,
00626                                         boost::spirit::ascii::space);
00627
00628     if (hasParsingBeenSuccesful == false) {
00629       // TODO: decide whether to throw an exceqption
00630       STDAIR_LOG_ERROR ("Parsing of DCP input file: " << _filename
00631                         << " failed");
00632     }
00633     if  (start != end) {
00634       // TODO: decide whether to throw an exception
00635       STDAIR_LOG_ERROR ("Parsing of DCP input file: " << _filename
00636                         << " failed");
00637     }
00638     if (hasParsingBeenSuccesful == true && start == end) {
00639       STDAIR_LOG_DEBUG ("Parsing of DCP input file: " << _filename
00640       << " succeeded");
00641     }
00642     return hasParsingBeenSuccesful;
00643   }
00644
00645 }
```

23.171 airinv/command/vault/DCPParserHelper.hpp File Reference

```
#include <stdair/basic/BasParserTypes.hpp>
#include <stdair/command/CmdAbstract.hpp>
#include <airinv/AIRINV_Types.hpp>
#include <airinv/bom/DCPRuleStruct.hpp>
```

Classes

- struct AIRINV::DCPParserHelper::ParserSemanticAction
- struct AIRINV::DCPParserHelper::storeDCPId
- struct AIRINV::DCPParserHelper::storeOrigin
- struct AIRINV::DCPParserHelper::storeDestination
- struct AIRINV::DCPParserHelper::storeDateRangeStart
- struct AIRINV::DCPParserHelper::storeDateRangeEnd
- struct AIRINV::DCPParserHelper::storeStartRangeTime
- struct AIRINV::DCPParserHelper::storeEndRangeTime
- struct AIRINV::DCPParserHelper::storePOS
- struct AIRINV::DCPParserHelper::storeCabinCode
- struct AIRINV::DCPParserHelper::storeChannel

- struct AIRINV::DCPParserHelper::storeAdvancePurchase
- struct AIRINV::DCPParserHelper::storeSaturdayStay
- struct AIRINV::DCPParserHelper::storeChangeFees
- struct AIRINV::DCPParserHelper::storeNonRefundable
- struct AIRINV::DCPParserHelper::storeMinimumStay
- struct AIRINV::DCPParserHelper::storeDCP
- struct AIRINV::DCPParserHelper::storeAirlineCode
- struct AIRINV::DCPParserHelper::storeClass
- struct AIRINV::DCPParserHelper::doEndDCP
- struct AIRINV::DCPParserHelper::DCPRuleParser
- class AIRINV::DCPRuleFileParser

Namespaces

- namespace stdair

  Forward declarations.
- namespace AIRINV
- namespace AIRINV::DCPParserHelper

23.172 DCPParserHelper.hpp

```
00001 #ifndef __AIRINV_CMD_DCPPARSERHELPER_HPP
00002 #define __AIRINV_CMD_DCPPARSERHELPER_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // StdAir
00008 // The stdair/basic/BasParserTypes.hpp header includes Boost.Spirit headers
00009 //#define BOOST_SPIRIT_DEBUG
00010 #include <stdair/basic/BasParserTypes.hpp>
00011 #include <stdair/command/CmdAbstract.hpp>
00012 // AirInv
00013 #include <airinv/AIRINV_Types.hpp>
00014 #include <airinv/bom/DCPRuleStruct.hpp>
00015
00016 // Forward declarations
00017 namespace stdair {
00018   class BomRoot;
00019 }
00020
00021 namespace AIRINV {
00022
00023   namespace DCPParserHelper {
00024
00025     // //////////////////////////////////////////////////////////////////
00026     //  Semantic actions
00027     // //////////////////////////////////////////////////////////////////
00029
00030     struct ParserSemanticAction {
00032       ParserSemanticAction (DCPRuleStruct&);
00034       DCPRuleStruct& _DCPRule;
00035     };
00036
00038     struct storeDCPId : public ParserSemanticAction
      {
00040       storeDCPId (DCPRuleStruct&);
00042       void operator() (unsigned int,
00043                        boost::spirit::qi::unused_type,
00044                        boost::spirit::qi::unused_type) const;
00045     };
00046
00048     struct storeOrigin : public ParserSemanticAction
      {
00050       storeOrigin (DCPRuleStruct&);
00052       void operator() (std::vector<char>,
00053                        boost::spirit::qi::unused_type,
00054                        boost::spirit::qi::unused_type) const;
00055     };
00056
00058     struct storeDestination : public ParserSemanticAction
      {
00060       storeDestination (DCPRuleStruct&);
```

```
00062       void operator() (std::vector<char>,
00063                        boost::spirit::qi::unused_type,
00064                        boost::spirit::qi::unused_type) const;
00065     };
00066
00068     struct storeDateRangeStart : public ParserSemanticAction
      {
00070       storeDateRangeStart (DCPRuleStruct&);
00072       void operator() (boost::spirit::qi::unused_type,
00073                        boost::spirit::qi::unused_type,
00074                        boost::spirit::qi::unused_type) const;
00075     };
00076
00078     struct storeDateRangeEnd : public ParserSemanticAction
      {
00080       storeDateRangeEnd (DCPRuleStruct&);
00082       void operator() (boost::spirit::qi::unused_type,
00083                        boost::spirit::qi::unused_type,
00084                        boost::spirit::qi::unused_type) const;
00085     };
00086
00088     struct storeStartRangeTime : public ParserSemanticAction
      {
00090       storeStartRangeTime (DCPRuleStruct&);
00092       void operator() (boost::spirit::qi::unused_type,
00093                        boost::spirit::qi::unused_type,
00094                        boost::spirit::qi::unused_type) const;
00095     };
00096
00098     struct storeEndRangeTime : public ParserSemanticAction
      {
00100       storeEndRangeTime (DCPRuleStruct&);
00102       void operator() (boost::spirit::qi::unused_type,
00103                        boost::spirit::qi::unused_type,
00104                        boost::spirit::qi::unused_type) const;
00105     };
00106
00108     struct storePOS : public ParserSemanticAction {
00110       storePOS (DCPRuleStruct&);
00112       void operator() (std::vector<char>,
00113                        boost::spirit::qi::unused_type,
00114                        boost::spirit::qi::unused_type) const;
00115     };
00116
00118     struct storeCabinCode : public ParserSemanticAction
      {
00120       storeCabinCode  (DCPRuleStruct&);
00122       void operator() (char,
00123                        boost::spirit::qi::unused_type,
00124                        boost::spirit::qi::unused_type) const;
00125     };
00126
00128     struct storeChannel : public ParserSemanticAction
      {
00130       storeChannel (DCPRuleStruct&);
00132       void operator() (std::vector<char>,
00133                        boost::spirit::qi::unused_type,
00134                        boost::spirit::qi::unused_type) const;
00135     };
00136
00138     struct storeAdvancePurchase : public
    ParserSemanticAction {
00140       storeAdvancePurchase (DCPRuleStruct&);
00142       void operator() (unsigned int,
00143                        boost::spirit::qi::unused_type,
00144                        boost::spirit::qi::unused_type) const;
00145     };
00146
00148     struct storeSaturdayStay : public ParserSemanticAction
      {
00150       storeSaturdayStay (DCPRuleStruct&);
00152       void operator() (char,
00153                        boost::spirit::qi::unused_type,
00154                        boost::spirit::qi::unused_type) const;
00155     };
00156
00158     struct storeChangeFees : public ParserSemanticAction
      {
00160       storeChangeFees (DCPRuleStruct&);
00162       void operator() (char,
00163                        boost::spirit::qi::unused_type,
00164                        boost::spirit::qi::unused_type) const;
00165     };
00166
00168     struct storeNonRefundable : public ParserSemanticAction
      {
00170       storeNonRefundable (DCPRuleStruct&);
```

```
00172       void operator() (char,
00173                        boost::spirit::qi::unused_type,
00174                        boost::spirit::qi::unused_type) const;
00175     };
00176
00178     struct storeMinimumStay : public ParserSemanticAction
    {
00180       storeMinimumStay (DCPRuleStruct&);
00182       void operator() (unsigned int,
00183                        boost::spirit::qi::unused_type,
00184                        boost::spirit::qi::unused_type) const;
00185     };
00186
00188     struct storeDCP : public ParserSemanticAction {
00190       storeDCP (DCPRuleStruct&);
00192       void operator() (double,
00193                        boost::spirit::qi::unused_type,
00194                        boost::spirit::qi::unused_type) const;
00195     };
00196
00198     struct storeAirlineCode : public ParserSemanticAction
    {
00200       storeAirlineCode (DCPRuleStruct&);
00202       void operator() (std::vector<char>,
00203                        boost::spirit::qi::unused_type,
00204                        boost::spirit::qi::unused_type) const;
00205     };
00206
00208     struct storeClass : public ParserSemanticAction
    {
00210       storeClass (DCPRuleStruct&);
00212       void operator() (std::vector<char>,
00213                        boost::spirit::qi::unused_type,
00214                        boost::spirit::qi::unused_type) const;
00215     };
00216
00218     struct doEndDCP : public ParserSemanticAction {
00220       doEndDCP (stdair::BomRoot&, DCPRuleStruct&);
00222       void operator() (boost::spirit::qi::unused_type,
00223                        boost::spirit::qi::unused_type,
00224                        boost::spirit::qi::unused_type) const;
00226       stdair::BomRoot& _bomRoot;
00227     };
00228
00229
00231     //
00232     //  (Boost Spirit) Grammar Definition
00233     //
00235
00304     struct DCPRuleParser :
00305       public boost::spirit::qi::grammar<stdair::iterator_t,
00306                                         boost::spirit::ascii::space_type> {
00307
00308       DCPRuleParser (stdair::BomRoot&, DCPRuleStruct&);
00309
00310       // Instantiation of rules
00311       boost::spirit::qi::rule<stdair::iterator_t,
00312                               boost::spirit::ascii::space_type>
00313       start, comments, DCP_rule, DCP_rule_end,
      DCP_key, DCP_id, origin,
00314         destination, dateRangeStart, dateRangeEnd
      , date, timeRangeStart,
00315         timeRangeEnd, time, position, cabinCode
      , channel, advancePurchase,
00316         saturdayStay, changeFees, nonRefundable
      , minimumStay, DCP,
00317         segment, list_class;
00318
00319       // Parser Context
00320       stdair::BomRoot& _bomRoot;
00321       DCPRuleStruct& _DCPRule;
00322     };
00323
00324   }
00325
00327   //
00328   //  Entry class for the file parser
00329   //
00331
00337   class DCPRuleFileParser : public stdair::CmdAbstract {
00338   public:
00340     DCPRuleFileParser (stdair::BomRoot& ioBomRoot,
00341                        const stdair::Filename_T& iFilename);
00342
00344     bool generateDCPRules ();
00345
00346   private:
```

```
00348     void init();
00349
00350   private:
00351     // Attributes
00353     stdair::Filename_T _filename;
00354
00356     stdair::BomRoot& _bomRoot;
00357
00359     DCPRuleStruct _DCPRule;
00360   };
00361
00362 }
00363 #endif // __AIRINV_CMD_DCPPARSERHELPER_HPP
```

23.173 airinv/config/airinv-paths.hpp File Reference

Macros

- #define PACKAGE "airinv"
- #define PACKAGE_NAME "AIRINV"
- #define PACKAGE_VERSION "1.00.0"
- #define PREFIXDIR "/usr"
- #define EXEC_PREFIX "/usr"
- #define BINDIR "/usr/bin"
- #define LIBDIR "/usr/lib"
- #define LIBEXECDIR "/usr/libexec"
- #define SBINDIR "/usr/sbin"
- #define SYSCONFDIR "/usr/etc"
- #define INCLUDEDIR "/usr/include"
- #define DATAROOTDIR "/usr/share"
- #define DATADIR "/usr/share"
- #define DOCDIR "/usr/share/doc/airinv-1.00.0"
- #define MANDIR "/usr/share/man"
- #define INFODIR "/usr/share/info"
- #define HTMLDIR "/usr/share/doc/airinv-1.00.0/html"
- #define PDFDIR "/usr/share/doc/airinv-1.00.0/html"
- #define STDAIR_SAMPLE_DIR "/usr/share/stdair/samples"

23.173.1 Macro Definition Documentation

23.173.1.1 #define PACKAGE "airinv"

Definition at line 4 of file airinv-paths.hpp.

23.173.1.2 #define PACKAGE_NAME "AIRINV"

Definition at line 5 of file airinv-paths.hpp.

23.173.1.3 #define PACKAGE_VERSION "1.00.0"

Definition at line 6 of file airinv-paths.hpp.

23.173.1.4 #define PREFIXDIR "/usr"

Definition at line 7 of file airinv-paths.hpp.

23.173.1.5 #define EXEC_PREFIX "/usr"

Definition at line 8 of file airinv-paths.hpp.

23.173.1.6 #define BINDIR "/usr/bin"

Definition at line 9 of file airinv-paths.hpp.

23.173.1.7 #define LIBDIR "/usr/lib"

Definition at line 10 of file airinv-paths.hpp.

23.173.1.8 #define LIBEXECDIR "/usr/libexec"

Definition at line 11 of file airinv-paths.hpp.

23.173.1.9 #define SBINDIR "/usr/sbin"

Definition at line 12 of file airinv-paths.hpp.

23.173.1.10 #define SYSCONFDIR "/usr/etc"

Definition at line 13 of file airinv-paths.hpp.

23.173.1.11 #define INCLUDEDIR "/usr/include"

Definition at line 14 of file airinv-paths.hpp.

23.173.1.12 #define DATAROOTDIR "/usr/share"

Definition at line 15 of file airinv-paths.hpp.

23.173.1.13 #define DATADIR "/usr/share"

Definition at line 16 of file airinv-paths.hpp.

23.173.1.14 #define DOCDIR "/usr/share/doc/airinv-1.00.0"

Definition at line 17 of file airinv-paths.hpp.

23.173.1.15 #define MANDIR "/usr/share/man"

Definition at line 18 of file airinv-paths.hpp.

23.173.1.16 #define INFODIR "/usr/share/info"

Definition at line 19 of file airinv-paths.hpp.

23.173.1.17 #define HTMLDIR "/usr/share/doc/airinv-1.00.0/html"

Definition at line 20 of file airinv-paths.hpp.

23.173.1.18 #define PDFDIR "/usr/share/doc/airinv-1.00.0/html"

Definition at line 21 of file airinv-paths.hpp.

23.173.1.19 #define STDAIR_SAMPLE_DIR "/usr/share/stdair/samples"

Definition at line 22 of file airinv-paths.hpp.

23.174 airinv-paths.hpp

```
00001 #ifndef __AIRINV_PATHS_HPP__
00002 #define __AIRINV_PATHS_HPP__
00003
00004 #define PACKAGE "airinv"
00005 #define PACKAGE_NAME "AIRINV"
```

```
00006 #define PACKAGE_VERSION "1.00.0"
00007 #define PREFIXDIR "/usr"
00008 #define EXEC_PREFIX "/usr"
00009 #define BINDIR "/usr/bin"
00010 #define LIBDIR "/usr/lib"
00011 #define LIBEXECDIR "/usr/libexec"
00012 #define SBINDIR "/usr/sbin"
00013 #define SYSCONFDIR "/usr/etc"
00014 #define INCLUDEDIR "/usr/include"
00015 #define DATAROOTDIR "/usr/share"
00016 #define DATADIR "/usr/share"
00017 #define DOCDIR "/usr/share/doc/airinv-1.00.0"
00018 #define MANDIR "/usr/share/man"
00019 #define INFODIR "/usr/share/info"
00020 #define HTMLDIR "/usr/share/doc/airinv-1.00.0/html"
00021 #define PDFDIR "/usr/share/doc/airinv-1.00.0/html"
00022 #define STDAIR_SAMPLE_DIR "/usr/share/stdair/samples"
00023
00024 #endif // __AIRINV_PATHS_HPP__
```

23.175 airinv/config/airinv-paths.hpp.in File Reference

Macros

- #define __AIRINV_PATHS_HPP__
- #define PACKAGE "@PACKAGE@"
- #define PACKAGE_NAME "@PACKAGE_NAME@"
- #define PACKAGE_VERSION "@PACKAGE_VERSION@"
- #define PREFIXDIR "@prefix@"
- #define EXEC_PREFIX "@exec_prefix@"
- #define BINDIR "@bindir@"
- #define LIBDIR "@libdir@"
- #define LIBEXECDIR "@libexecdir@"
- #define SBINDIR "@sbindir@"
- #define SYSCONFDIR "@sysconfdir@"
- #define INCLUDEDIR "@includedir@"
- #define DATAROOTDIR "@datarootdir@"
- #define DATADIR "@datadir@"
- #define DOCDIR "@docdir@"
- #define MANDIR "@mandir@"
- #define INFODIR "@infodir@"
- #define HTMLDIR "@htmldir@"
- #define PDFDIR "@pdfdir@"
- #define STDAIR_SAMPLE_DIR "@sampledir@"

23.175.1 Macro Definition Documentation

23.175.1.1 #define __AIRINV_PATHS_HPP__

Definition at line 2 of file airinv-paths.hpp.in.

23.175.1.2 #define PACKAGE "@PACKAGE@"

Definition at line 4 of file airinv-paths.hpp.in.

23.175.1.3 #define PACKAGE_NAME "@PACKAGE_NAME@"

Definition at line 5 of file airinv-paths.hpp.in.

23.175.1.4 #define PACKAGE_VERSION "@PACKAGE_VERSION@"

Definition at line 6 of file airinv-paths.hpp.in.

23.175.1.5 #define PREFIXDIR "@prefix@"

Definition at line 7 of file airinv-paths.hpp.in.

23.175.1.6 #define EXEC_PREFIX "@exec_prefix@"

Definition at line 8 of file airinv-paths.hpp.in.

23.175.1.7 #define BINDIR "@bindir@"

Definition at line 9 of file airinv-paths.hpp.in.

23.175.1.8 #define LIBDIR "@libdir@"

Definition at line 10 of file airinv-paths.hpp.in.

23.175.1.9 #define LIBEXECDIR "@libexecdir@"

Definition at line 11 of file airinv-paths.hpp.in.

23.175.1.10 #define SBINDIR "@sbindir@"

Definition at line 12 of file airinv-paths.hpp.in.

23.175.1.11 #define SYSCONFDIR "@sysconfdir@"

Definition at line 13 of file airinv-paths.hpp.in.

23.175.1.12 #define INCLUDEDIR "@includedir@"

Definition at line 14 of file airinv-paths.hpp.in.

23.175.1.13 #define DATAROOTDIR "@datarootdir@"

Definition at line 15 of file airinv-paths.hpp.in.

23.175.1.14 #define DATADIR "@datadir@"

Definition at line 16 of file airinv-paths.hpp.in.

23.175.1.15 #define DOCDIR "@docdir@"

Definition at line 17 of file airinv-paths.hpp.in.

23.175.1.16 #define MANDIR "@mandir@"

Definition at line 18 of file airinv-paths.hpp.in.

23.175.1.17 #define INFODIR "@infodir@"

Definition at line 19 of file airinv-paths.hpp.in.

23.175.1.18 #define HTMLDIR "@htmldir@"

Definition at line 20 of file airinv-paths.hpp.in.

23.175.1.19 #define PDFDIR "@pdfdir@"

Definition at line 21 of file airinv-paths.hpp.in.

23.175.1.20 #define STDAIR_SAMPLE_DIR "@sampledir@"

Definition at line 22 of file airinv-paths.hpp.in.

23.176 airinv-paths.hpp.in

```
00001 #ifndef __AIRINV_PATHS_HPP__
00002 #define __AIRINV_PATHS_HPP__
00003
00004 #define PACKAGE "@PACKAGE@"
00005 #define PACKAGE_NAME "@PACKAGE_NAME@"
00006 #define PACKAGE_VERSION "@PACKAGE_VERSION@"
00007 #define PREFIXDIR "@prefix@"
00008 #define EXEC_PREFIX "@exec_prefix@"
00009 #define BINDIR "@bindir@"
00010 #define LIBDIR "@libdir@"
00011 #define LIBEXECDIR "@libexecdir@"
00012 #define SBINDIR "@sbindir@"
00013 #define SYSCONFDIR "@sysconfdir@"
00014 #define INCLUDEDIR "@includedir@"
00015 #define DATAROOTDIR "@datarootdir@"
00016 #define DATADIR "@datadir@"
00017 #define DOCDIR "@docdir@"
00018 #define MANDIR "@mandir@"
00019 #define INFODIR "@infodir@"
00020 #define HTMLDIR "@htmldir@"
00021 #define PDFDIR "@pdfdir@"
00022 #define STDAIR_SAMPLE_DIR "@sampledir@"
00023
00024 #endif // __AIRINV_PATHS_HPP__
```

23.177 airinv/factory/FacAirinvMasterServiceContext.cpp File Reference

```
#include <cassert>
#include <stdair/service/FacSupervisor.hpp>
#include <airinv/factory/FacAirinvMasterServiceContext.hpp>
#include <airinv/service/AIRINV_Master_ServiceContext.hpp>
```

Namespaces

- namespace AIRINV

23.178 FacAirinvMasterServiceContext.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // StdAir
00007 #include <stdair/service/FacSupervisor.hpp>
00008 // AirInv
00009 #include <airinv/factory/FacAirinvMasterServiceContext.hpp
      >
00010 #include <airinv/service/AIRINV_Master_ServiceContext.hpp
      >
00011
00012 namespace AIRINV {
00013
00014   FacAirinvMasterServiceContext* FacAirinvMasterServiceContext::_instance =
      NULL;
00015
00016   // ////////////////////////////////////////////////////////////////////
00017   FacAirinvMasterServiceContext::~FacAirinvMasterServiceContext
      () {
00018     _instance = NULL;
00019   }
00020
00021   // ////////////////////////////////////////////////////////////////////
00022   FacAirinvMasterServiceContext&
      FacAirinvMasterServiceContext::instance(
```

```
    ) {
00023
00024     if (_instance == NULL) {
00025       _instance = new FacAirinvMasterServiceContext
    ();
00026       assert (_instance != NULL);
00027
00028       stdair::FacSupervisor::instance().
    registerServiceFactory (_instance);
00029     }
00030     return *_instance;
00031   }
00032
00033   // ////////////////////////////////////////////////////////////////////
00034   AIRINV_Master_ServiceContext&
    FacAirinvMasterServiceContext::create() {
00035     AIRINV_Master_ServiceContext*
    aAIRINV_Master_ServiceContext_ptr = NULL;
00036
00037     aAIRINV_Master_ServiceContext_ptr = new AIRINV_Master_ServiceContext
    ();
00038     assert (aAIRINV_Master_ServiceContext_ptr != NULL);
00039
00040     // The new object is added to the Bom pool
00041     _pool.push_back (aAIRINV_Master_ServiceContext_ptr);
00042
00043     return *aAIRINV_Master_ServiceContext_ptr;
00044   }
00045
00046 }
```

23.179 airinv/factory/FacAirinvMasterServiceContext.hpp File Reference

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

Classes

- class AIRINV::FacAirinvMasterServiceContext

  Factory for Bucket.

Namespaces

- namespace AIRINV

23.180 FacAirinvMasterServiceContext.hpp

```
00001 #ifndef __AIRINV_FAC_FACAIRINVMASTERSERVICECONTEXT_HPP
00002 #define __AIRINV_FAC_FACAIRINVMASTERSERVICECONTEXT_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/service/FacServiceAbstract.hpp>
00011
00012 namespace AIRINV {
00013
00015   class AIRINV_Master_ServiceContext;
00016
00020   class FacAirinvMasterServiceContext : public
    stdair::FacServiceAbstract {
00021   public:
00022
00026     static FacAirinvMasterServiceContext& instance
    ();
00027
00032     ~FacAirinvMasterServiceContext();
00033
00037     AIRINV_Master_ServiceContext& create();
```

```
00038
00039
00040   protected:
00044     FacAirinvMasterServiceContext() {}
00045
00046   private:
00048     static FacAirinvMasterServiceContext*
      _instance;
00049   };
00050
00051 }
00052 #endif // __AIRINV_FAC_FACAIRINVMASTERSERVICECONTEXT_HPP
```

23.181 airinv/factory/FacAirinvServiceContext.cpp File Reference

```
#include <cassert>
#include <stdair/service/FacSupervisor.hpp>
#include <airinv/factory/FacAirinvServiceContext.hpp>
#include <airinv/service/AIRINV_ServiceContext.hpp>
```

Namespaces

- namespace AIRINV

23.182 FacAirinvServiceContext.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // StdAir
00007 #include <stdair/service/FacSupervisor.hpp>
00008 // AirInv
00009 #include <airinv/factory/FacAirinvServiceContext.hpp
      >
00010 #include <airinv/service/AIRINV_ServiceContext.hpp
      >
00011
00012 namespace AIRINV {
00013
00014   FacAirinvServiceContext* FacAirinvServiceContext::_instance = NULL;
00015
00016   // ////////////////////////////////////////////////////////////////////
00017   FacAirinvServiceContext::~FacAirinvServiceContext
      () {
00018     _instance = NULL;
00019   }
00020
00021   // ////////////////////////////////////////////////////////////////////
00022   FacAirinvServiceContext&
      FacAirinvServiceContext::instance() {
00023
00024     if (_instance == NULL) {
00025       _instance = new FacAirinvServiceContext();
00026       assert (_instance != NULL);
00027
00028       stdair::FacSupervisor::instance().
      registerServiceFactory (_instance);
00029     }
00030     return *_instance;
00031   }
00032
00033   // ////////////////////////////////////////////////////////////////////
00034   AIRINV_ServiceContext& FacAirinvServiceContext::create
      () {
00035     AIRINV_ServiceContext* aAIRINV_ServiceContext_ptr =
      NULL;
00036
00037     aAIRINV_ServiceContext_ptr = new AIRINV_ServiceContext
      ();
00038     assert (aAIRINV_ServiceContext_ptr != NULL);
00039
00040     // The new object is added to the Bom pool
00041     _pool.push_back (aAIRINV_ServiceContext_ptr);
```

```
00042
00043     return *aAIRINV_ServiceContext_ptr;
00044   }
00045
00046 }
```

23.183 airinv/factory/FacAirinvServiceContext.hpp File Reference

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

Classes

- class AIRINV::FacAirinvServiceContext

Namespaces

- namespace AIRINV

23.184 FacAirinvServiceContext.hpp

```
00001 #ifndef __AIRINV_FAC_FACAIRINVSERVICECONTEXT_HPP
00002 #define __AIRINV_FAC_FACAIRINVSERVICECONTEXT_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/service/FacServiceAbstract.hpp>
00011
00012 namespace AIRINV {
00013
00015   class AIRINV_ServiceContext;
00016
00018   class FacAirinvServiceContext : public
      stdair::FacServiceAbstract {
00019   public:
00020
00024     static FacAirinvServiceContext& instance();
00025
00030     ~FacAirinvServiceContext();
00031
00035     AIRINV_ServiceContext& create();
00036
00037
00038   protected:
00042     FacAirinvServiceContext() {}
00043
00044   private:
00046     static FacAirinvServiceContext* _instance;
00047   };
00048
00049 }
00050 #endif // __AIRINV_FAC_FACAIRINVSERVICECONTEXT_HPP
```

23.185 airinv/factory/FacBomAbstract.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/functional/hash/hash.hpp>
#include <airinv/bom/BomAbstract.hpp>
#include <airinv/factory/FacBomAbstract.hpp>
```

Namespaces

- namespace AIRINV

23.186 FacBomAbstract.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost (STL Extension)
00008 #include <boost/functional/hash/hash.hpp>
00009 // Airinv
00010 #include <airinv/bom/BomAbstract.hpp>
00011 #include <airinv/factory/FacBomAbstract.hpp>
00012
00013 namespace AIRINV {
00014
00015   // ////////////////////////////////////////////////////////////////////
00016   FacBomAbstract::~FacBomAbstract() {
00017     clean ();
00018   }
00019
00020   // ////////////////////////////////////////////////////////////////////
00021   void FacBomAbstract::clean() {
00022     for (BomPool_T::iterator itBom = _pool.begin();
00023          itBom != _pool.end(); itBom++) {
00024       BomAbstract* currentBom_ptr = *itBom;
00025       assert (currentBom_ptr != NULL);
00026
00027       delete (currentBom_ptr); currentBom_ptr = NULL;
00028     }
00029
00030     // Empty the pool of Factories
00031     _pool.clear();
00032   }
00033
00034   // ////////////////////////////////////////////////////////////////////
00035   std::size_t FacBomAbstract::getID (const BomAbstract
      * iBomAbstract_ptr) {
00036     const void* lPtr = iBomAbstract_ptr;
00037     boost::hash<const void*> ptr_hash;
00038     const std::size_t lID = ptr_hash (lPtr);
00039     return lID;
00040   }
00041
00042   // ////////////////////////////////////////////////////////////////////
00043   std::size_t FacBomAbstract::getID (const BomAbstract
      & iBomAbstract) {
00044     return getID (&iBomAbstract);
00045   }
00046
00047   // ////////////////////////////////////////////////////////////////////
00048   std::string FacBomAbstract::getIDString(const
      BomAbstract* iBomAbstract_ptr) {
00049     const std::size_t lID = getID (iBomAbstract_ptr);
00050     std::ostringstream oStr;
00051     oStr << lID;
00052     return oStr.str();
00053   }
00054
00055   // ////////////////////////////////////////////////////////////////////
00056   std::string FacBomAbstract::getIDString (const
      BomAbstract& iBomAbstract) {
00057     return getIDString (&iBomAbstract);
00058   }
00059
00060 }
```

23.187 airinv/factory/FacBomAbstract.hpp File Reference

```
#include <string>
#include <vector>
```

Classes

- class AIRINV::FacBomAbstract

Namespaces

- namespace AIRINV

23.188 FacBomAbstract.hpp

```
00001 #ifndef __AIRINV_FAC_FACBOMABSTRACT_HPP
00002 #define __AIRINV_FAC_FACBOMABSTRACT_HPP
00003
00004 // //////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010
00011 namespace AIRINV {
00012
00013   // Forward declarations
00014   class BomAbstract;
00015
00017   class FacBomAbstract {
00018     friend class FacSupervisor;
00019   public:
00020
00022     typedef std::vector<BomAbstract*> BomPool_T;
00023
00025     static std::size_t getID (const BomAbstract*);
00026
00028     static std::size_t getID (const BomAbstract&);
00029
00032     static std::string getIDString (const BomAbstract*);
00033
00036     static std::string getIDString (const BomAbstract&);
00037
00038   protected:
00041     FacBomAbstract() {}
00042     FacBomAbstract(const FacBomAbstract&) {}
00043
00045     virtual ~FacBomAbstract();
00046
00047   private:
00049     void clean();
00050
00051   protected:
00053     BomPool_T _pool;
00054   };
00055 }
00056 #endif // __AIRINV_FAC_FACBOMABSTRACT_HPP
```

23.189 airinv/factory/FacServiceAbstract.cpp File Reference

```
#include <cassert>
#include <airinv/service/ServiceAbstract.hpp>
#include <airinv/factory/FacServiceAbstract.hpp>
```

Namespaces

- namespace AIRINV

23.190 FacServiceAbstract.cpp

```
00001 // //////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////////
```

```
00004 // STL
00005 #include <cassert>
00006 // AIRINV
00007 #include <airinv/service/ServiceAbstract.hpp>
00008 #include <airinv/factory/FacServiceAbstract.hpp
      >
00009
00010 namespace AIRINV {
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 }
```

23.191 airinv/factory/FacServiceAbstract.hpp File Reference

```
#include <vector>
```

Classes

- class AIRINV::FacServiceAbstract

Namespaces

- namespace AIRINV

23.192 FacServiceAbstract.hpp

```
00001 #ifndef __AIRINV_FAC_FACSERVICEABSTRACT_HPP
00002 #define __AIRINV_FAC_FACSERVICEABSTRACT_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <vector>
00009
00010 namespace AIRINV {
00011
00012   // Forward declarations
00013   class ServiceAbstract;
00014
00016   class FacServiceAbstract {
00017   public:
00018
00020     typedef std::vector<ServiceAbstract*> ServicePool_T;
00021
00023     virtual ~FacServiceAbstract();
00024
00026     void clean();
00027
00028   protected:
00031     FacServiceAbstract() {}
00032
00034     ServicePool_T _pool;
00035   };
```

```
00036
00037 }
00038 #endif // __AIRINV_FAC_FACSERVICEABSTRACT_HPP
```

23.193 airinv/factory/FacSupervisor.cpp File Reference

```
#include <cassert>
#include <airinv/factory/FacBomAbstract.hpp>
#include <airinv/factory/FacServiceAbstract.hpp>
#include <airinv/factory/FacSupervisor.hpp>
```

Namespaces

- namespace AIRINV

23.194 FacSupervisor.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // AIRINV
00007 #include <airinv/factory/FacBomAbstract.hpp>
00008 #include <airinv/factory/FacServiceAbstract.hpp
      >
00009 #include <airinv/factory/FacSupervisor.hpp>
00010
00011 namespace AIRINV {
00012
00013   FacSupervisor* FacSupervisor::_instance = NULL;
00014
00015   // ////////////////////////////////////////////////////////////////////
00016   FacSupervisor::FacSupervisor () {
00017   }
00018
00019   // ////////////////////////////////////////////////////////////////////
00020   FacSupervisor& FacSupervisor::instance()
      {
00021     if (_instance == NULL) {
00022       _instance = new FacSupervisor();
00023     }
00024
00025     return *_instance;
00026   }
00027
00028   // ////////////////////////////////////////////////////////////////////
00029   void FacSupervisor::
00030   registerBomFactory (FacBomAbstract*
      ioFacBomAbstract_ptr) {
00031     _bomPool.push_back (ioFacBomAbstract_ptr);
00032   }
00033
00034   // ////////////////////////////////////////////////////////////////////
00035   void FacSupervisor::
00036   registerServiceFactory (FacServiceAbstract
      * ioFacServiceAbstract_ptr) {
00037     _svcPool.push_back (ioFacServiceAbstract_ptr);
00038   }
00039
00040   // ////////////////////////////////////////////////////////////////////
00041   FacSupervisor::~FacSupervisor() {
00042     cleanBomLayer();
00043     cleanServiceLayer();
00044   }
00045
00046   // ////////////////////////////////////////////////////////////////////
00047   void FacSupervisor::cleanBomLayer() {
00048     for (BomFactoryPool_T::const_iterator itFactory = _bomPool.begin();
00049          itFactory != _bomPool.end(); itFactory++) {
00050       const FacBomAbstract* currentFactory_ptr = *itFactory;
00051       assert (currentFactory_ptr != NULL);
00052
00053       delete (currentFactory_ptr); currentFactory_ptr = NULL;
00054     }
```

```
00055
00056     // Empty the pool of Bom Factories
00057     _bomPool.clear();
00058   }
00059
00060   // ////////////////////////////////////////////////////////////////////
00061   void FacSupervisor::cleanServiceLayer() {
00062     for (ServiceFactoryPool_T::const_iterator itFactory = _svcPool.begin();
00063          itFactory != _svcPool.end(); itFactory++) {
00064       const FacServiceAbstract* currentFactory_ptr = *
      itFactory;
00065       assert (currentFactory_ptr != NULL);
00066
00067       delete (currentFactory_ptr); currentFactory_ptr = NULL;
00068     }
00069
00070     // Empty the pool of Service Factories
00071     _svcPool.clear();
00072   }
00073
00074   // ////////////////////////////////////////////////////////////////////
00075   void FacSupervisor::cleanFactory () {
00076         if (_instance != NULL) {
00077                 _instance->cleanBomLayer();
00078                 _instance->cleanServiceLayer();
00079         }
00080     delete (_instance); _instance = NULL;
00081   }
00082
00083 }
```

23.195 airinv/factory/FacSupervisor.hpp File Reference

```
#include <vector>
```

Classes

- class AIRINV::FacSupervisor

Namespaces

- namespace AIRINV

23.196 FacSupervisor.hpp

```
00001 #ifndef __AIRINV_FAC_FACSUPERVISOR_HPP
00002 #define __AIRINV_FAC_FACSUPERVISOR_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <vector>
00009
00010 namespace AIRINV {
00011
00012   // Forward declarations
00013   class FacBomAbstract;
00014   class FacServiceAbstract;
00015
00017   class FacSupervisor {
00018   public:
00019
00021     typedef std::vector<FacBomAbstract*> BomFactoryPool_T;
00022       typedef std::vector<FacServiceAbstract*> ServiceFactoryPool_T
      ;
00023
00027     static FacSupervisor& instance();
00028
00033     void registerBomFactory (FacBomAbstract*);
00034
00039     void registerServiceFactory (FacServiceAbstract
      *);
```

```
00040
00044     void cleanBomLayer();
00045
00049     void cleanServiceLayer();
00050
00053     static void cleanFactory ();
00054
00058     ~FacSupervisor();
00059
00060
00061   protected:
00065     FacSupervisor ();
00066     FacSupervisor (const FacSupervisor&) {}
00067
00068
00069   private:
00071     static FacSupervisor* _instance;
00072
00074     BomFactoryPool_T _bomPool;
00075
00077     ServiceFactoryPool_T _svcPool;
00078   };
00079 }
00080 #endif // __AIRINV_FAC_FACSUPERVISOR_HPP
```

23.197 airinv/FlightRequestStatus.hpp File Reference

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

Classes

- struct AIRINV::FlightRequestStatus

Namespaces

- namespace AIRINV

23.198 FlightRequestStatus.hpp

```
00001 #ifndef __AIRINV_BAS_FLIGHTREQUESTSTATUS_HPP
00002 #define __AIRINV_BAS_FLIGHTREQUESTSTATUS_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/basic/StructAbstract.hpp>
00011
00012 namespace AIRINV {
00013
00015   struct FlightRequestStatus : public stdair::StructAbstract
      {
00016   public:
00017     typedef enum {
00018       OK = 0,
00019       NOT_FOUND,
00020       INTERNAL_ERROR,
00021       LAST_VALUE
00022     } EN_FlightRequestStatus;
00023
00025     static const std::string& getLabel (const EN_FlightRequestStatus
      &);
00026
00028     static const std::string& getCodeLabel (const
      EN_FlightRequestStatus&);
00029
00031     static std::string describeLabels();
00032
00034     EN_FlightRequestStatus getCode() const;
00035
```

```
00037     const std::string describe() const;
00038
00039
00040   public:
00042     FlightRequestStatus (const EN_FlightRequestStatus
      &);
00044     FlightRequestStatus (const std::string& iCode);
00045
00046
00047   private:
00049     static const std::string _labels[LAST_VALUE];
00051     static const std::string _codeLabels[LAST_VALUE];
00052
00053
00054   private:
00055     // //////// Attributes /////////
00057     EN_FlightRequestStatus _code;
00058   };
00059
00060 }
00061 #endif // __AIRINV_BAS_FLIGHTREQUESTSTATUS_HPP
```

23.199 airinv/server/AirInvClient.cpp File Reference

```
#include <string>
#include <iostream>
#include <zmq.hpp>
```

Functions

- int main (int argc, char ∗argv[])

23.199.1 Function Documentation

23.199.1.1 int main (int *argc*, char ∗ *argv[]*)

Definition at line 11 of file AirInvClient.cpp.

23.200 AirInvClient.cpp

```
00001 // ///////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ///////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <string>
00006 #include <iostream>
00007 // ZeroMQ
00008 #include <zmq.hpp>
00009
00010 // ////////////////// M A I N //////////////////////
00011 int main (int argc, char* argv[]) {
00012   // Prepare our context and socket
00013   zmq::context_t context (1);
00014   zmq::socket_t socket (context, ZMQ_REQ);
00015
00016   std::cout << "Connecting to hello world server..." << std::endl;
00017   socket.connect ("tcp://localhost:5555");
00018
00019   // Do 10 requests, waiting each time for a response
00020   for (int request_nbr = 0; request_nbr != 10; request_nbr++) {
00021     zmq::message_t request (6);
00022     memcpy ((void *) request.data (), "Hello", 5);
00023     std::cout << "Sending Hello " << request_nbr << "..." << std::endl;
00024     socket.send (request);
00025
00026     // Get the reply.
00027     zmq::message_t reply;
00028     socket.recv (&reply);
00029     std::cout << "Received World " << request_nbr << std::endl;
00030   }
00031   return 0;
00032 }
```

23.201 airinv/server/AirInvClient_ASIO.cpp File Reference

```
#include <cassert>
#include <iostream>
#include <string>
#include <boost/asio.hpp>
#include <boost/array.hpp>
```

Functions

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

23.201.1 Function Documentation

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

Definition at line 14 of file AirInvClient_ASIO.cpp.

23.202 AirInvClient_ASIO.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <iostream>
00007 #include <string>
00008 // Boost.ASIO
00009 #include <boost/asio.hpp>
00010 // Boost.Array
00011 #include <boost/array.hpp>
00012
00013 // ////////// M A I N /////////////////
00014 int main (int argc, char* argv[]) {
00015
00016   // Host name
00017   std::string lHostname = "localhost";
00018
00019   // Service name (as specified within /etc/services)
00020   // The "aria" service corresponds to the port 2624
00021   const std::string lServiceName = "aria";
00022
00023   try {
00024
00025     if (argc >= 2) {
00026       lHostname = argv[1];
00027     }
00028
00029     boost::asio::io_service lIOService;
00030
00031     boost::asio::ip::tcp::resolver lResolver (lIOService);
00032
00033     boost::asio::ip::tcp::resolver::query lQuery (lHostname, lServiceName);
00034
00035     boost::asio::ip::tcp::resolver::iterator itEndPoint =
00036       lResolver.resolve (lQuery);
00037     boost::asio::ip::tcp::resolver::iterator lEnd;
00038
00039     boost::asio::ip::tcp::socket lSocket (lIOService);
00040     boost::system::error_code lError = boost::asio::error::host_not_found;
00041
00042     //
00043     while (lError && itEndPoint != lEnd) {
00044       const boost::asio::ip::tcp::endpoint lEndPoint = *itEndPoint;
00045
00046       // DEBUG
00047       std::cout << "Testing end point: " << std::endl;
00048
00049       lSocket.close();
00050       lSocket.connect (lEndPoint, lError);
00051       ++itEndPoint;
00052     }
00053
```

```
00054     //
00055     if (lError) {
00056       throw boost::system::system_error (lError);
00057     }
00058     assert (!lError);
00059
00060     // DEBUG
00061     const boost::asio::ip::tcp::endpoint lValidEndPoint;
00062     std::cout << "Valid end point: " << lValidEndPoint << std::endl;
00063
00064     // Send a message to the server
00065     const std::string lMessage ("Hello AirInv Server!");
00066     boost::asio::write (lSocket, boost::asio::buffer (lMessage),
00067                         boost::asio::transfer_all(), lError);
00068
00069     // Read the reply from the server
00070     boost::array<char, 256> lBuffer;
00071
00072     size_t lLength = lSocket.read_some (boost::asio::buffer(lBuffer), lError);
00073
00074     // Some other error than connection closed cleanly by peer
00075     if (lError && lError != boost::asio::error::eof) {
00076       throw boost::system::system_error (lError);
00077     }
00078
00079     // DEBUG
00080     std::cout << "Reply from the server: ";
00081     std::cout.write (lBuffer.data(), lLength);
00082     std::cout << std::endl;
00083
00084   } catch (std::exception& lException) {
00085     std::cerr << lException.what() << std::endl;
00086   }
00087
00088   return 0;
00089 }
```

23.203 airinv/server/AirInvServer.cpp File Reference

23.204 AirInvServer.cpp

```
00001
00005 // ////////////////////////////////////////////////////////////////////
00006 // Import section
00007 // ////////////////////////////////////////////////////////////////////
00008 // STL
00009 #include <cassert>
00010 #include <sstream>
00011 #include <fstream>
00012 #include <string>
00013 #include <unistd.h>
00014 // Boost (Extended STL)
00015 #include <boost/program_options.hpp>
00016 #include <boost/tokenizer.hpp>
00017 // ZeroMQ
00018 #include <zmq.hpp>
00019 // StdAir
00020 #include <stdair/basic/BasLogParams.hpp>
00021 #include <stdair/basic/BasDBParams.hpp>
00022 #include <stdair/service/Logger.hpp>
00023 #include <stdair/stdair_json.hpp>
00024 // AirInvServer
00025 #include <airinv/config/airinv-paths.hpp>
00026 #include <airinv/AIRINV_Master_Service.hpp>
00027
00028 // ///////// Type definitions ///////////
00029 typedef unsigned int ServerPort_T;
00030
00031 // //////// Constants //////
00033 const std::string K_AIRINV_DEFAULT_LOG_FILENAME ("airinvServer.log");
00034
00036 const std::string K_AIRINV_DEFAULT_SERVER_PROTOCOL ("tcp://");
00037
00039 const std::string K_AIRINV_DEFAULT_SERVER_ADDRESS ("*");
00040
00042 const ServerPort_T K_AIRINV_DEFAULT_SERVER_PORT (5555);
00043
00045 const std::string K_AIRINV_DEFAULT_INVENTORY_FILENAME (STDAIR_SAMPLE_DIR
00046                                                         "/invdump01.csv");
00048 const std::string K_AIRINV_DEFAULT_SCHEDULE_FILENAME (STDAIR_SAMPLE_DIR
00049                                                        "/schedule01.csv");
00051 const std::string K_AIRINV_DEFAULT_OND_FILENAME (STDAIR_SAMPLE_DIR
00052                                                  "/ond01.csv");
```

```
00054 const std::string K_AIRINV_DEFAULT_FRAT5_INPUT_FILENAME (STDAIR_SAMPLE_DIR
00055                                                         "/frat5.csv");
00057 const std::string K_AIRINV_DEFAULT_FF_DISUTILITY_INPUT_FILENAME (
      STDAIR_SAMPLE_DIR
00058                                                               "
      /ffDisutility.csv");
00059
00061 const std::string K_AIRINV_DEFAULT_YIELD_FILENAME (STDAIR_SAMPLE_DIR
00062                                                    "/yield01.csv");
00063
00068 const bool K_AIRINV_DEFAULT_BUILT_IN_INPUT = false;
00069
00074 const bool K_AIRINV_DEFAULT_FOR_SCHEDULE = false;
00075
00079 const int K_AIRINV_EARLY_RETURN_STATUS = 99;
00080
00084 struct Command_T {
00085   typedef enum {
00086     NOP = 0,
00087     QUIT,
00088     DISPLAY,
00089     SELL,
00090     LAST_VALUE
00091   } Type_T;
00092 };
00093
00094 // ********* Parsing of Options & Configuration *********
00095 // A helper function to simplify the main part.
00096 template<class T> std::ostream& operator<< (std::ostream& os,
00097                                             const std::vector<T>& v) {
00098   std::copy (v.begin(), v.end(), std::ostream_iterator<T> (std::cout, " "));
00099   return os;
00100 }
00101
00103 int readConfiguration (int argc, char* argv[], std::string& ioServerProtocol,
00104                        std::string& ioServerAddress, ServerPort_T& ioServerPort
      ,
00105                        bool& ioIsBuiltin, bool& ioIsForSchedule,
00106                        stdair::Filename_T& ioInventoryFilename,
00107                        stdair::Filename_T& ioScheduleInputFilename,
00108                        stdair::Filename_T& ioODInputFilename,
00109                        stdair::Filename_T& ioFRAT5Filename,
00110                        stdair::Filename_T& ioFFDisutilityFilename,
00111                        stdair::Filename_T& ioYieldInputFilename,
00112                        std::string& ioLogFilename) {
00113   // Default for the built-in input
00114   ioIsBuiltin = K_AIRINV_DEFAULT_BUILT_IN_INPUT;
00115
00116   // Default for the inventory or schedule option
00117   ioIsForSchedule = K_AIRINV_DEFAULT_FOR_SCHEDULE;
00118
00119   // Declare a group of options that will be allowed only on command line
00120   boost::program_options::options_description generic ("Generic options");
00121   generic.add_options()
00122     ("prefix", "print installation prefix")
00123     ("version,v", "print version string")
00124     ("help,h", "produce help message");
00125
00126   // Declare a group of options that will be allowed both on command
00127   // line and in config file
00128
00129   boost::program_options::options_description config ("Configuration");
00130   config.add_options()
00131     ("builtin,b",
00132      "The sample BOM tree can be either built-in or parsed from an input file.
       That latter must then be given with the -i/--inventory or -s/--schedule option")
00133     ("for_schedule,f",
00134      "The BOM tree should be built from a schedule file (instead of from an
       inventory dump)")
00135     ("inventory,i",
00136      boost::program_options::value< std::string >(&ioInventoryFilename)->
      default_value(K_AIRINV_DEFAULT_INVENTORY_FILENAME),
00137      "(CVS) input file for the inventory")
00138     ("schedule,s",
00139      boost::program_options::value< std::string >(&ioScheduleInputFilename)->
      default_value(K_AIRINV_DEFAULT_SCHEDULE_FILENAME),
00140      "(CVS) input file for the schedule")
00141     ("ond,o",
00142      boost::program_options::value< std::string >(&ioODInputFilename)->
      default_value(K_AIRINV_DEFAULT_OND_FILENAME),
00143      "(CVS) input file for the O&D")
00144     ("frat5,r",
00145      boost::program_options::value< std::string >(&ioFRAT5Filename)->
      default_value(K_AIRINV_DEFAULT_FRAT5_INPUT_FILENAME),
00146      "(CSV) input file for the FRAT5 Curve")
00147     ("ff_disutility,d",
00148      boost::program_options::value< std::string >(&ioFFDisutilityFilename)->
```

```
      default_value(K_AIRINV_DEFAULT_FF_DISUTILITY_INPUT_FILENAME),
00149      "(CSV) input file for the FF disutility Curve")
00150     ("yield,y",
00151      boost::program_options::value< std::string >(&ioYieldInputFilename)->
      default_value(K_AIRINV_DEFAULT_YIELD_FILENAME),
00152      "(CVS) input file for the yield")
00153     ("protocol,t",
00154      boost::program_options::value< std::string >(&ioServerProtocol)->
      default_value(K_AIRINV_DEFAULT_SERVER_PROTOCOL),
00155      "Server protocol")
00156     ("address,a",
00157      boost::program_options::value< std::string >(&ioServerAddress)->
      default_value(K_AIRINV_DEFAULT_SERVER_ADDRESS),
00158      "Server address")
00159     ("port,p",
00160      boost::program_options::value< ServerPort_T >(&ioServerPort)->
      default_value(K_AIRINV_DEFAULT_SERVER_PORT),
00161      "Server port")
00162     ("log,l",
00163      boost::program_options::value< std::string >(&ioLogFilename)->
      default_value(K_AIRINV_DEFAULT_LOG_FILENAME),
00164      "Filename for the output logs")
00165     ;
00166
00167   // Hidden options, will be allowed both on command line and
00168   // in config file, but will not be shown to the user.
00169   boost::program_options::options_description hidden ("Hidden options");
00170   hidden.add_options()
00171     ("copyright",
00172      boost::program_options::value< std::vector<std::string> >(),
00173      "Show the copyright (license)");
00174
00175   boost::program_options::options_description cmdline_options;
00176   cmdline_options.add(generic).add(config).add(hidden);
00177
00178   boost::program_options::options_description config_file_options;
00179   config_file_options.add(config).add(hidden);
00180   boost::program_options::options_description visible ("Allowed options");
00181   visible.add(generic).add(config);
00182
00183   boost::program_options::positional_options_description p;
00184   p.add ("copyright", -1);
00185
00186   boost::program_options::variables_map vm;
00187   boost::program_options::
00188     store (boost::program_options::command_line_parser (argc, argv).
00189            options (cmdline_options).positional(p).run(), vm);
00190
00191   std::ifstream ifs ("airinvServer.cfg");
00192   boost::program_options::store (parse_config_file (ifs, config_file_options),
00193                                  vm);
00194   boost::program_options::notify (vm);
00195
00196   if (vm.count ("help")) {
00197     std::cout << visible << std::endl;
00198     return K_AIRINV_EARLY_RETURN_STATUS;
00199   }
00200
00201   if (vm.count ("version")) {
00202     std::cout << PACKAGE_NAME << ", version " << PACKAGE_VERSION
      << std::endl;
00203     return K_AIRINV_EARLY_RETURN_STATUS;
00204   }
00205
00206   if (vm.count ("prefix")) {
00207     std::cout << "Installation prefix: " << PREFIXDIR << std::endl;
00208     return K_AIRINV_EARLY_RETURN_STATUS;
00209   }
00210
00211   if (vm.count ("protocol")) {
00212     ioServerProtocol = vm["protocol"].as< std::string >();
00213     std::cout << "Server protocol is: " << ioServerProtocol << std::endl;
00214   }
00215
00216   if (vm.count ("address")) {
00217     ioServerAddress = vm["address"].as< std::string >();
00218     std::cout << "Server address is: " << ioServerAddress << std::endl;
00219   }
00220
00221   if (vm.count ("port")) {
00222     ioServerPort = vm["port"].as< ServerPort_T >();
00223     std::cout << "Server port is: " << ioServerPort << std::endl;
00224   }
00225
00226   if (vm.count ("builtin")) {
00227     ioIsBuiltin = true;
00228   }
```

```
00229   const std::string isBuiltinStr = (ioIsBuiltin == true)?"yes":"no";
00230   std::cout << "The BOM should be built-in? " << isBuiltinStr << std::endl;
00231
00232   if (vm.count ("for_schedule")) {
00233     ioIsForSchedule = true;
00234   }
00235   const std::string isForScheduleStr = (ioIsForSchedule == true)?"yes":"no";
00236   std::cout << "The BOM should be built from schedule? " << isForScheduleStr
00237             << std::endl;
00238
00239   if (ioIsBuiltin == false) {
00240
00241     if (ioIsForSchedule == false) {
00242       // The BOM tree should be built from parsing an inventory dump
00243       if (vm.count ("inventory")) {
00244         ioInventoryFilename = vm["inventory"].as< std::string >();
00245         std::cout << "Input inventory filename is: " << ioInventoryFilename
00246                   << std::endl;
00247
00248       } else {
00249         // The built-in option is not selected. However, no inventory dump
00250         // file is specified
00251         std::cerr << "Either one among the -b/--builtin, -i/--inventory or "
00252                   << " -f/--for_schedule and -s/--schedule options "
00253                   << "must be specified" << std::endl;
00254       }
00255
00256     } else {
00257       // The BOM tree should be built from parsing a schedule (and O&D) file
00258       if (vm.count ("schedule")) {
00259         ioScheduleInputFilename = vm["schedule"].as< std::string >();
00260         std::cout << "Input schedule filename is: " << ioScheduleInputFilename
00261                   << std::endl;
00262
00263       } else {
00264         // The built-in option is not selected. However, no schedule file
00265         // is specified
00266         std::cerr << "Either one among the -b/--builtin, -i/--inventory or "
00267                   << " -f/--for_schedule and -s/--schedule options "
00268                   << "must be specified" << std::endl;
00269       }
00270
00271       if (vm.count ("ond")) {
00272         ioODInputFilename = vm["ond"].as< std::string >();
00273         std::cout << "Input O&D filename is: " << ioODInputFilename <<
      std::endl;
00274       }
00275
00276       if (vm.count ("frat5")) {
00277         ioFRAT5Filename = vm["frat5"].as< std::string >();
00278         std::cout << "FRAT5 input filename is: " << ioFRAT5Filename <<
      std::endl;
00279
00280       }
00281
00282       if (vm.count ("ff_disutility")) {
00283         ioFFDisutilityFilename = vm["ff_disutility"].as< std::string >();
00284         std::cout << "FF disutility input filename is: "
00285                   << ioFFDisutilityFilename << std::endl;
00286
00287       }
00288
00289       if (vm.count ("yield")) {
00290         ioYieldInputFilename = vm["yield"].as< std::string >();
00291         std::cout << "Input yield filename is: " << ioYieldInputFilename <<
      std::endl;
00292       }
00293     }
00294   }
00295
00296   if (vm.count ("log")) {
00297     ioLogFilename = vm["log"].as< std::string >();
00298     std::cout << "Log filename is: " << ioLogFilename << std::endl;
00299   }
00300
00301   return 0;
00302 }
00303
00304
00305 // ///////// Utility functions on top of the ZeroMQ library /////////
00309 static std::string s_recv (zmq::socket_t& socket) {
00310   zmq::message_t message;
00311   socket.recv (&message);
00312
00313   return std::string (static_cast<char*> (message.data()), message.size());
00314 }
00315
```

```
00319 static bool s_send (zmq::socket_t& socket, const std::string& string) {
00320   zmq::message_t message (string.size());
00321   memcpy (message.data(), string.data(), string.size());
00322
00323   bool rc = socket.send (message);
00324   return rc;
00325 }
00326
00327
00328 // ////////////////////// M A I N /////////////////////////
00329 int main (int argc, char* argv[]) {
00330
00331   // Server parameters (for ZeroMQ)
00332   std::string ioServerProtocol;
00333   std::string ioServerAddress;
00334   ServerPort_T ioServerPort;
00335
00336   // State whether the BOM tree should be built-in or parsed from an
00337   // input file
00338   bool isBuiltin;
00339   bool isForSchedule;
00340
00341   // Input file names
00342   stdair::Filename_T lInventoryFilename;
00343   stdair::Filename_T lScheduleInputFilename;
00344   stdair::Filename_T lODInputFilename;
00345   stdair::Filename_T lFRAT5InputFilename;
00346   stdair::Filename_T lFFDisutilityInputFilename;
00347   stdair::Filename_T lYieldInputFilename;
00348
00349   // Output log File
00350   stdair::Filename_T lLogFilename;
00351
00352   // Call the command-line option parser
00353   const int lOptionParserStatus =
00354     readConfiguration (argc, argv, ioServerProtocol, ioServerAddress,
00355                        ioServerPort, isBuiltin, isForSchedule,
00356                        lInventoryFilename, lScheduleInputFilename,
00357                        lODInputFilename, lFRAT5InputFilename,
00358                        lFFDisutilityInputFilename, lYieldInputFilename,
00359                        lLogFilename);
00360
00361   if (lOptionParserStatus == K_AIRINV_EARLY_RETURN_STATUS) {
00362     return 0;
00363   }
00364
00365   // Set the log parameters
00366   std::ofstream logOutputFile;
00367   // Open and clean the log outputfile
00368   logOutputFile.open (lLogFilename.c_str());
00369   logOutputFile.clear();
00370
00371   // Initialise the inventory service
00372   const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
00373   AIRINV::AIRINV_Master_Service airinvService (
      lLogParams);
00374
00375   // DEBUG
00376   STDAIR_LOG_DEBUG ("Initialisation of the AirInv server");
00377
00378   // Check wether or not a (CSV) input file should be read
00379   if (isBuiltin == true) {
00380
00381     // Build the sample BOM tree for RMOL
00382     airinvService.buildSampleBom();
00383
00384   } else {
00385     if (isForSchedule == true) {
00386       // Build the BOM tree from parsing a schedule file (and O&D list)
00387       stdair::ScheduleFilePath lScheduleFilePath (lScheduleInputFilename);
00388       stdair::ODFilePath lODFilePath (lODInputFilename);
00389       stdair::FRAT5FilePath lFRAT5FilePath (lFRAT5InputFilename);
00390       stdair::FFDisutilityFilePath lFFDisutilityFilePath (
      lFFDisutilityInputFilename);
00391       AIRRAC::YieldFilePath lYieldFilePath (lYieldInputFilename);
00392       airinvService.parseAndLoad (lScheduleFilePath, lODFilePath,
00393                                   lFRAT5FilePath, lFFDisutilityFilePath,
00394                                   lYieldFilePath);
00395
00396     } else {
00397       // Build the BOM tree from parsing an inventory dump file
00398       AIRINV::InventoryFilePath lInventoryFilePath (
      lInventoryFilename);
00399       airinvService.parseAndLoad (lInventoryFilePath);
00400     }
00401   }
00402
```

```
00403   // Build the connection string (e.g., "tcp://*:5555", which is the default)
00404   std::ostringstream oZeroMQBindStream;
00405   oZeroMQBindStream << ioServerProtocol << ioServerAddress
00406                     << ":" << ioServerPort;
00407   const std::string lZeroMQBindString (oZeroMQBindStream.str());
00408
00409   // Prepare the context and socket of the server
00410   zmq::context_t context (1);
00411   zmq::socket_t socket (context, ZMQ_REP);
00412   socket.bind (lZeroMQBindString.c_str());
00413
00414   // DEBUG
00415   STDAIR_LOG_DEBUG ("The AirInv server is ready to receive requests...");
00416
00417   while (true) {
00418
00419     // Wait for next request from client, which is expected to give
00420     // a JSON-ified command.
00421     const std::string& lReceivedString = s_recv (socket);
00422
00423     // DEBUG
00424     STDAIR_LOG_DEBUG ("Received: '" << lReceivedString << "'");
00425
00426     const stdair::JSONString lJSONCommandString (lReceivedString);
00427     const std::string& lJSONDump =
00428       airinvService.jsonHandler (lJSONCommandString);
00429
00430     // DEBUG
00431     STDAIR_LOG_DEBUG ("Send: '" << lJSONDump << "'");
00432
00433     // Send back the answer details to the client
00434     s_send (socket, lJSONDump);
00435   }
00436
00437   return 0;
00438 }
00439
```

23.205 airinv/server/AirInvServer.hpp File Reference

```
#include <string>
#include <vector>
#include <boost/asio.hpp>
#include <boost/noncopyable.hpp>
#include <boost/shared_ptr.hpp>
#include <stdair/stdair_basic_types.hpp>
#include <airinv/server/Connection.hpp>
#include <airinv/server/RequestHandler.hpp>
```

Classes

- class AIRINV::AirInvServer

Namespaces

- namespace AIRINV

23.206 AirInvServer.hpp

```
00001 #ifndef __AIRINV_SVR_AIRINVSERVER_HPP
00002 #define __AIRINV_SVR_AIRINVSERVER_HPP
00003
00004 // //////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010 // Boost
00011 #include <boost/asio.hpp>
```

```
00012 #include <boost/noncopyable.hpp>
00013 #include <boost/shared_ptr.hpp>
00014 // StdAir
00015 #include <stdair/stdair_basic_types.hpp>
00016 // AirInv
00017 #include <airinv/server/Connection.hpp>
00018 #include <airinv/server/RequestHandler.hpp>
00019
00020 namespace AIRINV {
00021
00023   class AirInvServer : private boost::noncopyable {
00024   public:
00025     // ////////////// Constructors and Destructors /////////////////
00029     AirInvServer (const std::string& address, const std::string&
      port,
00030                   const stdair::AirlineCode_T& iAirlineCode,
00031                   std::size_t thread_pool_size);
00033     ~AirInvServer();
00034
00035
00036   public:
00037     // ////////////////// Business Methods ////////////////////
00039     void run();
00040
00042     void stop();
00043
00044
00045   private:
00046     // ////////////// Constructors and Destructors /////////////////
00048     AirInvServer();
00049     AirInvServer(const AirInvServer&);
00050
00051
00052   private:
00053     // ////////////// Attributes /////////////////
00055     void handleAccept (const boost::system::error_code& e);
00056
00058     std::size_t _threadPoolSize;
00059
00061     boost::asio::io_service _ioService;
00062
00064     boost::asio::ip::tcp::acceptor _acceptor;
00065
00067     ConnectionShrPtr_T _newConnection;
00068
00070     RequestHandler _requestHandler;
00071   };
00072
00073 }
00074 #endif // __AIRINV_SVR_AIRINVSERVER_HPP
```

23.207 airinv/server/AirInvServer_ASIO.cpp File Reference

```
#include <cassert>
#include <boost/thread.hpp>
#include <boost/bind.hpp>
#include <airinv/server/AirInvServer.hpp>
```

Namespaces

- namespace AIRINV

Typedefs

- typedef boost::shared_ptr < boost::thread > AIRINV::ThreadShrPtr_T
- typedef std::vector < ThreadShrPtr_T > AIRINV::ThreadShrPtrList_T

23.208 AirInvServer_ASIO.cpp

```
00001 // //////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // Boost
00007 #include <boost/thread.hpp>
00008 #include <boost/bind.hpp>
00009 // AirInv
00010 #include <airinv/server/AirInvServer.hpp>
00011
00012 namespace AIRINV {
00013
00014   // Type definitions
00015   typedef boost::shared_ptr<boost::thread> ThreadShrPtr_T;
00016   typedef std::vector<ThreadShrPtr_T> ThreadShrPtrList_T;
00017
00018
00019   // //////////////////////////////////////////////////////////////////////
00020   AirInvServer::AirInvServer (const std::string& address,
00021                               const std::string& port,
00022                               const stdair::AirlineCode_T& iAirlineCode,
00023                               std::size_t iThreadPoolSize)
00024     : _threadPoolSize (iThreadPoolSize), _acceptor (_ioService),
00025       _newConnection (new Connection (_ioService, _requestHandler)),
00026       _requestHandler (iAirlineCode) {
00027
00028     // Open the acceptor with the option to reuse the address
00029     // (i.e. SO_REUSEADDR).
00030     boost::asio::ip::tcp::resolver resolver (_ioService);
00031     boost::asio::ip::tcp::resolver::query query (address, port);
00032     boost::asio::ip::tcp::endpoint endpoint = *resolver.resolve(query);
00033
00034     _acceptor.open (endpoint.protocol());
00035     _acceptor.set_option (boost::asio::ip::tcp::acceptor::reuse_address(true));
00036     _acceptor.bind (endpoint);
00037     _acceptor.listen();
00038
00039     assert (_newConnection != NULL);
00040     _acceptor.async_accept (_newConnection->socket(),
00041                             boost::bind (&AirInvServer::handleAccept, this,
00042                                          boost::asio::placeholders::error));
00043   }
00044
00045   // //////////////////////////////////////////////////////////////////////
00046   AirInvServer::~AirInvServer () {
00047   }
00048
00049   // //////////////////////////////////////////////////////////////////////
00050   void AirInvServer::run() {
00051     // Create a pool of threads to run all of the io_services.
00052     ThreadShrPtrList_T lThreadList;
00053
00054     for (std::size_t itThread = 0; itThread != _threadPoolSize; ++itThread) {
00055       ThreadShrPtr_T lThread (new boost::thread (boost::bind (&
      boost::asio::io_service::run,
00056                                                               &_ioService)));
00057       lThreadList.push_back (lThread);
00058     }
00059
00060     // Wait for all threads in the pool to exit.
00061     for (std::size_t itThread = 0; itThread != lThreadList.size(); ++itThread)
     {
00062       boost::shared_ptr<boost::thread> lThread_ptr = lThreadList.at (itThread);
00063       assert (lThread_ptr != NULL);
00064       lThread_ptr->join();
00065     }
00066   }
00067
00068   // //////////////////////////////////////////////////////////////////////
00069   void AirInvServer::stop() {
00070     _ioService.stop();
00071   }
00072
00073   // //////////////////////////////////////////////////////////////////////
00074   void AirInvServer::handleAccept (const boost::system::error_code& iError) {
00075
00076     if (!iError) {
00077
00078       assert (_newConnection != NULL);
00079
00080       // The Connection object now takes in charge reading an incoming
00081       // message from the socket, and writing back a message.
00082       _newConnection->start();
00083
```

```
00084       // The (Boost) shared pointer is resetted to a newly allocated Connection
00085       // object. As the older Connection object is no longer pointed to, it is
00086       // deleted by the shared pointer mechanism.
00087       _newConnection.reset (new Connection (_ioService,
    _requestHandler));
00088
00089       _acceptor.async_accept (_newConnection->socket(),
00090                               boost::bind (&AirInvServer::handleAccept, this,
00091                                            boost::asio::placeholders::error));
00092     }
00093   }
00094
00095 }
```

23.209 airinv/server/BomPropertyTree.cpp File Reference

```
#include <boost/property_tree/ptree.hpp>
#include <boost/property_tree/json_parser.hpp>
#include <boost/foreach.hpp>
#include <airinv/server/BomPropertyTree.hpp>
```

Namespaces

- namespace stdair

  Forward declarations.

23.210 BomPropertyTree.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // Boost Property Tree
00005 #include <boost/property_tree/ptree.hpp>
00006 #include <boost/property_tree/json_parser.hpp>
00007 // Boost ForEach
00008 #include <boost/foreach.hpp>
00009 // AirInvServer
00010 #include <airinv/server/BomPropertyTree.hpp>
00011
00012 namespace bpt = boost::property_tree;
00013
00014 namespace stdair {
00015
00016   // Loads BomPropertyTree structure from the specified JSON file
00017   void BomPropertyTree::load (const std::string& iBomTree)
      {
00018     // Create an empty property tree object
00019     bpt::ptree pt;
00020
00021     // Load the JSON formatted string into the property tree. If reading fails
00022     // (cannot open stream, parse error), an exception is thrown.
00023     std::istringstream iStr (iBomTree);
00024     read_json (iStr, pt);
00025
00026     // Get the airline_code and store it in the _airlineCode variable.
00027     // Note that we construct the path to the value by separating
00028     // the individual keys with dots. If dots appear in the keys,
00029     // a path type with a different separator can be used.
00030     // If the flight_date.airline_code key is not found, an exception is
     thrown.
00031     _airlineCode = pt.get<stdair::AirlineCode_T> ("
    flight_date.airline_code");
00032
00033     // Get the departure_date and store it in the _departureDate variable.
00034     // This is another version of the get method: if the value is
00035     // not found, the default value (specified by the second
00036     // parameter) is returned instead. The type of the value
00037     // extracted is determined by the type of the second parameter,
00038     // so we can simply write get(...) instead of get<int>(...).
00039     _flightNumber =
00040       pt.get<stdair::FlightNumber_T> ("flight_date.flight_number", 100);
00041
00042     const std::string& lDepartureDateStr =
00043       pt.get<std::string> ("flight_date.departure_date");
```

```
00044     _departureDate = boost::gregorian::from_simple_string (
      lDepartureDateStr);
00045
00046     // Iterate over the flight_date.airport_codes section and store all found
00047     // codes in the _airportCodeList set. The get_child() function
00048     // returns a reference to the child at the specified path; if
00049     // there is no such child, it throws. Property tree iterators
00050     // are models of BidirectionalIterator.
00051     /*
00052     BOOST_FOREACH (bpt::ptree::value_type &v,
00053                    pt.get_child ("flight_date.airport_codes")) {
00054       _airportCodeList.insert (v.second.data());
00055     }
00056     */
00057   }
00058
00059   // Saves the BomPropertyTree structure to the specified JSON file
00060   std::string BomPropertyTree::save() const {
00061     std::ostringstream oStr;
00062
00063     // Create an empty property tree object
00064     bpt::ptree pt;
00065
00066     // Put airline code in property tree
00067     pt.put ("flight_date.airline_code", _airlineCode);
00068
00069     // Put flight number level in property tree
00070     pt.put ("flight_date.flight_number", _flightNumber);
00071
00072     // Put the flight departure date in property tree
00073     const std::string& lDepartureDateStr =
00074       boost::gregorian::to_simple_string (_departureDate);
00075     pt.put ("flight_date.departure_date", lDepartureDateStr);
00076
00077     // Iterate over the airport codes in the set and put them in the
00078     // property tree. Note that the put function places the new
00079     // key at the end of the list of keys. This is fine most of
00080     // the time. If you want to place an item at some other place
00081     // (i.e. at the front or somewhere in the middle), this can
00082     // be achieved using a combination of the insert and put_own
00083     // functions.
00084     bpt::ptree lAirportCodeArray;
00085     BOOST_FOREACH (const std::string& name, _airportCodeList) {
00086       lAirportCodeArray.push_back (std::pair<bpt::ptree::key_type,
00087                                              bpt::ptree::data_type> ("", name))
      ;
00088     }
00089     pt.put_child ("flight_date.airport_codes", lAirportCodeArray);
00090     //pt.push_back (std::make_pair ("flight_date.airport_codes",
       lAirportCodeArray));
00091
00092     // Write the property tree to the JSON stream.
00093     write_json (oStr, pt);
00094
00095     return oStr.str();
00096   }
00097
00098 }
```

23.211 airinv/server/BomPropertyTree.hpp File Reference

```
#include <string>
#include <set>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
```

Classes

- struct stdair::BomPropertyTree

Namespaces

- namespace stdair

  Forward declarations.

23.212 BomPropertyTree.hpp

```
00001 #ifndef __AIRINV_SVR_BOMPROPERTYTREE_HPP
00002 #define __AIRINV_SVR_BOMPROPERTYTREE_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <set>
00010 // StdAir
00011 #include <stdair/stdair_basic_types.hpp>
00012 #include <stdair/stdair_date_time_types.hpp>
00013
00014 namespace stdair {
00015
00019   struct BomPropertyTree {
00024     void load (const std::string& iBomTree);
00025
00029     std::string save() const;
00030
00031     // ************* Attributes ************
00033     stdair::AirlineCode_T _airlineCode;
00034
00036     stdair::FlightNumber_T _flightNumber;
00037
00039     stdair::Date_T _departureDate;
00040
00042     std::set<stdair::AirportCode_T> _airportCodeList;
00043   };
00044
00045 }
00046 #endif // __AIRINV_SVR_BOMPROPERTYTREE_HPP
```

23.213 airinv/server/Connection.cpp File Reference

```
#include <cassert>
#include <vector>
#include <boost/bind.hpp>
#include <airinv/server/RequestHandler.hpp>
#include <airinv/server/Connection.hpp>
```

Namespaces

- namespace AIRINV

23.214 Connection.cpp

```
00001 // //////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <vector>
00007 // Boost
00008 #include <boost/bind.hpp>
00009 // AirInv
00010 #include <airinv/server/RequestHandler.hpp>
00011 #include <airinv/server/Connection.hpp>
00012
00013 namespace AIRINV {
00014
00015   // //////////////////////////////////////////////////////////////////
00016   Connection::Connection (boost::asio::io_service&
      ioService,
00017                           RequestHandler& ioHandler)
00018     : _strand (ioService), _socket (ioService), _requestHandler (ioHandler) {
00019   }
00020
00021   // //////////////////////////////////////////////////////////////////
00022   boost::asio::ip::tcp::socket& Connection::socket() {
00023     return _socket;
00024   }
```

```
00025
00026   // ////////////////////////////////////////////////////////////////////
00027   void Connection::start() {
00028
00029     _socket.async_read_some (boost::asio::buffer (_buffer),
00030                              _strand.wrap (boost::bind (&Connection::handleRead
    ,
00031                                                         shared_from_this(),
00032
    boost::asio::placeholders::error,
00033
    boost::asio::placeholders::bytes_transferred)));
00034   }
00035
00036   // ////////////////////////////////////////////////////////////////////
00037   void Connection::handleRead (const boost::system::error_code& iErrorCode,
00038                                std::size_t bytes_transferred) {
00039     if (!iErrorCode) {
00040       _request._flightDetails = _buffer.data();
00041       const bool hasBeenSuccessfull = _requestHandler.handleRequest
     (_request,
00042                                                                    _reply);
00043
00044       if (hasBeenSuccessfull == true) {
00045
00046         boost::asio::async_write (_socket, _reply.to_buffers(),
00047                                   _strand.wrap (boost::bind (&
    Connection::handleWrite,
00048                                                              shared_from_this()
    ,
00049
    boost::asio::placeholders::error)));
00050
00051       } else {
00052
00053         boost::asio::async_write (_socket, _reply.to_buffers(),
00054                                   _strand.wrap (boost::bind (&
    Connection::handleWrite,
00055                                                              shared_from_this()
    ,
00056
    boost::asio::placeholders::error)));
00057
00058       }
00059     }
00060
00061     // If an error occurs then no new asynchronous operations are
00062     // started. This means that all shared_ptr references to the
00063     // connection object will disappear and the object will be
00064     // destroyed automatically after this handler returns. The
00065     // connection class's destructor closes the socket.
00066   }
00067
00068   // ////////////////////////////////////////////////////////////////////
00069   void Connection::handleWrite (const boost::system::error_code& iErrorCode) {
00070
00071     if (!iErrorCode) {
00072       // Initiate graceful connection closure.
00073       boost::system::error_code ignored_ec;
00074       _socket.shutdown (boost::asio::ip::tcp::socket::shutdown_both,
00075                         ignored_ec);
00076     }
00077
00078     // No new asynchronous operations are started. This means that all
00079     // shared_ptr references to the connection object will disappear
00080     // and the object will be destroyed automatically after this
00081     // handler returns. The connection class's destructor closes the
00082     // socket.
00083   }
00084
00085 }
```

23.215 airinv/server/Connection.hpp File Reference

```
#include <boost/asio.hpp>
#include <boost/array.hpp>
#include <boost/noncopyable.hpp>
#include <boost/shared_ptr.hpp>
#include <boost/enable_shared_from_this.hpp>
#include <airinv/server/Reply.hpp>
#include <airinv/server/Request.hpp>
```

Classes

- class AIRINV::Connection

Namespaces

- namespace AIRINV

Typedefs

- typedef boost::shared_ptr < Connection > AIRINV::ConnectionShrPtr_T

23.216 Connection.hpp

```
00001 #ifndef __AIRINV_SVR_CONNECTION_HPP
00002 #define __AIRINV_SVR_CONNECTION_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 // Boost
00009 #include <boost/asio.hpp>
00010 #include <boost/array.hpp>
00011 #include <boost/noncopyable.hpp>
00012 #include <boost/shared_ptr.hpp>
00013 #include <boost/enable_shared_from_this.hpp>
00014 // AirInv
00015 #include <airinv/server/Reply.hpp>
00016 #include <airinv/server/Request.hpp>
00017
00018 namespace AIRINV {
00019
00020   // Forward declarations.
00021   class RequestHandler;
00022
00023
00025   class Connection : public boost::enable_shared_from_this<Connection
      >,
00026                      private boost::noncopyable {
00027   public:
00028     // ////////////// Constructors and Destructors /////////////////
00031     Connection (boost::asio::io_service&, RequestHandler
      &);
00032
00033
00034     // /////////// Business Support Methods ////////////////
00036     boost::asio::ip::tcp::socket& socket();
00037
00039     void start();
00040
00041
00042   private:
00044     void handleRead (const boost::system::error_code& e,
00045                      std::size_t bytes_transferred);
00046
00048     void handleWrite (const boost::system::error_code& e);
00049
00052     boost::asio::io_service::strand _strand;
00053
00055     boost::asio::ip::tcp::socket _socket;
00056
00058     RequestHandler& _requestHandler;
00059
00061     boost::array<char, 8192> _buffer;
00062
00064     Request _request;
00065
00067     Reply _reply;
00068   };
00069
00071   typedef boost::shared_ptr<Connection> ConnectionShrPtr_T;
```

```
00072
00073 }
00074 #endif // __AIRINV_SVR_CONNECTION_HPP
```

23.217 airinv/server/header.hpp File Reference

```
#include <string>
```

Classes

- struct AIRINV::header

Namespaces

- namespace AIRINV

23.218 header.hpp

```
00001 #ifndef __AIRINV_SVR_HEADER_HPP
00002 #define __AIRINV_SVR_HEADER_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009
00010 namespace AIRINV {
00011
00013   struct header {
00014     std::string name;
00015     std::string value;
00016   };
00017
00018 }
00019 #endif // __AIRINV_SVR_HEADER_HPP
```

23.219 airinv/server/posix_main.cpp File Reference

```
#include <iostream>
#include <string>
#include <boost/asio.hpp>
#include <boost/thread.hpp>
#include <boost/bind.hpp>
#include <boost/lexical_cast.hpp>
#include <airinv/server/AirInvServer.hpp>
#include <pthread.h>
#include <signal.h>
```

Functions

- int main (int argc, char ∗argv[])

23.219.1 Function Documentation

23.219.1.1 int main (int *argc*, char ∗ *argv[]*)

Definition at line 25 of file posix_main.cpp.

References AIRINV::AirInvServer::run().

23.220 posix_main.cpp

```
00001 //
00002 // posix_main.cpp
00003 // ~~~~~~~~~~~~~~
00004 //
00005 // Copyright (c) 2003-2008 Christopher M. Kohlhoff (chris at kohlhoff dot com)
00006 //
00007 // Distributed under the Boost Software License, Version 1.0. (See accompanying
00008 // file LICENSE_1_0.txt or copy at http://www.boost.org/LICENSE_1_0.txt)
00009 //
00010
00011 #include <iostream>
00012 #include <string>
00013 #include <boost/asio.hpp>
00014 #include <boost/thread.hpp>
00015 #include <boost/bind.hpp>
00016 #include <boost/lexical_cast.hpp>
00017 #include <airinv/server/AirInvServer.hpp>
00018
00019 #if !defined(_WIN32)
00020
00021 #include <pthread.h>
00022 #include <signal.h>
00023
00024 // ////////////////// M A I N ////////////////////
00025 int main(int argc, char* argv[]) {
00026
00027   try {
00028
00029     // Check command line arguments.
00030     if (argc != 5) {
00031       std::cerr << "Usage: airinvServer <address> <port> <threads> <doc_root>"
00032                 << std::endl;
00033       std::cerr << "  For IPv4, try:" << std::endl;
00034       std::cerr << "    receiver 0.0.0.0 80 1 ." << std::endl;
00035       std::cerr << "  For IPv6, try:" << std::endl;
00036       std::cerr << "    receiver 0::0 80 1 ." << std::endl;
00037       return 1;
00038     }
00039
00040     // Block all signals for background thread.
00041     sigset_t new_mask;
00042     sigfillset (&new_mask);
00043     sigset_t old_mask;
00044     pthread_sigmask (SIG_BLOCK, &new_mask, &old_mask);
00045
00046     // Run server in background thread.
00047     std::size_t num_threads = boost::lexical_cast<std::size_t>(argv[3]);
00048     AIRINV::AirInvServer s (argv[1], argv[2], argv[4],
      num_threads);
00049     boost::thread t (boost::bind (&AIRINV::AirInvServer::run
      , &s));
00050
00051     // Restore previous signals.
00052     pthread_sigmask (SIG_SETMASK, &old_mask, 0);
00053
00054     // Wait for signal indicating time to shut down.
00055     sigset_t wait_mask;
00056     sigemptyset (&wait_mask);
00057     sigaddset (&wait_mask, SIGINT);
00058     sigaddset (&wait_mask, SIGQUIT);
00059     sigaddset (&wait_mask, SIGTERM);
00060     pthread_sigmask (SIG_BLOCK, &wait_mask, 0);
00061     int sig = 0;
00062     sigwait (&wait_mask, &sig);
00063
00064     // Stop the server.
00065     s.stop();
00066     t.join();
00067
00068   } catch (std::exception& e) {
00069     std::cerr << "exception: " << e.what() << "\n";
00070   }
00071
00072   return 0;
00073 }
00074
00075 #endif // !defined(_WIN32)
```

23.221 airinv/server/Reply.cpp File Reference

```
#include <cassert>
#include <string>
#include <boost/lexical_cast.hpp>
#include <airinv/server/Reply.hpp>
```

Namespaces

- namespace AIRINV

23.222 Reply.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <string>
00007 // Boost
00008 #include <boost/lexical_cast.hpp>
00009 // AirInv
00010 #include <airinv/server/Reply.hpp>
00011
00012 namespace AIRINV {
00013
00014   // ////////////////////////////////////////////////////////////////////
00015   std::vector<boost::asio::const_buffer> Reply::to_buffers() {
00016     std::vector<boost::asio::const_buffer> lBuffers;
00017     lBuffers.push_back (boost::asio::buffer(content));
00018     return lBuffers;
00019   }
00020
00021 }
```

23.223 airinv/server/Reply.hpp File Reference

```
#include <string>
#include <vector>
#include <boost/asio.hpp>
#include <airinv/FlightRequestStatus.hpp>
```

Classes

- struct AIRINV::Reply

Namespaces

- namespace AIRINV

23.224 Reply.hpp

```
00001 #ifndef __AIRINV_SVR_REPLY_HPP
00002 #define __AIRINV_SVR_REPLY_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010 // Boost
```

```
00011 #include <boost/asio.hpp>
00012 // AirInv
00013 #include <airinv/FlightRequestStatus.hpp>
00014
00015 namespace AIRINV {
00016
00018   struct Reply {
00020     FlightRequestStatus::EN_FlightRequestStatus
      _status;
00021
00023     std::string content;
00024
00029     std::vector<boost::asio::const_buffer> to_buffers();
00030   };
00031
00032 }
00033 #endif // __AIRINV_SVR_REPLY_HPP
```

23.225 airinv/server/Request.cpp File Reference

```
#include <cassert>
#include <airinv/server/Request.hpp>
```

Namespaces

- namespace AIRINV

23.226 Request.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // AirInv
00007 #include <airinv/server/Request.hpp>
00008
00009 namespace AIRINV {
00010
00011   // ////////////////////////////////////////////////////////////////////
00012   bool Request::parseFlightDate () {
00013     bool hasBeenSuccessfull = false;
00014
00015     //
00016     _airlineCode = "BA";
00017     _flightNumber = 341;
00018     _departureDate = stdair::Date_T (2010, 04, 20);
00019
00020     //
00021     hasBeenSuccessfull = true;
00022
00023     return hasBeenSuccessfull;
00024   }
00025
00026 }
```

23.227 airinv/server/Request.hpp File Reference

```
#include <string>
#include <vector>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
```

Classes

- struct AIRINV::Request

Namespaces

- namespace AIRINV

23.228 Request.hpp

```
00001 #ifndef __AIRINV_SVR_REQUEST_HPP
00002 #define __AIRINV_SVR_REQUEST_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010 // StdAir
00011 #include <stdair/stdair_basic_types.hpp>
00012 #include <stdair/stdair_date_time_types.hpp>
00013 // AirInv
00014
00015 namespace AIRINV {
00016
00018   struct Request {
00019   public:
00024     bool parseFlightDate();
00025
00026   public:
00027     // ////////// Attributes ///////////////
00029     std::string _flightDetails;
00031     stdair::AirlineCode_T _airlineCode;
00033     stdair::FlightNumber_T _flightNumber;
00035     stdair::Date_T _departureDate;
00036   };
00037
00038 }
00039 #endif // __AIRINV_SVR_REQUEST_HPP
```

23.229 airinv/server/RequestHandler.cpp File Reference

```
#include <cassert>
#include <string>
#include <fstream>
#include <sstream>
#include <boost/lexical_cast.hpp>
#include <airinv/server/Reply.hpp>
#include <airinv/server/Request.hpp>
#include <airinv/server/RequestHandler.hpp>
```

Namespaces

- namespace AIRINV

23.230 RequestHandler.cpp

```
00001 // //////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <string>
00007 #include <fstream>
00008 #include <sstream>
00009 // Boost
00010 #include <boost/lexical_cast.hpp>
00011 // StdAir
00012 // AirInv
00013 #include <airinv/server/Reply.hpp>
00014 #include <airinv/server/Request.hpp>
00015 #include <airinv/server/RequestHandler.hpp>
```

```
00016
00017 namespace AIRINV {
00018
00019   // ////////////////////////////////////////////////////////////////////
00020   RequestHandler::RequestHandler (const
      stdair::AirlineCode_T& iAirlineCode)
00021     : _airlineCode (iAirlineCode) {
00022   }
00023
00024   // ////////////////////////////////////////////////////////////////////
00025   bool RequestHandler::
00026   handleRequest (Request& ioRequest, Reply& ioReply)
      const {
00027     bool hasBeenSuccessfull = false;
00028
00029     // Decode request string to a flight-date details (airline code,
00030     // flight number and departure date)
00031     hasBeenSuccessfull = ioRequest.parseFlightDate();
00032
00033     if (hasBeenSuccessfull == false) {
00034       ioReply._status = FlightRequestStatus::INTERNAL_ERROR
    ;
00035       return hasBeenSuccessfull;
00036     }
00037
00045     // Fill out the reply to be sent to the client.
00046     ioReply._status = FlightRequestStatus::OK;
00047     ioReply.content = "Your are looking for: '" + ioRequest.
      _flightDetails + "'. Ok, I have found your flight-date. Be
      patient until I give you more information about it";
00048
00049     return hasBeenSuccessfull;
00050   }
00051
00052 }
```

23.231 airinv/server/RequestHandler.hpp File Reference

```
#include <string>
#include <boost/noncopyable.hpp>
#include <stdair/stdair_basic_types.hpp>
```

Classes

- class AIRINV::RequestHandler

  The common handler for all incoming requests.

Namespaces

- namespace stdair

  Forward declarations.
- namespace AIRINV

23.232 RequestHandler.hpp

```
00001 #ifndef __AIRINV_SVR_REQUESTHANDLER_HPP
00002 #define __AIRINV_SVR_REQUESTHANDLER_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // Boost
00010 #include <boost/noncopyable.hpp>
00011 // StdAir
00012 #include <stdair/stdair_basic_types.hpp>
00013 // AirInv
00014
00015 // Forward declarations
```

```
00016 namespace stdair {
00017   struct InventoryKey_T;
00018   struct FlightDateKey_T;
00019 }
00020
00021 namespace AIRINV {
00022
00023   // Forward declarations.
00024   struct Reply;
00025   struct Request;
00026
00028   class RequestHandler : private boost::noncopyable {
00029   public:
00030     // ////////////// Constructors and Destructors /////////////////
00034     RequestHandler (const stdair::AirlineCode_T&);
00035
00036
00037   public:
00038     // /////////// Business Support Methods ////////////////
00040     bool handleRequest (Request&, Reply&) const;
00041
00042
00043   private:
00044     // ///////////////// Attributes ////////////////////
00046     stdair::AirlineCode_T _airlineCode;
00047   };
00048
00049 }
00050 #endif // __AIRINV_SVR_REQUESTHANDLER_HPP
```

23.233 airinv/server/RequestParser.cpp File Reference

```
#include <cassert>
#include <airinv/server/RequestParser.hpp>
#include <airinv/server/Request.hpp>
```

Namespaces

- namespace AIRINV

23.234 RequestParser.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // AirInv
00007 #include <airinv/server/RequestParser.hpp>
00008 #include <airinv/server/Request.hpp>
00009
00010 namespace AIRINV {
00011
00012   // ////////////////////////////////////////////////////////////////////
00013   RequestParser::RequestParser()
00014     : state_(method_start) {
00015   }
00016
00017   // ////////////////////////////////////////////////////////////////////
00018   void RequestParser::reset() {
00019     state_ = method_start;
00020   }
00021
00022   // ////////////////////////////////////////////////////////////////////
00023   boost::tribool RequestParser::consume (Request& req, char input) {
00024
00025     switch (state_) {
00026
00027     case method_start:
00028       if (!is_char(input) || is_ctl(input) || is_tspecial(input)) {
00029         return false;
00030
00031       } else {
00032         state_ = method;
00033         req.method.push_back(input);
```

```
00034         return boost::indeterminate;
00035       }
00036
00037     case method:
00038       if (input == ' ') {
00039         state_ = uri;
00040         return boost::indeterminate;
00041
00042       } else if (!is_char(input) || is_ctl(input) || is_tspecial(input)) {
00043         return false;
00044
00045       } else {
00046         req.method.push_back(input);
00047         return boost::indeterminate;
00048       }
00049
00050     case uri_start:
00051       if (is_ctl(input)) {
00052         return false;
00053
00054       } else {
00055         state_ = uri;
00056         req.uri.push_back(input);
00057         return boost::indeterminate;
00058       }
00059
00060     case uri:
00061       if (input == ' ') {
00062         state_ = http_version_h;
00063         return boost::indeterminate;
00064
00065       } else if (is_ctl(input)) {
00066         return false;
00067
00068       } else {
00069         req.uri.push_back(input);
00070         return boost::indeterminate;
00071       }
00072
00073     case http_version_h:
00074       if (input == 'H') {
00075         state_ = http_version_t_1;
00076         return boost::indeterminate;
00077
00078       } else {
00079         return false;
00080       }
00081
00082     case http_version_t_1:
00083       if (input == 'T') {
00084         state_ = http_version_t_2;
00085         return boost::indeterminate;
00086
00087       } else {
00088         return false;
00089       }
00090
00091     case http_version_t_2:
00092       if (input == 'T') {
00093         state_ = http_version_p;
00094         return boost::indeterminate;
00095
00096       } else {
00097         return false;
00098       }
00099
00100     case http_version_p:
00101       if (input == 'P') {
00102         state_ = http_version_slash;
00103         return boost::indeterminate;
00104
00105       } else {
00106         return false;
00107       }
00108
00109     case http_version_slash:
00110       if (input == '/') {
00111         req.http_version_major = 0;
00112         req.http_version_minor = 0;
00113         state_ = http_version_major_start;
00114         return boost::indeterminate;
00115
00116       } else {
00117         return false;
00118       }
00119
00120     case http_version_major_start:
```

```
00121       if (is_digit(input)) {
00122         req.http_version_major = req.http_version_major * 10 + input - '0';
00123         state_ = http_version_major;
00124         return boost::indeterminate;
00125
00126       } else {
00127         return false;
00128       }
00129
00130     case http_version_major:
00131       if (input == '.') {
00132         state_ = http_version_minor_start;
00133         return boost::indeterminate;
00134
00135       } else if (is_digit(input)) {
00136         req.http_version_major = req.http_version_major * 10 + input - '0';
00137         return boost::indeterminate;
00138
00139       } else {
00140         return false;
00141       }
00142
00143     case http_version_minor_start:
00144       if (is_digit(input)) {
00145         req.http_version_minor = req.http_version_minor * 10 + input - '0';
00146         state_ = http_version_minor;
00147         return boost::indeterminate;
00148
00149       } else {
00150         return false;
00151       }
00152
00153     case http_version_minor:
00154       if (input == '\r') {
00155         state_ = expecting_newline_1;
00156         return boost::indeterminate;
00157
00158       } else if (is_digit(input)) {
00159         req.http_version_minor = req.http_version_minor * 10 + input - '0';
00160         return boost::indeterminate;
00161
00162       } else {
00163         return false;
00164       }
00165
00166     case expecting_newline_1:
00167       if (input == '\n') {
00168         state_ = header_line_start;
00169         return boost::indeterminate;
00170
00171       } else {
00172         return false;
00173       }
00174
00175     case header_line_start:
00176       if (input == '\r') {
00177         state_ = expecting_newline_3;
00178         return boost::indeterminate;
00179
00180       } else if (!is_char(input) || is_ctl(input) || is_tspecial(input)) {
00181         return false;
00182
00183       } else {
00184         state_ = header_name;
00185         return boost::indeterminate;
00186       }
00187
00188     case header_lws:
00189       if (input == '\r') {
00190         state_ = expecting_newline_2;
00191         return boost::indeterminate;
00192
00193       } else if (input == ' ' || input == '\t') {
00194         return boost::indeterminate;
00195
00196       } else if (is_ctl(input)) {
00197         return false;
00198
00199       } else {
00200         state_ = header_value;
00201         return boost::indeterminate;
00202       }
00203
00204     case header_name:
00205       if (input == ':') {
00206         state_ = space_before_header_value;
00207         return boost::indeterminate;
```

```
00208
00209       } else if (!is_char(input) || is_ctl(input) || is_tspecial(input)) {
00210         return false;
00211
00212       } else {
00213         return boost::indeterminate;
00214       }
00215
00216     case space_before_header_value:
00217       if (input == ' ') {
00218         state_ = header_value;
00219         return boost::indeterminate;
00220
00221       } else {
00222         return false;
00223       }
00224
00225     case header_value:
00226       if (input == '\r') {
00227         state_ = expecting_newline_2;
00228         return boost::indeterminate;
00229
00230       } else if (is_ctl(input)) {
00231         return false;
00232
00233       } else {
00234         return boost::indeterminate;
00235       }
00236
00237     case expecting_newline_2:
00238       if (input == '\n') {
00239         state_ = header_line_start;
00240         return boost::indeterminate;
00241
00242       } else {
00243         return false;
00244       }
00245
00246     case expecting_newline_3:
00247       return (input == '\n');
00248
00249     default:
00250       return false;
00251     }
00252   }
00253
00254   // ////////////////////////////////////////////////////////////////////////
00255   bool RequestParser::is_char(int c) {
00256     return c >= 0 && c <= 127;
00257   }
00258
00259   // ////////////////////////////////////////////////////////////////////////
00260   bool RequestParser::is_ctl(int c) {
00261     return (c >= 0 && c <= 31) || (c == 127);
00262   }
00263
00264   // ////////////////////////////////////////////////////////////////////////
00265   bool RequestParser::is_tspecial(int c) {
00266     switch (c) {
00267     case '(': case ')': case '<': case '>': case '@':
00268     case ',': case ';': case ':': case '\\': case '"':
00269     case '/': case '[': case ']': case '?': case '=':
00270     case '{': case '}': case ' ': case '\t':
00271       return true;
00272     default:
00273       return false;
00274     }
00275   }
00276
00277   // ////////////////////////////////////////////////////////////////////////
00278   bool RequestParser::is_digit(int c) {
00279     return c >= '0' && c <= '9';
00280   }
00281
00282 }
```

23.235 airinv/server/RequestParser.hpp File Reference

```
#include <boost/logic/tribool.hpp>
#include <boost/tuple/tuple.hpp>
```

Classes

- class AIRINV::RequestParser

  Parser for incoming requests.

Namespaces

- namespace AIRINV

23.236 RequestParser.hpp

```
00001 #ifndef __AIRINV_SVR_REQUESTPARSER_HPP
00002 #define __AIRINV_SVR_REQUESTPARSER_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////////
00007 // STL
00008 // Boost
00009 #include <boost/logic/tribool.hpp>
00010 #include <boost/tuple/tuple.hpp>
00011
00012 namespace AIRINV {
00013
00014   struct Request;
00015
00017   class RequestParser {
00018   public:
00020     RequestParser();
00021
00023     void reset();
00024
00029     template <typename InputIterator>
00030     boost::tuple<boost::tribool, InputIterator> parse (Request& req
      ,
00031                                                        InputIterator begin,
00032                                                        InputIterator end) {
00033
00034       while (begin != end) {
00035         boost::tribool result = consume(req, *begin++);
00036         if (result || !result)
00037           return boost::make_tuple(result, begin);
00038       }
00039
00040       boost::tribool result = boost::indeterminate;
00041       return boost::make_tuple(result, begin);
00042     }
00043
00044   private:
00046     boost::tribool consume (Request& req, char input);
00047
00049     static bool is_char(int c);
00050
00052     static bool is_ctl(int c);
00053
00055     static bool is_tspecial(int c);
00056
00058     static bool is_digit(int c);
00059
00061     enum state {
00062       method_start,
00063       method,
00064       uri_start,
00065       uri,
00066       http_version_h,
00067       http_version_t_1,
00068       http_version_t_2,
00069       http_version_p,
00070       http_version_slash,
00071       http_version_major_start,
00072       http_version_major,
00073       http_version_minor_start,
00074       http_version_minor,
00075       expecting_newline_1,
00076       header_line_start,
00077       header_lws,
00078       header_name,
00079       space_before_header_value,
00080       header_value,
00081       expecting_newline_2,
```

```
00082       expecting_newline_3
00083     } state_;
00084   };
00085
00086 }
00087 #endif // __AIRINV_SVR_REQUESTPARSER_HPP
```

23.237 airinv/server/win_main.cpp File Reference

```
#include <iostream>
#include <string>
#include <boost/asio.hpp>
#include <boost/bind.hpp>
#include <boost/function.hpp>
#include <boost/lexical_cast.hpp>
#include <airinv/server/AirInvServer.hpp>
```

23.238 win_main.cpp

```
00001 //
00002 // win_main.cpp
00003 // ~~~~~~~~~~~~
00004 //
00005 // Copyright (c) 2003-2008 Christopher M. Kohlhoff (chris at kohlhoff dot com)
00006 //
00007 // Distributed under the Boost Software License, Version 1.0. (See accompanying
00008 // file LICENSE_1_0.txt or copy at http://www.boost.org/LICENSE_1_0.txt)
00009 //
00010
00011 #include <iostream>
00012 #include <string>
00013 #include <boost/asio.hpp>
00014 #include <boost/bind.hpp>
00015 #include <boost/function.hpp>
00016 #include <boost/lexical_cast.hpp>
00017 #include <airinv/server/AirInvServer.hpp>
00018
00019 #if defined(_WIN32)
00020
00021 boost::function0<void> console_ctrl_function;
00022
00023 BOOL WINAPI console_ctrl_handler(DWORD ctrl_type) {
00024   switch (ctrl_type) {
00025   case CTRL_C_EVENT:
00026   case CTRL_BREAK_EVENT:
00027   case CTRL_CLOSE_EVENT:
00028   case CTRL_SHUTDOWN_EVENT:
00029     console_ctrl_function();
00030     return TRUE;
00031   default:
00032     return FALSE;
00033   }
00034 }
00035
00036 int main(int argc, char* argv[]) {
00037
00038   try {
00039
00040     // Check command line arguments.
00041     if (argc != 5) {
00042       std::cerr << "Usage: http_server <address> <port> <threads> <doc_root>\n"
     ;
00043       std::cerr << "  For IPv4, try:\n";
00044       std::cerr << "    http_server 0.0.0.0 80 1 .\n";
00045       std::cerr << "  For IPv6, try:\n";
00046       std::cerr << "    http_server 0::0 80 1 .\n";
00047       return 1;
00048     }
00049
00050     // Initialise server.
00051     std::size_t num_threads = boost::lexical_cast<std::size_t>(argv[3]);
00052     AIRINV::AirInvServer s (argv[1], argv[2], argv[4],
    num_threads);
00053
00054     // Set console control handler to allow server to be stopped.
00055     console_ctrl_function = boost::bind(&AIRINV::AirInvServer::stop
```

```
    , &s);
00056     SetConsoleCtrlHandler(console_ctrl_handler, TRUE);
00057
00058     // Run the server until stopped.
00059     s.run();
00060
00061   } catch (std::exception& e) {
00062     std::cerr << "exception: " << e.what() << "\n";
00063   }
00064
00065   return 0;
00066 }
00067 #endif // defined(_WIN32)
```

23.239 airinv/service/AIRINV_Master_Service.cpp File Reference

```
#include <cassert>
#include <cmath>
#include <boost/make_shared.hpp>
#include <stdair/stdair_json.hpp>
#include <stdair/basic/BasChronometer.hpp>
#include <stdair/basic/EventType.hpp>
#include <stdair/bom/BomKeyManager.hpp>
#include <stdair/bom/SnapshotStruct.hpp>
#include <stdair/bom/RMEventStruct.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/STDAIR_Service.hpp>
#include <sevmgr/SEVMGR_Service.hpp>
#include <airinv/basic/BasConst_AIRINV_Service.hpp>
#include <airinv/factory/FacAirinvMasterServiceContext.hpp>
#include <airinv/command/InventoryParser.hpp>
#include <airinv/command/InventoryManager.hpp>
#include <airinv/service/AIRINV_Master_ServiceContext.hpp>
#include <airinv/AIRINV_Service.hpp>
#include <airinv/AIRINV_Master_Service.hpp>
```

Namespaces

- namespace AIRINV

23.240 AIRINV_Master_Service.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <cmath>
00007 // Boost
00008 #include <boost/make_shared.hpp>
00009 // StdAir
00010 #include <stdair/stdair_json.hpp>
00011 #include <stdair/basic/BasChronometer.hpp>
00012 #include <stdair/basic/EventType.hpp>
00013 #include <stdair/bom/BomKeyManager.hpp>
00014 #include <stdair/bom/SnapshotStruct.hpp>
00015 #include <stdair/bom/RMEventStruct.hpp>
00016 #include <stdair/service/Logger.hpp>
00017 #include <stdair/STDAIR_Service.hpp>
00018 // SEvMgr
00019 #include <sevmgr/SEVMGR_Service.hpp>
00020 // AirInv
00021 #include <airinv/basic/BasConst_AIRINV_Service.hpp
      >
00022 #include <airinv/factory/FacAirinvMasterServiceContext.hpp
      >
00023 #include <airinv/command/InventoryParser.hpp>
```

```
00024 #include <airinv/command/InventoryManager.hpp
      >
00025 #include <airinv/service/AIRINV_Master_ServiceContext.hpp
      >
00026 #include <airinv/AIRINV_Service.hpp>
00027 #include <airinv/AIRINV_Master_Service.hpp>
00028
00029 namespace AIRINV {
00030
00031   // ////////////////////////////////////////////////////////////////////
00032   AIRINV_Master_Service::AIRINV_Master_Service()
00033     : _airinvMasterServiceContext (NULL) {
00034     assert (false);
00035   }
00036
00037   // ////////////////////////////////////////////////////////////////////
00038   AIRINV_Master_Service::
00039   AIRINV_Master_Service (const AIRINV_Master_Service& iService)
00040     : _airinvMasterServiceContext (NULL) {
00041     assert (false);
00042   }
00043
00044   // ////////////////////////////////////////////////////////////////////
00045   AIRINV_Master_Service::
00046   AIRINV_Master_Service (const stdair::BasLogParams& iLogParams,
00047                          const stdair::BasDBParams& iDBParams)
00048     : _airinvMasterServiceContext (NULL) {
00049
00050     // Initialise the STDAIR service handler
00051     stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00052       initStdAirService (iLogParams, iDBParams);
00053
00054     // Initialise the service context
00055     initServiceContext();
00056
00057     // Add the StdAir service context to the AIRINV service context
00058     // \note RMOL owns the STDAIR service resources here.
00059     const bool ownStdairService = true;
00060     addStdAirService (lSTDAIR_Service_ptr, ownStdairService);
00061
00062     // Initialise the (remaining of the) context
00063     initSlaveAirinvService();
00064   }
00065
00066   // ////////////////////////////////////////////////////////////////////
00067   AIRINV_Master_Service::
00068   AIRINV_Master_Service (const stdair::BasLogParams& iLogParams)
00069     : _airinvMasterServiceContext (NULL) {
00070
00071     // Initialise the STDAIR service handler
00072     stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00073       initStdAirService (iLogParams);
00074
00075     // Initialise the service context
00076     initServiceContext();
00077
00078     // Add the StdAir service context to the AIRINV service context
00079     // \note RMOL owns the STDAIR service resources here.
00080     const bool ownStdairService = true;
00081     addStdAirService (lSTDAIR_Service_ptr, ownStdairService);
00082
00083     // Initialise the (remaining of the) context
00084     initSlaveAirinvService();
00085   }
00086
00087   // ////////////////////////////////////////////////////////////////////
00088   AIRINV_Master_Service::
00089   AIRINV_Master_Service (stdair::STDAIR_ServicePtr_T ioSTDAIR_Service_ptr)
00090     : _airinvMasterServiceContext (NULL) {
00091
00092     // Initialise the service context
00093     initServiceContext();
00094
00095     // Store the STDAIR service object within the (AIRINV) service context
00096     // \note AirInv does not own the STDAIR service resources here.
00097     const bool doesNotOwnStdairService = false;
00098     addStdAirService (ioSTDAIR_Service_ptr, doesNotOwnStdairService);
00099
00100     // Initialise the (remaining of the) context
00101     initSlaveAirinvService();
00102   }
00103
00104   // ////////////////////////////////////////////////////////////////////
00105   AIRINV_Master_Service::
00106   AIRINV_Master_Service (stdair::STDAIR_ServicePtr_T ioSTDAIR_Service_ptr,
00107                          SEVMGR::SEVMGR_ServicePtr_T ioSEVMGR_Service_ptr)
00108     : _airinvMasterServiceContext (NULL) {
```

```
00109
00110     // Initialise the service context
00111     initServiceContext();
00112
00113     // Store the STDAIR service object within the (AIRINV) service context
00114     // \note AirInv does not own the STDAIR service resources here.
00115     const bool doesNotOwnStdairService = false;
00116     addStdAirService (ioSTDAIR_Service_ptr, doesNotOwnStdairService);
00117
00118     //Add the SEvMgr service to the TRADEMGEN service context.
00119     const bool doesNotOwnSEVMGRService = false;
00120     addSEVMGRService (ioSEVMGR_Service_ptr, doesNotOwnSEVMGRService);
00121
00122     // Initialise the (remaining of the) context
00123     initSlaveAirinvService();
00124   }
00125
00126   // ////////////////////////////////////////////////////////////////////
00127   AIRINV_Master_Service::~AIRINV_Master_Service
      () {
00128     // Delete/Clean all the objects from memory
00129     finalise();
00130   }
00131
00132   // ////////////////////////////////////////////////////////////////////
00133   void AIRINV_Master_Service::finalise() {
00134     assert (_airinvMasterServiceContext != NULL);
00135     // Reset the (Boost.)Smart pointer pointing on the STDAIR_Service object.
00136     _airinvMasterServiceContext->reset();
00137   }
00138
00139   // ////////////////////////////////////////////////////////////////////
00140   void AIRINV_Master_Service::initServiceContext() {
00141     // Initialise the context
00142     AIRINV_Master_ServiceContext& lAIRINV_Master_ServiceContext =
00143       FacAirinvMasterServiceContext::instance
      ().create();
00144     _airinvMasterServiceContext = &lAIRINV_Master_ServiceContext;
00145   }
00146
00147   // ////////////////////////////////////////////////////////////////////
00148   void AIRINV_Master_Service::
00149   addStdAirService (stdair::STDAIR_ServicePtr_T ioSTDAIR_Service_ptr,
00150                     const bool iOwnStdairService) {
00151
00152     // Retrieve the AirInv Master service context
00153     assert (_airinvMasterServiceContext != NULL);
00154     AIRINV_Master_ServiceContext& lAIRINV_Master_ServiceContext =
00155       *_airinvMasterServiceContext;
00156
00157     // Store the STDAIR service object within the (AIRINV) service context
00158     lAIRINV_Master_ServiceContext.setSTDAIR_Service (ioSTDAIR_Service_ptr,
00159                                                     iOwnStdairService);
00160   }
00161
00162 // ////////////////////////////////////////////////////////////////////
00163   void AIRINV_Master_Service::
00164   addSEVMGRService (SEVMGR::SEVMGR_ServicePtr_T ioSEVMGR_Service_ptr,
00165                     const bool iOwnSEVMGRService)  {
00166
00167     // Retrieve the AirInv Master service context
00168     assert (_airinvMasterServiceContext != NULL);
00169     AIRINV_Master_ServiceContext& lAIRINV_Master_ServiceContext =
00170       *_airinvMasterServiceContext;
00171
00172     // Store the STDAIR service object within the (TRADEMGEN) service context
00173     lAIRINV_Master_ServiceContext.setSEVMGR_Service (ioSEVMGR_Service_ptr,
00174                                                     iOwnSEVMGRService);
00175   }
00176
00177   // ////////////////////////////////////////////////////////////////////
00178   stdair::STDAIR_ServicePtr_T AIRINV_Master_Service::
00179   initStdAirService (const stdair::BasLogParams& iLogParams,
00180                      const stdair::BasDBParams& iDBParams) {
00181
00189     stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00190       boost::make_shared<stdair::STDAIR_Service> (iLogParams, iDBParams);
00191
00192     return lSTDAIR_Service_ptr;
00193   }
00194
00195   // ////////////////////////////////////////////////////////////////////
00196   stdair::STDAIR_ServicePtr_T AIRINV_Master_Service::
00197   initStdAirService (const stdair::BasLogParams& iLogParams) {
00198
00206     stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00207       boost::make_shared<stdair::STDAIR_Service> (iLogParams);
```

```
00208
00209     return lSTDAIR_Service_ptr;
00210   }
00211
00212   // ////////////////////////////////////////////////////////////////////
00213   void AIRINV_Master_Service::initSlaveAirinvService() {
00214
00215     // Retrieve the AirInv Master service context
00216     assert (_airinvMasterServiceContext != NULL);
00217     AIRINV_Master_ServiceContext& lAIRINV_Master_ServiceContext =
00218       *_airinvMasterServiceContext;
00219
00220     // Retrieve the StdAir service
00221     stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00222       lAIRINV_Master_ServiceContext.getSTDAIR_ServicePtr();
00223     assert (lSTDAIR_Service_ptr != NULL);
00224
00234     AIRINV_ServicePtr_T lAIRINV_Service_ptr;
00235     const bool ownSEVMGRService =
00236       lAIRINV_Master_ServiceContext.getOwnSEVMGRServiceFlag();
00237     if (ownSEVMGRService == false) {
00238       // Retrieve the SEVMGR service
00239       SEVMGR::SEVMGR_ServicePtr_T lSEVMGR_Service_ptr =
00240         lAIRINV_Master_ServiceContext.getSEVMGR_ServicePtr();
00241       assert (lSEVMGR_Service_ptr != NULL);
00242       lAIRINV_Service_ptr = boost::make_shared<AIRINV_Service> (
      lSTDAIR_Service_ptr,
00243
      lSEVMGR_Service_ptr);
00244     } else {
00245       lAIRINV_Service_ptr = boost::make_shared<AIRINV_Service> (
      lSTDAIR_Service_ptr);
00246     }
00247     assert (lAIRINV_Service_ptr != NULL);
00248
00249     // Store the AIRINV service object within the AIRINV Master service
       context.
00250     lAIRINV_Master_ServiceContext.setAIRINV_Service (lAIRINV_Service_ptr);
00251   }
00252
00253   // ////////////////////////////////////////////////////////////////////
00254   void AIRINV_Master_Service::
00255   parseAndLoad (const InventoryFilePath&
      iInventoryInputFilename) {
00256
00257     // Retrieve the AirInv Master service context
00258     if (_airinvMasterServiceContext == NULL) {
00259       throw stdair::NonInitialisedServiceException ("The AirInvMaster service "
00260                                                     "has not been initialised")
      ;
00261     }
00262     assert (_airinvMasterServiceContext != NULL);
00263
00264     // Retrieve the AirInv service context and whether it owns the Stdair
00265     // service
00266     AIRINV_Master_ServiceContext&
      lAIRINV_Master_ServiceContext =
00267       *_airinvMasterServiceContext;
00268     const bool doesOwnStdairService =
00269       lAIRINV_Master_ServiceContext.getOwnStdairServiceFlag();
00270
00271     // Retrieve the slave AIRINV service object from the (AIRINV)
00272     // service context
00273     AIRINV_Service& lAIRINV_Service =
00274       lAIRINV_Master_ServiceContext.getAIRINV_Service();
00275
00276     // Delegate the file parsing and BOM building to the dedicated service
00277     lAIRINV_Service.parseAndLoad (iInventoryInputFilename);
00278
00283     if (doesOwnStdairService == true) {
00284
00285       //
00286       clonePersistentBom ();
00287     }
00288   }
00289
00290   // ////////////////////////////////////////////////////////////////////
00291   void AIRINV_Master_Service::buildSampleBom
      () {
00292
00293     // Retrieve the AirInv Master service context
00294     if (_airinvMasterServiceContext == NULL) {
00295       throw stdair::NonInitialisedServiceException ("The AirInvMaster service "
00296                                                     "has not been initialised")
      ;
00297     }
00298     assert (_airinvMasterServiceContext != NULL);
```

```
00299
00300     // Retrieve the AirInv service context and whether it owns the Stdair
00301     // service
00302     AIRINV_Master_ServiceContext&
      lAIRINV_Master_ServiceContext =
00303       *_airinvMasterServiceContext;
00304     const bool doesOwnStdairService =
00305       lAIRINV_Master_ServiceContext.getOwnStdairServiceFlag();
00306
00307     // Retrieve the StdAir service object from the (AirInv) service context
00308     stdair::STDAIR_Service& lSTDAIR_Service =
00309       lAIRINV_Master_ServiceContext.getSTDAIR_Service();
00310     stdair::BomRoot& lPersistentBomRoot =
00311       lSTDAIR_Service.getPersistentBomRoot();
00312
00317     if (doesOwnStdairService == true) {
00318       //
00319       lSTDAIR_Service.buildSampleBom();
00320     }
00321
00330     AIRINV_Service& lAIRINV_Service =
00331       lAIRINV_Master_ServiceContext.getAIRINV_Service();
00332     lAIRINV_Service.buildSampleBom();
00333
00338     buildComplementaryLinks (lPersistentBomRoot);
00339
00344     if (doesOwnStdairService == true) {
00345
00346       //
00347       clonePersistentBom ();
00348     }
00349   }
00350
00351   // ////////////////////////////////////////////////////////////////////
00352   void AIRINV_Master_Service::
00353   parseAndLoad (const stdair::ScheduleFilePath&
      iScheduleInputFilename,
00354                 const stdair::ODFilePath& iODInputFilename,
00355                 const stdair::FRAT5FilePath& iFRAT5InputFilename,
00356                 const stdair::FFDisutilityFilePath& iFFDisutilityInputFilename,
00357                 const AIRRAC::YieldFilePath& iYieldFilename) {
00358
00359     // Retrieve the AirInv Master service context
00360     if (_airinvMasterServiceContext == NULL) {
00361       throw stdair::NonInitialisedServiceException ("The AirInvMaster service "
00362                                                     "has not been initialised")
      ;
00363     }
00364     assert (_airinvMasterServiceContext != NULL);
00365
00366     // Retrieve the AirInv service context and whether it owns the Stdair
00367     // service
00368     AIRINV_Master_ServiceContext&
      lAIRINV_Master_ServiceContext =
00369       *_airinvMasterServiceContext;
00370     const bool doesOwnStdairService =
00371       lAIRINV_Master_ServiceContext.getOwnStdairServiceFlag();
00372
00373     // Retrieve the slave AirInv service object from the (AirInv)
00374     // service context
00375     AIRINV_Service& lAIRINV_Service =
00376       lAIRINV_Master_ServiceContext.getAIRINV_Service();
00377
00378     // Retrieve the StdAir service object from the (AirInv) service context
00379     stdair::STDAIR_Service& lSTDAIR_Service =
00380       lAIRINV_Master_ServiceContext.getSTDAIR_Service();
00381     stdair::BomRoot& lPersistentBomRoot =
00382       lSTDAIR_Service.getPersistentBomRoot();
00383
00387     lAIRINV_Service.parseAndLoad (iScheduleInputFilename,
      iODInputFilename,
00388                                   iFRAT5InputFilename,
00389                                   iFFDisutilityInputFilename, iYieldFilename);

00390
00401     buildComplementaryLinks (lPersistentBomRoot);
00402
00407     if (doesOwnStdairService == true) {
00408
00409       //
00410       clonePersistentBom ();
00411     }
00412   }
00413
00414   // ////////////////////////////////////////////////////////////////////
00415   void AIRINV_Master_Service::clonePersistentBom
       () {
```

```
00416
00417     // Retrieve the AirInv Master service context
00418     if (_airinvMasterServiceContext == NULL) {
00419       throw stdair::NonInitialisedServiceException ("The AirInvMaster service "
00420                                                     "has not been initialised")
      ;
00421     }
00422     assert (_airinvMasterServiceContext != NULL);
00423
00424     // Retrieve the AirInv service context and whether it owns the Stdair
00425     // service
00426     AIRINV_Master_ServiceContext&
      lAIRINV_Master_ServiceContext =
00427       *_airinvMasterServiceContext;
00428     const bool doesOwnStdairService =
00429       lAIRINV_Master_ServiceContext.getOwnStdairServiceFlag();
00430
00431     // Retrieve the slave AIRINV service object from the (AIRINV)
00432     // service context
00433     AIRINV_Service& lAIRINV_Service =
00434       lAIRINV_Master_ServiceContext.getAIRINV_Service();
00435
00436     // Retrieve the StdAir service object from the (AIRINV) service context
00437     stdair::STDAIR_Service& lSTDAIR_Service =
00438       lAIRINV_Master_ServiceContext.getSTDAIR_Service();
00439
00444     if (doesOwnStdairService == true) {
00445
00446       //
00447       lSTDAIR_Service.clonePersistentBom ();
00448     }
00449
00454     lAIRINV_Service.clonePersistentBom ();
00455
00460     stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
00461     buildComplementaryLinks (lBomRoot);
00462   }
00463
00464   // ////////////////////////////////////////////////////////////////////
00465   void AIRINV_Master_Service::
00466   buildComplementaryLinks (stdair::BomRoot& ioBomRoot)
       {
00467     // Currently, no more things to do by AIRINV_Master at that stage.
00468   }
00469
00470   // ////////////////////////////////////////////////////////////////////
00471   std::string AIRINV_Master_Service::
00472   jsonHandler (const stdair::JSONString& lJSONString) const {
00473
00474     // Retrieve the AirInv Master service context
00475     if (_airinvMasterServiceContext == NULL) {
00476       throw stdair::NonInitialisedServiceException ("The AirInvMaster service "
00477                                                     "has not been initialised")
      ;
00478     }
00479     assert (_airinvMasterServiceContext != NULL);
00480
00481     AIRINV_Master_ServiceContext&
      lAIRINV_Master_ServiceContext =
00482       *_airinvMasterServiceContext;
00483
00484     // Retrieve the slave AirInv (slave) service object from
00485     // the (AirInv master) service context
00486     AIRINV_Service& lAIRINV_Service =
00487       lAIRINV_Master_ServiceContext.getAIRINV_Service();
00488
00489     // Delegate the BOM dump to the dedicated service
00490     return lAIRINV_Service.jsonHandler (lJSONString);
00491
00492   }
00493
00494   // ////////////////////////////////////////////////////////////////////
00495   std::string AIRINV_Master_Service::
00496   jsonExportFlightDateList (const
      stdair::AirlineCode_T& iAirlineCode,
00497                             const stdair::FlightNumber_T& iFlightNumber) const
      {
00498
00499     // Retrieve the AirInv Master service context
00500     if (_airinvMasterServiceContext  == NULL) {
00501       throw stdair::NonInitialisedServiceException ("The AirInvMaster service "
00502                                                     "has not been initialised")
      ;
00503     }
00504     assert (_airinvMasterServiceContext != NULL);
00505
00506     AIRINV_Master_ServiceContext&
```

```
      lAIRINV_Master_ServiceContext =
00507       *_airinvMasterServiceContext;
00508
00509     // Retrieve the slave AirInv (slave) service object from
00510     // the (AirInv master) service context
00511     AIRINV_Service& lAIRINV_Service =
00512       lAIRINV_Master_ServiceContext.getAIRINV_Service();
00513
00514     // Delegate the JSON export to the dedicated service
00515     return lAIRINV_Service.jsonExportFlightDateList (
      iAirlineCode,
00516                                                      iFlightNumber);
00517   }
00518
00519   // ////////////////////////////////////////////////////////////////////
00520   std::string AIRINV_Master_Service::
00521   jsonExportFlightDateObjects (const
      stdair::AirlineCode_T& iAirlineCode,
00522                                const stdair::FlightNumber_T& iFlightNumber,
00523                                const stdair::Date_T& iDepartureDate) const {
00524
00525     // Retrieve the AirInv Master service context
00526     if (_airinvMasterServiceContext == NULL) {
00527       throw stdair::NonInitialisedServiceException ("The AirInvMaster service "
00528                                                     "has not been initialised")
      ;
00529     }
00530     assert (_airinvMasterServiceContext != NULL);
00531
00532     AIRINV_Master_ServiceContext&
      lAIRINV_Master_ServiceContext =
00533       *_airinvMasterServiceContext;
00534
00535     // Retrieve the slave AirInv (slave) service object from
00536     // the (AirInv master) service context
00537     AIRINV_Service& lAIRINV_Service =
00538       lAIRINV_Master_ServiceContext.getAIRINV_Service();
00539
00540     // Delegate the BOM dump to the dedicated service
00541     return lAIRINV_Service.jsonExportFlightDateObjects
       (iAirlineCode,
00542                                                        iFlightNumber,
00543                                                        iDepartureDate);
00544   }
00545
00546   // ////////////////////////////////////////////////////////////////////
00547   std::string AIRINV_Master_Service::
00548   list (const stdair::AirlineCode_T& iAirlineCode,
00549         const stdair::FlightNumber_T& iFlightNumber) const {
00550     std::ostringstream oFlightListStr;
00551
00552     // Retrieve the AirInv Master service context
00553     if (_airinvMasterServiceContext == NULL) {
00554       throw stdair::NonInitialisedServiceException ("The AirInvMaster service "
00555                                                     "has not been initialised")
      ;
00556     }
00557     assert (_airinvMasterServiceContext != NULL);
00558
00559     AIRINV_Master_ServiceContext&
      lAIRINV_Master_ServiceContext =
00560       *_airinvMasterServiceContext;
00561
00562     // Retrieve the slave AirInv (slave) service object from
00563     // the (AirInv master) service context
00564     AIRINV_Service& lAIRINV_Service =
00565       lAIRINV_Master_ServiceContext.getAIRINV_Service();
00566
00567     // Delegate the BOM display to the dedicated service
00568     return lAIRINV_Service.list (iAirlineCode, iFlightNumber);
00569   }
00570
00571   // ////////////////////////////////////////////////////////////////////
00572   bool AIRINV_Master_Service::
00573   check (const stdair::AirlineCode_T& iAirlineCode,
00574          const stdair::FlightNumber_T& iFlightNumber,
00575          const stdair::Date_T& iDepartureDate) const {
00576     std::ostringstream oFlightListStr;
00577
00578     // Retrieve the AirInv Master service context
00579     if (_airinvMasterServiceContext == NULL) {
00580       throw stdair::NonInitialisedServiceException ("The AirInvMaster service "
00581                                                     "has not been initialised")
      ;
00582     }
00583     assert (_airinvMasterServiceContext != NULL);
00584
```

```
00585     AIRINV_Master_ServiceContext&
      lAIRINV_Master_ServiceContext =
00586       *_airinvMasterServiceContext;
00587
00588     // Retrieve the slave AirInv (slave) service object from
00589     // the (AirInv master) service context
00590     AIRINV_Service& lAIRINV_Service =
00591       lAIRINV_Master_ServiceContext.getAIRINV_Service();
00592
00593     // Delegate the BOM display to the dedicated service
00594     return lAIRINV_Service.check (iAirlineCode, iFlightNumber,
      iDepartureDate);
00595   }
00596
00597   // ////////////////////////////////////////////////////////////////////
00598   std::string AIRINV_Master_Service::csvDisplay
      () const {
00599
00600     // Retrieve the AirInv Master service context
00601     if (_airinvMasterServiceContext == NULL) {
00602       throw stdair::NonInitialisedServiceException ("The AirInvMaster service "
00603                                                     "has not been initialised")
      ;
00604     }
00605     assert (_airinvMasterServiceContext != NULL);
00606
00607     AIRINV_Master_ServiceContext&
      lAIRINV_Master_ServiceContext =
00608       *_airinvMasterServiceContext;
00609
00610     // Retrieve the slave AIRINV service object from
00611     // the (AIRINV) service context
00612     AIRINV_Service& lAIRINV_Service =
00613       lAIRINV_Master_ServiceContext.getAIRINV_Service();
00614
00615     // Delegate the BOM display to the dedicated service
00616     return lAIRINV_Service.csvDisplay();
00617   }
00618
00619   // ////////////////////////////////////////////////////////////////////
00620   std::string AIRINV_Master_Service::
00621   csvDisplay (const stdair::AirlineCode_T& iAirlineCode,
00622               const stdair::FlightNumber_T& iFlightNumber,
00623               const stdair::Date_T& iDepartureDate) const {
00624
00625     // Retrieve the AirInv Master service context
00626     if (_airinvMasterServiceContext == NULL) {
00627       throw stdair::NonInitialisedServiceException ("The AirInvMaster service "
00628                                                     "has not been initialised")
      ;
00629     }
00630     assert (_airinvMasterServiceContext != NULL);
00631
00632     AIRINV_Master_ServiceContext&
      lAIRINV_Master_ServiceContext =
00633       *_airinvMasterServiceContext;
00634
00635     // Retrieve the slave AIRINV service object from
00636     // the (AIRINV) service context
00637     AIRINV_Service& lAIRINV_Service =
00638       lAIRINV_Master_ServiceContext.getAIRINV_Service();
00639
00640     // Delegate the BOM display to the dedicated service
00641     return lAIRINV_Service.csvDisplay (iAirlineCode, iFlightNumber,
00642                                        iDepartureDate);
00643   }
00644
00645   // ////////////////////////////////////////////////////////////////////
00646   void AIRINV_Master_Service::
00647   initSnapshotAndRMEvents (const stdair::Date_T&
      iStartDate,
00648                            const stdair::Date_T& iEndDate) {
00649
00650     // Retrieve the AirInv Master service context
00651     if (_airinvMasterServiceContext == NULL) {
00652       throw stdair::NonInitialisedServiceException ("The AirInvMaster service "
00653                                                     "has not been initialised")
      ;
00654     }
00655     assert (_airinvMasterServiceContext != NULL);
00656
00657     AIRINV_Master_ServiceContext&
      lAIRINV_Master_ServiceContext =
00658       *_airinvMasterServiceContext;
00659
00660     // Retrieve the pointer on the SEvMgr service context
00661     SEVMGR::SEVMGR_ServicePtr_T lSEVMGR_Service_ptr =
```

```
00662       lAIRINV_Master_ServiceContext.getSEVMGR_ServicePtr();
00663     assert (lSEVMGR_Service_ptr != NULL);
00664
00665     // Initialise the snapshot events
00666     InventoryManager::initSnapshotEvents (lSEVMGR_Service_ptr, iStartDate,
      iEndDate);
00667
00668     // \todo Browse the list of inventories and itinialise the RM events of
00669     //       each inventory.
00670
00671     // Retrieve the slave AIRINV service object from the (AIRINV)
00672     // service context
00673     AIRINV_Service& lAIRINV_Service =
00674       lAIRINV_Master_ServiceContext.getAIRINV_Service();
00675     lSEVMGR_Service_ptr->addStatus (stdair::EventType::RM, 0);
00676     stdair::RMEventList_T lRMEventList =
00677       lAIRINV_Service.initRMEvents (iStartDate, iEndDate);
00678     assert (lRMEventList.empty() == false);
00679     InventoryManager::addRMEventsToEventQueue (lSEVMGR_Service_ptr,
      lRMEventList);
00680   }
00681
00682   // ////////////////////////////////////////////////////////////////////
00683   void AIRINV_Master_Service::
00684   calculateAvailability (stdair::TravelSolutionStruct&
      ioTravelSolution) {
00685
00686     // Retrieve the AirInv Master service context
00687     if (_airinvMasterServiceContext == NULL) {
00688       throw stdair::NonInitialisedServiceException ("The AirInvMaster service "
00689                                                     "has not been initialised")
      ;
00690     }
00691     assert (_airinvMasterServiceContext != NULL);
00692
00693     AIRINV_Master_ServiceContext&
      lAIRINV_Master_ServiceContext =
00694       *_airinvMasterServiceContext;
00695
00696     // Retrieve the slave AIRINV service object from the (AIRINV)
00697     // service context
00698     AIRINV_Service& lAIRINV_Service =
00699       lAIRINV_Master_ServiceContext.getAIRINV_Service();
00700
00701     // Delegate the availability retrieval to the dedicated service
00702     stdair::BasChronometer lAvlChronometer;
00703     lAvlChronometer.start();
00704
00705     lAIRINV_Service.calculateAvailability (
      ioTravelSolution);
00706
00707     // DEBUG
00708     // const double lAvlMeasure = lAvlChronometer.elapsed();
00709     // STDAIR_LOG_DEBUG ("Availability retrieval: " << lAvlMeasure << " - "
00710     //                   << lAIRINV_Master_ServiceContext.display());
00711   }
00712
00713   // ////////////////////////////////////////////////////////////////////
00714   bool AIRINV_Master_Service::sell (const
      std::string& iSegmentDateKey,
00715                                     const stdair::ClassCode_T& iClassCode,
00716                                     const stdair::PartySize_T& iPartySize) {
00717
00718     // Retrieve the AirInv Master service context
00719     if (_airinvMasterServiceContext == NULL) {
00720       throw stdair::NonInitialisedServiceException ("The AirInvMaster service "
00721                                                     "has not been initialised")
      ;
00722     }
00723     assert (_airinvMasterServiceContext != NULL);
00724
00725     AIRINV_Master_ServiceContext&
      lAIRINV_Master_ServiceContext =
00726       *_airinvMasterServiceContext;
00727
00728     // Retrieve the corresponding inventory key
00729     // const stdair::InventoryKey& lInventoryKey =
00730     // stdair::BomKeyManager::extractInventoryKey (iSegmentDateKey);
00731
00732     // Retrieve the slave AirInv service object from the (AirInv Master)
00733     // service context
00734     AIRINV_Service& lAIRINV_Service =
00735       lAIRINV_Master_ServiceContext.getAIRINV_Service();
00736
00737     // Delegate the booking to the dedicated command
00738     stdair::BasChronometer lSellChronometer;
00739     lSellChronometer.start();
```

```
00740
00741     // Delegate the BOM building to the dedicated service
00742     const bool hasBeenSaleSuccessful =
00743       lAIRINV_Service.sell (iSegmentDateKey, iClassCode, iPartySize);
00744
00745     // const double lSellMeasure = lSellChronometer.elapsed();
00746
00747     // DEBUG
00748     // STDAIR_LOG_DEBUG ("Booking sell: " << lSellMeasure << " - "
00749     //                   << lAIRINV_Master_ServiceContext.display());
00750
00751     //
00752     return hasBeenSaleSuccessful;
00753   }
00754
00755   // ////////////////////////////////////////////////////////////////////
00756   bool AIRINV_Master_Service::sell (const
      stdair::BookingClassID_T& iClassID,
00757                                     const stdair::PartySize_T& iPartySize) {
00758
00759     // Retrieve the AirInv Master service context
00760     if (_airinvMasterServiceContext == NULL) {
00761       throw stdair::NonInitialisedServiceException ("The AirInvMaster service "
00762                                                     "has not been initialised")
      ;
00763     }
00764     assert (_airinvMasterServiceContext != NULL);
00765
00766     AIRINV_Master_ServiceContext&
      lAIRINV_Master_ServiceContext =
00767       *_airinvMasterServiceContext;
00768
00769     // Retrieve the corresponding inventory key
00770     // const stdair::InventoryKey& lInventoryKey =
00771     // stdair::BomKeyManager::extractInventoryKey (iSegmentDateKey);
00772
00773     // Retrieve the slave AirInv service object from the (AirInv Master)
00774     // service context
00775     AIRINV_Service& lAIRINV_Service =
00776       lAIRINV_Master_ServiceContext.getAIRINV_Service();
00777
00778     // Delegate the booking to the dedicated command
00779     stdair::BasChronometer lSellChronometer;
00780     lSellChronometer.start();
00781
00782     // Delegate the BOM building to the dedicated service
00783     const bool hasBeenSaleSuccessful =
00784       lAIRINV_Service.sell (iClassID, iPartySize);
00785
00786     // const double lSellMeasure = lSellChronometer.elapsed();
00787
00788     // DEBUG
00789     // STDAIR_LOG_DEBUG ("Booking sell: " << lSellMeasure << " - "
00790     //                   << lAIRINV_Master_ServiceContext.display());
00791
00792     //
00793     return hasBeenSaleSuccessful;
00794   }
00795
00796   // ////////////////////////////////////////////////////////////////////
00797   bool AIRINV_Master_Service::cancel (const
      std::string& iSegmentDateKey,
00798                                       const stdair::ClassCode_T& iClassCode,
00799                                       const stdair::PartySize_T& iPartySize) {
00800
00801     // Retrieve the AirInv Master service context
00802     if (_airinvMasterServiceContext == NULL) {
00803       throw stdair::NonInitialisedServiceException ("The AirInvMaster service "
00804                                                     "has not been initialised")
      ;
00805     }
00806     assert (_airinvMasterServiceContext != NULL);
00807
00808     AIRINV_Master_ServiceContext&
      lAIRINV_Master_ServiceContext =
00809       *_airinvMasterServiceContext;
00810
00811     // Retrieve the corresponding inventory key
00812     // const stdair::InventoryKey& lInventoryKey =
00813     // stdair::BomKeyManager::extractInventoryKey (iSegmentDateKey);
00814
00815     // Retrieve the slave AirInv service object from the (AirInv Master)
00816     // service context
00817     AIRINV_Service& lAIRINV_Service =
00818       lAIRINV_Master_ServiceContext.getAIRINV_Service();
00819
00820     // Delegate the booking to the dedicated command
```

```
00821     stdair::BasChronometer lCancelChronometer;
00822     lCancelChronometer.start();
00823
00824     // Delegate the BOM building to the dedicated service
00825     const bool hasBeenSaleSuccessful =
00826       lAIRINV_Service.cancel (iSegmentDateKey, iClassCode, iPartySize);
00827
00828     // const double lCancelMeasure = lCancelChronometer.elapsed();
00829
00830     // DEBUG
00831     // STDAIR_LOG_DEBUG ("Booking cancel: " << lCancelMeasure << " - "
00832     //                   << lAIRINV_Master_ServiceContext.display());
00833
00834     //
00835     return hasBeenSaleSuccessful;
00836   }
00837
00838   // ////////////////////////////////////////////////////////////////////
00839   bool AIRINV_Master_Service::cancel (const
      stdair::BookingClassID_T& iClassID,
00840                                       const stdair::PartySize_T& iPartySize) {
00841
00842     // Retrieve the AirInv Master service context
00843     if (_airinvMasterServiceContext == NULL) {
00844       throw stdair::NonInitialisedServiceException ("The AirInvMaster service "
00845                                                     "has not been initialised")
      ;
00846     }
00847     assert (_airinvMasterServiceContext != NULL);
00848
00849     AIRINV_Master_ServiceContext&
      lAIRINV_Master_ServiceContext =
00850       *_airinvMasterServiceContext;
00851
00852     // Retrieve the corresponding inventory key
00853     // const stdair::InventoryKey& lInventoryKey =
00854     // stdair::BomKeyManager::extractInventoryKey (iSegmentDateKey);
00855
00856     // Retrieve the slave AirInv service object from the (AirInv Master)
00857     // service context
00858     AIRINV_Service& lAIRINV_Service =
00859       lAIRINV_Master_ServiceContext.getAIRINV_Service();
00860
00861     // Delegate the booking to the dedicated command
00862     stdair::BasChronometer lCancelChronometer;
00863     lCancelChronometer.start();
00864
00865     // Delegate the BOM building to the dedicated service
00866     const bool hasBeenSaleSuccessful =
00867       lAIRINV_Service.cancel (iClassID, iPartySize);
00868
00869     // const double lCancelMeasure = lCancelChronometer.elapsed();
00870
00871     // DEBUG
00872     // STDAIR_LOG_DEBUG ("Booking cancel: " << lCancelMeasure << " - "
00873     //                   << lAIRINV_Master_ServiceContext.display());
00874
00875     //
00876     return hasBeenSaleSuccessful;
00877   }
00878
00879   // ////////////////////////////////////////////////////////////////////
00880   void AIRINV_Master_Service::
00881   takeSnapshots (const stdair::SnapshotStruct& iSnapshot) {
00882
00883     // Retrieve the AirInv Master service context
00884     if (_airinvMasterServiceContext == NULL) {
00885       throw stdair::NonInitialisedServiceException ("The AirInvMaster service "
00886                                                     "has not been initialised")
      ;
00887     }
00888     assert (_airinvMasterServiceContext != NULL);
00889
00890     AIRINV_Master_ServiceContext&
      lAIRINV_Master_ServiceContext =
00891       *_airinvMasterServiceContext;
00892
00893     // Retrieve the slave AIRINV service object from the (AIRINV)
00894     // service context
00895     AIRINV_Service& lAIRINV_Service =
00896       lAIRINV_Master_ServiceContext.getAIRINV_Service();
00897
00898     // Retrieve  the snapshot time and the airline code.
00899     const stdair::DateTime_T& lSnapshotTime = iSnapshot.getSnapshotTime();
00900     const stdair::AirlineCode_T& lAirlineCode = iSnapshot.getAirlineCode();
00901
00902     lAIRINV_Service.takeSnapshots (lAirlineCode, lSnapshotTime);
```

```
00903   }
00904
00905   // ////////////////////////////////////////////////////////////////////
00906   void AIRINV_Master_Service::
00907   optimise (const stdair::RMEventStruct& iRMEvent) {
00908
00909     // Retrieve the AirInv Master service context
00910     if (_airinvMasterServiceContext == NULL) {
00911       throw stdair::NonInitialisedServiceException ("The AirInvMaster service "
00912                                                     "has not been initialised")
      ;
00913     }
00914     assert (_airinvMasterServiceContext != NULL);
00915
00916     AIRINV_Master_ServiceContext&
      lAIRINV_Master_ServiceContext =
00917       *_airinvMasterServiceContext;
00918
00919     // Retrieve the slave AIRINV service object from the (AIRINV)
00920     // service context
00921     AIRINV_Service& lAIRINV_Service =
00922       lAIRINV_Master_ServiceContext.getAIRINV_Service();
00923
00924     // Retrieve  the snapshot time and the airline code.
00925     const stdair::DateTime_T& lRMEventTime = iRMEvent.getRMEventTime();
00926     const stdair::AirlineCode_T& lAirlineCode = iRMEvent.getAirlineCode();
00927     const stdair::KeyDescription_T& lFDDescription =
00928       iRMEvent.getFlightDateDescription();
00929
00930     lAIRINV_Service.optimise (lAirlineCode, lFDDescription,
      lRMEventTime);
00931   }
00932 }
```

23.241 airinv/service/AIRINV_Master_ServiceContext.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <airinv/basic/BasConst_AIRINV_Service.hpp>
#include <airinv/service/AIRINV_Master_ServiceContext.hpp>
```

Namespaces

- namespace AIRINV

23.242 AIRINV_Master_ServiceContext.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Airinv
00008 #include <airinv/basic/BasConst_AIRINV_Service.hpp
      >
00009 #include <airinv/service/AIRINV_Master_ServiceContext.hpp
      >
00010
00011 namespace AIRINV {
00012
00013   // ////////////////////////////////////////////////////////////////////
00014   AIRINV_Master_ServiceContext::AIRINV_Master_ServiceContext()
00015     : _ownStdairService (false),
00016       _ownSEVMGRService (true) {
00017   }
00018
00019   // ////////////////////////////////////////////////////////////////////
00020   AIRINV_Master_ServiceContext::~AIRINV_Master_ServiceContext() {
00021   }
00022
00023   // ////////////////////////////////////////////////////////////////////
00024   const std::string AIRINV_Master_ServiceContext::shortDisplay() const {
00025     std::ostringstream oStr;
00026     oStr << "AIRINV_Master_ServiceContext -- Owns StdAir service: "
```

```
00027         << _ownStdairService;
00028     return oStr.str();
00029   }
00030
00031   // ////////////////////////////////////////////////////////////////////
00032   const std::string AIRINV_Master_ServiceContext::display() const {
00033     std::ostringstream oStr;
00034     oStr << shortDisplay();
00035     return oStr.str();
00036   }
00037
00038   // ////////////////////////////////////////////////////////////////////
00039   const std::string AIRINV_Master_ServiceContext::describe() const {
00040     return shortDisplay();
00041   }
00042
00043   // ////////////////////////////////////////////////////////////////////
00044   void AIRINV_Master_ServiceContext::reset() {
00045
00046     // The shared_ptr<>::reset() method drops the refcount by one.
00047     // If the count result is dropping to zero, the resource pointed to
00048     // by the shared_ptr<> will be freed.
00049
00050     // Reset the stdair shared pointer
00051     _stdairService.reset();
00052
00053     // Reset the sevmgr shared pointer
00054     _sevmgrService.reset();
00055
00056     // Reset the airinv shared pointer
00057     _airinvService.reset();
00058   }
00059
00060 }
```

23.243 airinv/service/AIRINV_Master_ServiceContext.hpp File Reference

```
#include <string>
#include <boost/shared_ptr.hpp>
#include <stdair/stdair_service_types.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/service/ServiceAbstract.hpp>
#include <sevmgr/SEVMGR_Types.hpp>
#include <airinv/AIRINV_Types.hpp>
```

Classes

- class AIRINV::AIRINV_Master_ServiceContext

Namespaces

- namespace AIRINV

23.244 AIRINV_Master_ServiceContext.hpp

```
00001 #ifndef __AIRINV_SVC_AIRINVMASTERSERVICECONTEXT_HPP
00002 #define __AIRINV_SVC_AIRINVMASTERSERVICECONTEXT_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // Boost
00010 #include <boost/shared_ptr.hpp>
00011 // StdAir
00012 #include <stdair/stdair_service_types.hpp>
00013 #include <stdair/bom/Inventory.hpp>
00014 #include <stdair/service/ServiceAbstract.hpp>
00015 // SEvMgr
```

```
00016 #include <sevmgr/SEVMGR_Types.hpp>
00017 // AirInv
00018 #include <airinv/AIRINV_Types.hpp>
00019
00020 namespace AIRINV {
00021
00023   class AIRINV_Service;
00024
00028   class AIRINV_Master_ServiceContext : public
      stdair::ServiceAbstract {
00034     friend class AIRINV_Master_Service;
00035     friend class FacAirinvMasterServiceContext;
00036
00037   private:
00038     // ////////////////// Getters ////////////////////
00042     stdair::STDAIR_ServicePtr_T getSTDAIR_ServicePtr() const {
00043       return _stdairService;
00044     }
00045
00049     SEVMGR::SEVMGR_ServicePtr_T getSEVMGR_ServicePtr() const {
00050       return _sevmgrService;
00051     }
00052
00056     stdair::STDAIR_Service& getSTDAIR_Service() const {
00057       assert (_stdairService != NULL);
00058       return *_stdairService;
00059     }
00060
00064     const bool getOwnStdairServiceFlag() const {
00065       return _ownStdairService;
00066     }
00067
00071     const bool getOwnSEVMGRServiceFlag() const {
00072       return _ownSEVMGRService;
00073     }
00074
00079     AIRINV_Service& getAIRINV_Service() const {
00080       assert (_airinvService != NULL);
00081       return *_airinvService;
00082     }
00083
00084     // ////////////////// Setters ////////////////////
00088     void setSTDAIR_Service (stdair::STDAIR_ServicePtr_T ioSTDAIR_ServicePtr,
00089                             const bool iOwnStdairService) {
00090       _stdairService = ioSTDAIR_ServicePtr;
00091       _ownStdairService = iOwnStdairService;
00092     }
00093
00097     void setSEVMGR_Service (SEVMGR::SEVMGR_ServicePtr_T ioSEVMGR_ServicePtr,
00098                             const bool iOwnSEVMGRService) {
00099       _sevmgrService = ioSEVMGR_ServicePtr;
00100       _ownSEVMGRService = iOwnSEVMGRService;
00101     }
00102
00106     void setAIRINV_Service (AIRINV_ServicePtr_T
      ioAIRINV_ServicePtr) {
00107       _airinvService = ioAIRINV_ServicePtr;
00108     }
00109
00110
00111   private:
00112     // ////////////////////// Display Methods /////////////////////
00116     const std::string shortDisplay() const;
00117
00121     const std::string display() const;
00122
00126     const std::string describe() const;
00127
00128
00129   private:
00131
00134     AIRINV_Master_ServiceContext();
00138     AIRINV_Master_ServiceContext (const AIRINV_Master_ServiceContext&);
00139
00143     ~AIRINV_Master_ServiceContext();
00144
00148     void reset();
00149
00150
00151   private:
00152     // ///////////////// Children ////////////////
00156     stdair::STDAIR_ServicePtr_T _stdairService;
00157
00161     bool _ownStdairService;
00162
00166     SEVMGR::SEVMGR_ServicePtr_T _sevmgrService;
00167
```

```
00171     bool _ownSEVMGRService;
00172
00173
00174   private:
00175     // ////////////// Attributes ////////////////////
00179     AIRINV_ServicePtr_T _airinvService;
00180   };
00181
00182 }
00183 #endif // __AIRINV_SVC_AIRINVMASTERSERVICECONTEXT_HPP
```

23.245 airinv/service/AIRINV_Service.cpp File Reference

```
#include <cassert>
#include <boost/make_shared.hpp>
#include <stdair/stdair_json.hpp>
#include <stdair/basic/BasChronometer.hpp>
#include <stdair/basic/JSonCommand.hpp>
#include <stdair/basic/PartnershipTechnique.hpp>
#include <stdair/basic/UnconstrainingMethod.hpp>
#include <stdair/basic/OptimisationMethod.hpp>
#include <stdair/bom/BomKeyManager.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/AirlineFeature.hpp>
#include <stdair/bom/RMEventStruct.hpp>
#include <stdair/bom/BomJSONImport.hpp>
#include <stdair/bom/BomJSONExport.hpp>
#include <stdair/factory/FacBomManager.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/STDAIR_Service.hpp>
#include <rmol/RMOL_Service.hpp>
#include <airrac/AIRRAC_Service.hpp>
#include <sevmgr/SEVMGR_Service.hpp>
#include <airinv/basic/BasConst_AIRINV_Service.hpp>
#include <airinv/factory/FacAirinvServiceContext.hpp>
#include <airinv/command/ScheduleParser.hpp>
#include <airinv/command/FRAT5Parser.hpp>
#include <airinv/command/FFDisutilityParser.hpp>
#include <airinv/command/InventoryParser.hpp>
#include <airinv/command/InventoryManager.hpp>
#include <airinv/command/InventoryBuilder.hpp>
#include <airinv/service/AIRINV_ServiceContext.hpp>
#include <airinv/AIRINV_Service.hpp>
```

Namespaces

- namespace AIRINV

23.246 AIRINV_Service.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // Boost
```

```
00007 #include <boost/make_shared.hpp>
00008 // StdAir
00009 #include <stdair/stdair_json.hpp>
00010 #include <stdair/basic/BasChronometer.hpp>
00011 #include <stdair/basic/JSonCommand.hpp>
00012 #include <stdair/basic/PartnershipTechnique.hpp>
00013 #include <stdair/basic/UnconstrainingMethod.hpp>
00014 #include <stdair/basic/OptimisationMethod.hpp>
00015 #include <stdair/bom/BomKeyManager.hpp>
00016 #include <stdair/bom/BomManager.hpp>
00017 #include <stdair/bom/BomRoot.hpp>
00018 #include <stdair/bom/Inventory.hpp>
00019 #include <stdair/bom/FlightDate.hpp>
00020 #include <stdair/bom/SegmentCabin.hpp>
00021 #include <stdair/bom/AirlineFeature.hpp>
00022 #include <stdair/bom/RMEventStruct.hpp>
00023 #include <stdair/bom/BomJSONImport.hpp>
00024 #include <stdair/bom/BomJSONExport.hpp>
00025 #include <stdair/factory/FacBomManager.hpp>
00026 #include <stdair/service/Logger.hpp>
00027 #include <stdair/STDAIR_Service.hpp>
00028 // RMOL
00029 #include <rmol/RMOL_Service.hpp>
00030 // AirRAC
00031 #include <airrac/AIRRAC_Service.hpp>
00032 // SEvMgr
00033 #include <sevmgr/SEVMGR_Service.hpp>
00034 // AirInv
00035 #include <airinv/basic/BasConst_AIRINV_Service.hpp
      >
00036 #include <airinv/factory/FacAirinvServiceContext.hpp
      >
00037 #include <airinv/command/ScheduleParser.hpp>
00038 #include <airinv/command/FRAT5Parser.hpp>
00039 #include <airinv/command/FFDisutilityParser.hpp
      >
00040 #include <airinv/command/InventoryParser.hpp>
00041 #include <airinv/command/InventoryManager.hpp
      >
00042 #include <airinv/command/InventoryBuilder.hpp
      >
00043 #include <airinv/service/AIRINV_ServiceContext.hpp
      >
00044 #include <airinv/AIRINV_Service.hpp>
00045
00046 namespace AIRINV {
00047
00048   // ////////////////////////////////////////////////////////////////////
00049   AIRINV_Service::AIRINV_Service () : _airinvServiceContext (NULL) {
00050     assert (false);
00051   }
00052
00053   // ////////////////////////////////////////////////////////////////////
00054   AIRINV_Service::AIRINV_Service (const AIRINV_Service& iService)
00055   : _airinvServiceContext (NULL) {
00056     assert (false);
00057   }
00058
00059   // ////////////////////////////////////////////////////////////////////
00060   AIRINV_Service::AIRINV_Service (const stdair::BasLogParams& iLogParams)
00061     : _airinvServiceContext (NULL) {
00062
00063     // Initialise the STDAIR service handler
00064     stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00065       initStdAirService (iLogParams);
00066
00067     // Initialise the service context
00068     initServiceContext();
00069
00070     // Add the StdAir service context to the AIRINV service context
00071     // \note AIRINV owns the STDAIR service resources here.
00072     const bool ownStdairService = true;
00073     addStdAirService (lSTDAIR_Service_ptr, ownStdairService);
00074
00075     // Initalise the RMOL service.
00076     initRMOLService();
00077
00078     // Initalise the AIRRAC service.
00079     initAIRRACService();
00080
00081     // Initalise the SEvMgr service.
00082     initSEVMGRService();
00083
00084     // Initialise the (remaining of the) context
00085     initAirinvService();
00086   }
00087
```

```
00088   // ////////////////////////////////////////////////////////////////////
00089   AIRINV_Service::AIRINV_Service (const stdair::BasLogParams& iLogParams,
00090                                   const stdair::BasDBParams& iDBParams)
00091     : _airinvServiceContext (NULL) {
00092
00093     // Initialise the STDAIR service handler
00094     stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00095       initStdAirService (iLogParams, iDBParams);
00096
00097     // Initialise the service context
00098     initServiceContext();
00099
00100    // Add the StdAir service context to the AIRINV service context
00101    // \note AIRINV owns the STDAIR service resources here.
00102    const bool ownStdairService = true;
00103    addStdAirService (lSTDAIR_Service_ptr, ownStdairService);
00104
00105    // Initalise the RMOL service.
00106    initRMOLService();
00107
00108    // Initalise the AIRRAC service.
00109    initAIRRACService();
00110
00111    // Initalise the SEVMGR service.
00112    initSEVMGRService();
00113
00114    // Initialise the (remaining of the) context
00115    initAirinvService();
00116  }
00117
00118  // ////////////////////////////////////////////////////////////////////
00119  AIRINV_Service::
00120  AIRINV_Service (stdair::STDAIR_ServicePtr_T ioSTDAIR_Service_ptr)
00121
00122    : _airinvServiceContext (NULL) {
00123
00124    // Initialise the service context
00125    initServiceContext();
00126
00127    // Store the STDAIR service object within the (AIRINV) service context
00128    // \note AirInv does not own the STDAIR service resources here.
00129    const bool doesNotOwnStdairService = false;
00130    addStdAirService (ioSTDAIR_Service_ptr, doesNotOwnStdairService);
00131
00132    // Initalise the RMOL service.
00133    initRMOLService();
00134
00135    // Initalise the AIRRAC service.
00136    initAIRRACService();
00137
00138    // Initalise the SEVMGR service.
00139    initSEVMGRService();
00140
00141    // Initialise the (remaining of the) context
00142    initAirinvService();
00143
00144  }
00145
00146  // ////////////////////////////////////////////////////////////////////
00147  AIRINV_Service::
00148  AIRINV_Service (stdair::STDAIR_ServicePtr_T ioSTDAIR_Service_ptr,
00149                  SEVMGR::SEVMGR_ServicePtr_T ioSEVMGR_Service_ptr)
00150
00151    : _airinvServiceContext (NULL) {
00152
00153    // Initialise the service context
00154    initServiceContext();
00155
00156    // Store the STDAIR service object within the (AIRINV) service context
00157    // \note AirInv does not own the STDAIR service resources here.
00158    const bool doesNotOwnStdairService = false;
00159    addStdAirService (ioSTDAIR_Service_ptr, doesNotOwnStdairService);
00160
00161    //Add the SEvMgr service to the TRADEMGEN service context.
00162    const bool doesNotOwnSEVMGRService = false;
00163    addSEVMGRService (ioSEVMGR_Service_ptr, doesNotOwnSEVMGRService);
00164
00165    // Initalise the RMOL service.
00166    initRMOLService();
00167
00168    // Initalise the AIRRAC service.
00169    initAIRRACService();
00170
00171    // Initialise the (remaining of the) context
00172    initAirinvService();
00173
00174  }
```

```
00175
00176   // ////////////////////////////////////////////////////////////////////
00177   AIRINV_Service::~AIRINV_Service() {
00178     // Delete/Clean all the objects from memory
00179     finalise();
00180   }
00181
00182   // ////////////////////////////////////////////////////////////////////
00183   void AIRINV_Service::finalise() {
00184     assert (_airinvServiceContext != NULL);
00185     // Reset the (Boost.)Smart pointer pointing on the STDAIR_Service object.
00186     _airinvServiceContext->reset();
00187   }
00188
00189   // ////////////////////////////////////////////////////////////////////
00190   void AIRINV_Service::initServiceContext() {
00191     // Initialise the context
00192     AIRINV_ServiceContext& lAIRINV_ServiceContext =
00193       FacAirinvServiceContext::instance().
      create();
00194     _airinvServiceContext = &lAIRINV_ServiceContext;
00195   }
00196
00197   // ////////////////////////////////////////////////////////////////////
00198   void AIRINV_Service::
00199   addStdAirService (stdair::STDAIR_ServicePtr_T ioSTDAIR_Service_ptr,
00200                     const bool iOwnStdairService) {
00201
00202     // Retrieve the Airinv service context
00203     assert (_airinvServiceContext != NULL);
00204     AIRINV_ServiceContext& lAIRINV_ServiceContext = *_airinvServiceContext;
00205
00206     // Store the STDAIR service object within the (AIRINV) service context
00207     lAIRINV_ServiceContext.setSTDAIR_Service (ioSTDAIR_Service_ptr,
00208                                               iOwnStdairService);
00209   }
00210
00211  // ////////////////////////////////////////////////////////////////////
00212   void AIRINV_Service::
00213   addSEVMGRService (SEVMGR::SEVMGR_ServicePtr_T ioSEVMGR_Service_ptr,
00214                     const bool iOwnSEVMGRService) {
00215
00216     // Retrieve the Airinv service context
00217     assert (_airinvServiceContext != NULL);
00218     AIRINV_ServiceContext& lAIRINV_ServiceContext = *_airinvServiceContext;
00219
00220     // Store the STDAIR service object within the (TRADEMGEN) service context
00221     lAIRINV_ServiceContext.setSEVMGR_Service (ioSEVMGR_Service_ptr,
00222                                               iOwnSEVMGRService);
00223   }
00224
00225   // ////////////////////////////////////////////////////////////////////
00226   stdair::STDAIR_ServicePtr_T AIRINV_Service::
00227   initStdAirService (const stdair::BasLogParams& iLogParams,
00228                      const stdair::BasDBParams& iDBParams) {
00229
00237     stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00238       boost::make_shared<stdair::STDAIR_Service> (iLogParams, iDBParams);
00239
00240     return lSTDAIR_Service_ptr;
00241   }
00242
00243   // ////////////////////////////////////////////////////////////////////
00244   stdair::STDAIR_ServicePtr_T AIRINV_Service::
00245   initStdAirService (const stdair::BasLogParams& iLogParams) {
00246
00254     stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00255       boost::make_shared<stdair::STDAIR_Service> (iLogParams);
00256
00257     return lSTDAIR_Service_ptr;
00258   }
00259
00260   // ////////////////////////////////////////////////////////////////////
00261   void AIRINV_Service::initRMOLService() {
00262
00263     // Retrieve the AirInv service context
00264     assert (_airinvServiceContext != NULL);
00265     AIRINV_ServiceContext& lAIRINV_ServiceContext = *_airinvServiceContext;
00266
00267     // Retrieve the StdAir service context
00268     stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00269       lAIRINV_ServiceContext.getSTDAIR_ServicePtr();
00270
00278     RMOL::RMOL_ServicePtr_T lRMOL_Service_ptr =
00279       boost::make_shared<RMOL::RMOL_Service> (lSTDAIR_Service_ptr);
00280
00281     // Store the RMOL service object within the (AIRINV) service context
```

```
00282     lAIRINV_ServiceContext.setRMOL_Service (lRMOL_Service_ptr);
00283   }
00284
00285   // ////////////////////////////////////////////////////////////////////
00286   void AIRINV_Service::initAIRRACService() {
00287
00288     // Retrieve the AirInv service context
00289     assert (_airinvServiceContext != NULL);
00290     AIRINV_ServiceContext& lAIRINV_ServiceContext = *_airinvServiceContext;
00291
00292     // Retrieve the StdAir service context
00293     stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00294       lAIRINV_ServiceContext.getSTDAIR_ServicePtr();
00295
00303     AIRRAC::AIRRAC_ServicePtr_T lAIRRAC_Service_ptr =
00304       boost::make_shared<AIRRAC::AIRRAC_Service> (lSTDAIR_Service_ptr);
00305
00306     // Store the AIRRAC service object within the (AIRINV) service context
00307     lAIRINV_ServiceContext.setAIRRAC_Service (lAIRRAC_Service_ptr);
00308   }
00309
00310   // ////////////////////////////////////////////////////////////////////
00311   void AIRINV_Service::initSEVMGRService() {
00312
00313     // Retrieve the AIRINV service context
00314     assert (_airinvServiceContext != NULL);
00315     AIRINV_ServiceContext& lAIRINV_ServiceContext =
00316       *_airinvServiceContext;
00317
00318     // Retrieve the StdAir service context
00319     stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00320       lAIRINV_ServiceContext.getSTDAIR_ServicePtr();
00321
00329     SEVMGR::SEVMGR_ServicePtr_T lSEVMGR_Service_ptr =
00330       boost::make_shared<SEVMGR::SEVMGR_Service> (lSTDAIR_Service_ptr);
00331
00332     // Store the SEvMgr service object within the (TraDemGen) service context
00333     const bool doesOwnSEVMGRService = true;
00334     lAIRINV_ServiceContext.setSEVMGR_Service (lSEVMGR_Service_ptr,
00335                                               doesOwnSEVMGRService);
00336   }
00337
00338   // ////////////////////////////////////////////////////////////////////
00339   void AIRINV_Service::initAirinvService() {
00340     // Do nothing at this stage. A sample BOM tree may be built by
00341     // calling the buildSampleBom() method
00342   }
00343
00344   // ////////////////////////////////////////////////////////////////////
00345   void AIRINV_Service::
00346   parseAndLoad (const AIRINV::InventoryFilePath
      & iInventoryInputFilename) {
00347
00348     // Retrieve the AirInv service context
00349     if (_airinvServiceContext == NULL) {
00350       throw stdair::NonInitialisedServiceException("The AirInv service has not
      "
00351                                                    "been initialised");
00352     }
00353     assert (_airinvServiceContext != NULL);
00354
00355     // Retrieve the AirInv service context and whether it owns the Stdair
00356     // service
00357     AIRINV_ServiceContext& lAIRINV_ServiceContext = *
      _airinvServiceContext;
00358     const bool doesOwnStdairService =
00359       lAIRINV_ServiceContext.getOwnStdairServiceFlag();
00360
00361     // Retrieve the StdAir service object from the (AirInv) service context
00362     stdair::STDAIR_Service& lSTDAIR_Service =
00363       lAIRINV_ServiceContext.getSTDAIR_Service();
00364
00365     // Retrieve the persistent BOM root object.
00366     stdair::BomRoot& lPersistentBomRoot =
00367       lSTDAIR_Service.getPersistentBomRoot();
00368
00372     InventoryParser::buildInventory (
      iInventoryInputFilename,
00373                                      lPersistentBomRoot);
00374
00378     InventoryBuilder::buildPartnerInventories (lPersistentBomRoot);
00379
00383     lSTDAIR_Service.updateAirlineFeatures();
00384
00389     buildComplementaryLinks (lPersistentBomRoot);
00390
00395     if (doesOwnStdairService == true) {
```

```
00396
00397       //
00398       clonePersistentBom ();
00399     }
00400   }
00401
00402   // ////////////////////////////////////////////////////////////////////
00403   void AIRINV_Service::
00404   parseAndLoad (const stdair::ScheduleFilePath&
      iScheduleInputFilename,
00405                 const stdair::ODFilePath& iODInputFilename,
00406                 const stdair::FRAT5FilePath& iFRAT5InputFilename,
00407                 const stdair::FFDisutilityFilePath& iFFDisutilityInputFilename,
00408                 const AIRRAC::YieldFilePath& iYieldFilename) {
00409
00410     // Retrieve the AirInv service context
00411     if (_airinvServiceContext == NULL) {
00412       throw stdair::NonInitialisedServiceException("The AirInv service has not
      "
00413                                                    "been initialised");
00414     }
00415     assert (_airinvServiceContext != NULL);
00416
00417     // Retrieve the AirInv service context and whether it owns the Stdair
00418     // service
00419     AIRINV_ServiceContext& lAIRINV_ServiceContext = *
      _airinvServiceContext;
00420     const bool doesOwnStdairService =
00421       lAIRINV_ServiceContext.getOwnStdairServiceFlag();
00422
00423     // Retrieve the StdAir service object from the (AirInv) service context
00424     stdair::STDAIR_Service& lSTDAIR_Service =
00425       lAIRINV_ServiceContext.getSTDAIR_Service();
00426
00427     // Retrieve the perssitent BOM root object.
00428     stdair::BomRoot& lPersistentBomRoot =
00429       lSTDAIR_Service.getPersistentBomRoot();
00430
00434     FRAT5Parser::parse (iFRAT5InputFilename,
      lPersistentBomRoot);
00435     FFDisutilityParser::parse (
      iFFDisutilityInputFilename, lPersistentBomRoot);
00436     ScheduleParser::generateInventories (
      iScheduleInputFilename,
00437                                          lPersistentBomRoot);
00438
00442     InventoryBuilder::buildPartnerInventories (lPersistentBomRoot);
00443
00447     lSTDAIR_Service.updateAirlineFeatures();
00448
00453     buildComplementaryLinks (lPersistentBomRoot);
00454
00463     AIRRAC::AIRRAC_Service& lAIRRAC_Service =
00464       lAIRINV_ServiceContext.getAIRRAC_Service();
00465     lAIRRAC_Service.parseAndLoad (iYieldFilename);
00473     lAIRRAC_Service.updateYields(lPersistentBomRoot);
00474
00479     if (doesOwnStdairService == true) {
00480
00481       //
00482       clonePersistentBom ();
00483     }
00484   }
00485
00486   // ////////////////////////////////////////////////////////////////////
00487   void AIRINV_Service::buildSampleBom() {
00488
00489     // Retrieve the AirInv service context
00490     if (_airinvServiceContext == NULL) {
00491       throw stdair::NonInitialisedServiceException("The AirInv service has not
      "
00492                                                    "been initialised");
00493     }
00494     assert (_airinvServiceContext != NULL);
00495
00496     // Retrieve the AirInv service context and whether it owns the Stdair
00497     // service
00498     AIRINV_ServiceContext& lAIRINV_ServiceContext = *
      _airinvServiceContext;
00499     const bool doesOwnStdairService =
00500       lAIRINV_ServiceContext.getOwnStdairServiceFlag();
00501
00502     // Retrieve the StdAir service object from the (AirInv) service context
00503     stdair::STDAIR_Service& lSTDAIR_Service =
00504       lAIRINV_ServiceContext.getSTDAIR_Service();
00505
00506     // Retrieve the perssitent BOM root object.
```

```
00507     stdair::BomRoot& lPersistentBomRoot =
00508       lSTDAIR_Service.getPersistentBomRoot();
00509
00514     if (doesOwnStdairService == true) {
00515       //
00516       lSTDAIR_Service.buildSampleBom();
00517     }
00518
00529     AIRRAC::AIRRAC_Service& lAIRRAC_Service =
00530       lAIRINV_ServiceContext.getAIRRAC_Service();
00531     lAIRRAC_Service.buildSampleBom();
00532
00538     RMOL::RMOL_Service& lRMOL_Service =
00539       lAIRINV_ServiceContext.getRMOL_Service();
00540     lRMOL_Service.buildSampleBom();
00541
00545     InventoryBuilder::buildPartnerInventories (lPersistentBomRoot);
00546
00551     buildComplementaryLinks (lPersistentBomRoot);
00552
00556     lSTDAIR_Service.updateAirlineFeatures();
00557
00562     if (doesOwnStdairService == true) {
00563
00564       //
00565       clonePersistentBom ();
00566     }
00567   }
00568
00569   // ////////////////////////////////////////////////////////////////////
00570   void AIRINV_Service::clonePersistentBom ()
    {
00571
00572     // Retrieve the AirInv service context
00573     if (_airinvServiceContext == NULL) {
00574       throw stdair::NonInitialisedServiceException("The AirInv service has not
    "
00575                                                    "been initialised");
00576     }
00577     assert (_airinvServiceContext != NULL);
00578
00579     // Retrieve the AirInv service context and whether it owns the Stdair
00580     // service
00581     AIRINV_ServiceContext& lAIRINV_ServiceContext = *
    _airinvServiceContext;
00582     const bool doesOwnStdairService =
00583       lAIRINV_ServiceContext.getOwnStdairServiceFlag();
00584
00585     // Retrieve the StdAir service object from the (AirInv) service context
00586     stdair::STDAIR_Service& lSTDAIR_Service =
00587       lAIRINV_ServiceContext.getSTDAIR_Service();
00588
00593     if (doesOwnStdairService == true) {
00594
00595       //
00596       lSTDAIR_Service.clonePersistentBom ();
00597     }
00598
00602     stdair::BomRoot& lBomRoot =
00603       lSTDAIR_Service.getBomRoot();
00604     buildComplementaryLinks (lBomRoot);
00605   }
00606
00607   // ////////////////////////////////////////////////////////////////////
00608   void AIRINV_Service::buildComplementaryLinks
    (stdair::BomRoot& ioBomRoot) {
00609
00620     InventoryManager::createDirectAccesses
    (ioBomRoot);
00621
00626     InventoryManager::buildSimilarSegmentCabinSets
    (ioBomRoot);
00627
00631     InventoryManager::setDefaultBidPriceVector
    (ioBomRoot);
00632
00636     InventoryManager::initialiseYieldBasedNestingStructures
    (ioBomRoot);
00637
00641     InventoryManager::initialiseListsOfUsablePolicies
    (ioBomRoot);
00642   }
00643
00644   // ////////////////////////////////////////////////////////////////////
00645   std::string AIRINV_Service::
00646   jsonHandler (const stdair::JSONString& iJSONString) const {
00647
```

```
00648     //
00649     // Extract from the JSON-ified string the command
00650     //
00651     stdair::JSonCommand::EN_JSonCommand lEN_JSonCommand;
00652     const bool hasCommandBeenRetrieved =
00653       stdair::BomJSONImport::jsonImportCommand (iJSONString,
00654                                                 lEN_JSonCommand);
00655
00656     if (hasCommandBeenRetrieved == false) {
00657       // Return an error JSON-ified string
00658       std::ostringstream oErrorStream;
00659       oErrorStream << "{\"error\": \"Wrong JSON-ified string: "
00660                    << "the command is not understood.\"}";
00661       return oErrorStream.str();
00662     }
00663     assert (hasCommandBeenRetrieved == true);
00664
00665     //
00666     // Extract from the JSON-ified string an airline code
00667     //
00668     stdair::AirlineCode_T lAirlineCode;
00669     const bool hasKeyBeenRetrieved =
00670       stdair::BomJSONImport::jsonImportInventoryKey (iJSONString,
00671                                                      lAirlineCode);
00672
00673     if (hasKeyBeenRetrieved == false) {
00674       // Return an error JSON-ified string
00675       std::ostringstream oErrorStream;
00676       oErrorStream << "{\"error\": \"Wrong JSON-ified string: "
00677                    << "the inventory key is not understood.\"}";
00678       return oErrorStream.str();
00679     }
00680     assert (hasKeyBeenRetrieved == true);
00681
00682     //
00683     // Extract from the JSON-ified string a flight number
00684     //
00685     stdair::FlightNumber_T lFlightNumber;
00686     const bool hasFlightNumBeenRetrieved =
00687       stdair::BomJSONImport::jsonImportFlightNumber (iJSONString,
00688                                                      lFlightNumber);
00689
00690     if (hasFlightNumBeenRetrieved == false) {
00691       // Return an error JSON-ified string
00692       std::ostringstream oErrorStream;
00693       oErrorStream << "{\"error\": \"Wrong JSON-ified string: "
00694                    << "the flight number is not understood.\"}";
00695       return oErrorStream.str();
00696     }
00697     assert (hasFlightNumBeenRetrieved == true);
00698
00699     switch (lEN_JSonCommand) {
00700     case stdair::JSonCommand::FLIGHT_DATE:{
00701
00702       // Extract from the JSON-ified string a flight date
00703       stdair::Date_T lDate;
00704       const bool hasDateBeenRetrieved =
00705         stdair::BomJSONImport::jsonImportFlightDate (iJSONString,
00706                                                      lDate);
00707
00708       if (hasDateBeenRetrieved == false) {
00709         // Return an error JSON-ified string
00710         std::ostringstream oErrorStream;
00711         oErrorStream << "{\"error\": \"Wrong JSON-ified string: "
00712                      << "the flight date is not understood.\"}";
00713         return oErrorStream.str();
00714       }
00715
00716       // DEBUG
00717       STDAIR_LOG_DEBUG ("=> airline code = '" << lAirlineCode
00718                         << "', flight number = " << lFlightNumber
00719                         << "', departure date = '" << lDate << "'");
00720
00721       // DEBUG: Display the flight-date details dump
00722       const std::string& lFlightDateDetailsCSVDump =
00723         csvDisplay (lAirlineCode, lFlightNumber, lDate);
00724       STDAIR_LOG_DEBUG (std::endl << lFlightDateDetailsCSVDump);
00725
00726       // Dump the full details of the flight-date into the JSON-ified
    flight-date
00727       const std::string& lFlightDateDetailsJSONDump =
00728         jsonExportFlightDateObjects (lAirlineCode,
00729                                      lFlightNumber,
00730                                      lDate);
00731
00732       // DEBUG
00733       STDAIR_LOG_DEBUG ("Send: '" << lFlightDateDetailsJSONDump << "'");
```

```
00734
00735       return lFlightDateDetailsJSONDump;
00736       break;
00737     }
00738     case stdair::JSonCommand::LIST:{
00739
00740       // DEBUG
00741       STDAIR_LOG_DEBUG ("=> airline code = '" << lAirlineCode
00742                         << "', flight number = " << lFlightNumber << "'");
00743
00744       // DEBUG: Display the flight-date list dump
00745       const std::string& lFlightDateListCSVDump =
00746         list (lAirlineCode, lFlightNumber);
00747       STDAIR_LOG_DEBUG (std::endl << lFlightDateListCSVDump);
00748
00749       // Dump the full list of the flight-date into the JSON-ified flight-date
00750       const std::string& lFlightDateListJSONDump =
00751         jsonExportFlightDateList (lAirlineCode,
      lFlightNumber);
00752
00753       // DEBUG
00754       STDAIR_LOG_DEBUG ("Send: '" << lFlightDateListCSVDump << "'");
00755
00756       return lFlightDateListJSONDump;
00757       break;
00758     }
00759     default: {
00760       // Return an Error string
00761       std::ostringstream lErrorCmdMessage;
00762       const std::string& lCommandStr =
00763         stdair::JSonCommand::getLabel(lEN_JSonCommand);
00764       lErrorCmdMessage << "{\"error\": \"The command '" << lCommandStr
00765                        << "' is not handled by the AirInv service.\"}";
00766       return lErrorCmdMessage.str();
00767       break;
00768     }
00769     }
00770     // Return an error JSON-ified string
00771     assert (false);
00772     std::string lJSONDump ("{\"error\": \"Wrong JSON-ified string\"}");
00773     return lJSONDump;
00774   }
00775
00776   // ////////////////////////////////////////////////////////////////////
00777   std::string AIRINV_Service::
00778   jsonExportFlightDateList (const
      stdair::AirlineCode_T& iAirlineCode,
00779                             const stdair::FlightNumber_T& iFlightNumber) const
      {
00780
00781     // Retrieve the AIRINV service context
00782     if (_airinvServiceContext == NULL) {
00783       throw stdair::NonInitialisedServiceException ("The AirInv service "
00784                                                     "has not been initialised")
      ;
00785     }
00786     assert (_airinvServiceContext != NULL);
00787
00788     AIRINV_ServiceContext& lAIRINV_ServiceContext = *
      _airinvServiceContext;
00789
00790     // Retrieve the STDAIR service object from the (AIRINV) service context
00791     stdair::STDAIR_Service& lSTDAIR_Service =
00792       lAIRINV_ServiceContext.getSTDAIR_Service();
00793
00794     // Delegate the JSON export to the dedicated service
00795     return lSTDAIR_Service.jsonExportFlightDateList (iAirlineCode,
00796                                                      iFlightNumber);
00797   }
00798
00799   // ////////////////////////////////////////////////////////////////////
00800   std::string AIRINV_Service::
00801   jsonExportFlightDateObjects (const
      stdair::AirlineCode_T& iAirlineCode,
00802                                const stdair::FlightNumber_T& iFlightNumber,
00803                                const stdair::Date_T& iDepartureDate) const {
00804
00805     // Retrieve the AIRINV service context
00806     if (_airinvServiceContext == NULL) {
00807       throw stdair::NonInitialisedServiceException ("The AirInv service "
00808                                                     "has not been initialised")
      ;
00809     }
00810     assert (_airinvServiceContext != NULL);
00811
00812     AIRINV_ServiceContext& lAIRINV_ServiceContext = *
      _airinvServiceContext;
```

```
00813
00814     // Retrieve the STDAIR service object from the (AIRINV) service context
00815     stdair::STDAIR_Service& lSTDAIR_Service =
00816       lAIRINV_ServiceContext.getSTDAIR_Service();
00817
00818     // Delegate the JSON export to the dedicated service
00819     return lSTDAIR_Service.jsonExportFlightDateObjects (iAirlineCode,
00820                                                         iFlightNumber,
00821                                                         iDepartureDate);
00822   }
00823
00824   // ////////////////////////////////////////////////////////////////////
00825   std::string AIRINV_Service::
00826   list (const stdair::AirlineCode_T& iAirlineCode,
00827         const stdair::FlightNumber_T& iFlightNumber) const {
00828     std::ostringstream oFlightListStr;
00829
00830     if (_airinvServiceContext == NULL) {
00831       throw stdair::NonInitialisedServiceException ("The AirInv service "
00832                                                     "has not been initialised")
      ;
00833     }
00834     assert (_airinvServiceContext != NULL);
00835     AIRINV_ServiceContext& lAIRINV_ServiceContext = *
      _airinvServiceContext;
00836
00837     // \todo Check that the current AIRINV_Service is actually operating for
00838     //       the given airline
00839
00840     // Retrieve the STDAIR service object from the (AirInv) service context
00841     stdair::STDAIR_Service& lSTDAIR_Service =
00842       lAIRINV_ServiceContext.getSTDAIR_Service();
00843
00844     // Delegate the BOM display to the dedicated service
00845     return lSTDAIR_Service.list (iAirlineCode, iFlightNumber);
00846   }
00847
00848   // ////////////////////////////////////////////////////////////////////
00849   bool AIRINV_Service::
00850   check (const stdair::AirlineCode_T& iAirlineCode,
00851          const stdair::FlightNumber_T& iFlightNumber,
00852          const stdair::Date_T& iDepartureDate) const {
00853     std::ostringstream oFlightListStr;
00854
00855     if (_airinvServiceContext == NULL) {
00856       throw stdair::NonInitialisedServiceException ("The AirInv service "
00857                                                     "has not been initialised")
      ;
00858     }
00859     assert (_airinvServiceContext != NULL);
00860     AIRINV_ServiceContext& lAIRINV_ServiceContext = *
      _airinvServiceContext;
00861
00862     // \todo Check that the current AIRINV_Service is actually operating for
00863     //       the given airline
00864
00865     // Retrieve the STDAIR service object from the (AirInv) service context
00866     stdair::STDAIR_Service& lSTDAIR_Service =
00867       lAIRINV_ServiceContext.getSTDAIR_Service();
00868
00869     // Delegate the BOM display to the dedicated service
00870     return lSTDAIR_Service.check (iAirlineCode, iFlightNumber, iDepartureDate);
00871   }
00872
00873   // ////////////////////////////////////////////////////////////////////
00874   std::string AIRINV_Service::csvDisplay() const {
00875
00876     // Retrieve the AIRINV service context
00877     if (_airinvServiceContext == NULL) {
00878       throw stdair::NonInitialisedServiceException ("The AirInv service "
00879                                                     "has not been initialised")
      ;
00880     }
00881     assert (_airinvServiceContext != NULL);
00882
00883     AIRINV_ServiceContext& lAIRINV_ServiceContext = *
      _airinvServiceContext;
00884
00885     // Retrieve the STDAIR service object from the (AirInv) service context
00886     stdair::STDAIR_Service& lSTDAIR_Service =
00887       lAIRINV_ServiceContext.getSTDAIR_Service();
00888     const stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
00889
00890     // Delegate the BOM display to the dedicated service
00891     return lSTDAIR_Service.csvDisplay(lBomRoot);
00892   }
00893
```

```
00894   // ////////////////////////////////////////////////////////////////////
00895   std::string AIRINV_Service::
00896   csvDisplay (const stdair::AirlineCode_T& iAirlineCode,
00897               const stdair::FlightNumber_T& iFlightNumber,
00898               const stdair::Date_T& iDepartureDate) const {
00899
00900     // Retrieve the AIRINV service context
00901     if (_airinvServiceContext == NULL) {
00902       throw stdair::NonInitialisedServiceException ("The AirInv service "
00903                                                     "has not been initialised")
      ;
00904     }
00905     assert (_airinvServiceContext != NULL);
00906
00907     AIRINV_ServiceContext& lAIRINV_ServiceContext = *
      _airinvServiceContext;
00908
00909     // Retrieve the STDAIR service object from the (AirInv) service context
00910     stdair::STDAIR_Service& lSTDAIR_Service =
00911       lAIRINV_ServiceContext.getSTDAIR_Service();
00912
00913     // Delegate the BOM display to the dedicated service
00914     return lSTDAIR_Service.csvDisplay (iAirlineCode, iFlightNumber,
00915                                        iDepartureDate);
00916   }
00917
00918   // ////////////////////////////////////////////////////////////////////
00919   stdair::RMEventList_T AIRINV_Service::
00920   initRMEvents (const stdair::Date_T& iStartDate,
00921                 const stdair::Date_T& iEndDate) {
00922
00923     if (_airinvServiceContext == NULL) {
00924       throw stdair::NonInitialisedServiceException ("The AirInv service "
00925                                                     "has not been initialised")
      ;
00926     }
00927     assert (_airinvServiceContext != NULL);
00928     AIRINV_ServiceContext& lAIRINV_ServiceContext = *
      _airinvServiceContext;
00929
00930     // \todo Retrieve the corresponding inventory
00931     stdair::STDAIR_Service& lSTDAIR_Service =
00932       lAIRINV_ServiceContext.getSTDAIR_Service();
00933     stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
00934
00935     stdair::RMEventList_T oRMEventList;
00936     const stdair::InventoryList_T& lInventoryList =
00937       stdair::BomManager::getList<stdair::Inventory> (lBomRoot);
00938     for (stdair::InventoryList_T::const_iterator itInv = lInventoryList.begin()
      ;
00939          itInv != lInventoryList.end(); ++itInv) {
00940       const stdair::Inventory* lInv_ptr = *itInv;
00941       assert (lInv_ptr != NULL);
00942
00943       InventoryManager::initRMEvents (*lInv_ptr,
      oRMEventList,
00944                                       iStartDate, iEndDate);
00945     }
00946
00947     return oRMEventList;
00948   }
00949
00950   // ////////////////////////////////////////////////////////////////////
00951   void AIRINV_Service::
00952   calculateAvailability (stdair::TravelSolutionStruct&
      ioTravelSolution) {
00953
00954     if (_airinvServiceContext == NULL) {
00955       throw stdair::NonInitialisedServiceException ("The AirInv service "
00956                                                     "has not been initialised")
      ;
00957     }
00958     assert (_airinvServiceContext != NULL);
00959     AIRINV_ServiceContext& lAIRINV_ServiceContext = *
      _airinvServiceContext;
00960
00961     // Retrieve the corresponding inventory.
00962     stdair::STDAIR_Service& lSTDAIR_Service =
00963       lAIRINV_ServiceContext.getSTDAIR_Service();
00964     stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
00965
00966     // Delegate the booking to the dedicated command
00967     stdair::BasChronometer lAvlChronometer;
00968     lAvlChronometer.start();
00969     InventoryManager::calculateAvailability
       (lBomRoot, ioTravelSolution);
00970     // const double lAvlMeasure = lAvlChronometer.elapsed();
```

```
00971
00972     // DEBUG
00973     // STDAIR_LOG_DEBUG ("Availability retrieval: " << lAvlMeasure << " - "
00974     //                   << lAIRINV_ServiceContext.display());
00975   }
00976
00977   // ////////////////////////////////////////////////////////////////////
00978   bool AIRINV_Service::sell (const std::string&
      iSegmentDateKey,
00979                              const stdair::ClassCode_T& iClassCode,
00980                              const stdair::PartySize_T& iPartySize) {
00981     bool isSellSuccessful = false;
00982
00983     if (_airinvServiceContext == NULL) {
00984       throw stdair::NonInitialisedServiceException ("The AirInv service "
00985                                                     "has not been initialised")
      ;
00986     }
00987     assert (_airinvServiceContext != NULL);
00988     AIRINV_ServiceContext& lAIRINV_ServiceContext = *
      _airinvServiceContext;
00989
00990     // \todo Check that the current AIRINV_Service is actually operating for
00991     //       the given airline (inventory key)
00992     // Retrieve the corresponding inventory key
00993     const stdair::InventoryKey& lInventoryKey =
00994       stdair::BomKeyManager::extractInventoryKey (iSegmentDateKey);
00995
00996     // Retrieve the root of the BOM tree
00997     stdair::STDAIR_Service& lSTDAIR_Service =
00998       lAIRINV_ServiceContext.getSTDAIR_Service();
00999     stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
01000
01001     // Retrieve the corresponding inventory
01002     stdair::Inventory& lInventory = stdair::BomManager::
01003       getObject<stdair::Inventory> (lBomRoot, lInventoryKey.toString());
01004
01005     // Delegate the booking to the dedicated command
01006     stdair::BasChronometer lSellChronometer; lSellChronometer.start();
01007     isSellSuccessful = InventoryManager::sell (lInventory
      , iSegmentDateKey,
01008                                                iClassCode, iPartySize);
01009     // const double lSellMeasure = lSellChronometer.elapsed();
01010
01011     // DEBUG
01012     // STDAIR_LOG_DEBUG ("Booking sell: " << lSellMeasure << " - "
01013     //                   << lAIRINV_ServiceContext.display());
01014
01015     return isSellSuccessful;
01016   }
01017
01018   // ////////////////////////////////////////////////////////////////////
01019   bool AIRINV_Service::sell (const stdair::BookingClassID_T
      & iClassID,
01020                              const stdair::PartySize_T& iPartySize) {
01021     bool isSellSuccessful = false;
01022
01023     // Delegate the booking to the dedicated command
01024     stdair::BasChronometer lSellChronometer; lSellChronometer.start();
01025     isSellSuccessful = InventoryManager::sell (iClassID,
      iPartySize);
01026     // const double lSellMeasure = lSellChronometer.elapsed();
01027
01028     // DEBUG
01029     // STDAIR_LOG_DEBUG ("Booking sell: " << lSellMeasure << " - "
01030     //                   << lAIRINV_ServiceContext.display());
01031
01032     return isSellSuccessful;
01033   }
01034
01035   // ////////////////////////////////////////////////////////////////////
01036   bool AIRINV_Service::cancel (const std::string&
      iSegmentDateKey,
01037                              const stdair::ClassCode_T& iClassCode,
01038                              const stdair::PartySize_T& iPartySize) {
01039     bool isCancellationSuccessful = false;
01040
01041     if (_airinvServiceContext == NULL) {
01042       throw stdair::NonInitialisedServiceException ("The AirInv service "
01043                                                     "has not been initialised")
      ;
01044     }
01045     assert (_airinvServiceContext != NULL);
01046     AIRINV_ServiceContext& lAIRINV_ServiceContext = *
      _airinvServiceContext;
01047
01048     // \todo Check that the current AIRINV_Service is actually operating for
```

```
01049     //       the given airline (inventory key)
01050     // Retrieve the corresponding inventory key
01051     const stdair::InventoryKey& lInventoryKey =
01052       stdair::BomKeyManager::extractInventoryKey (iSegmentDateKey);
01053
01054     // Retrieve the root of the BOM tree
01055     stdair::STDAIR_Service& lSTDAIR_Service =
01056       lAIRINV_ServiceContext.getSTDAIR_Service();
01057     stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
01058
01059     // Retrieve the corresponding inventory
01060     stdair::Inventory& lInventory = stdair::BomManager::
01061       getObject<stdair::Inventory> (lBomRoot, lInventoryKey.toString());
01062
01063     // Delegate the booking to the dedicated command
01064     stdair::BasChronometer lCancellationChronometer;
01065     lCancellationChronometer.start();
01066     isCancellationSuccessful = InventoryManager::cancel
      (lInventory,
01067                                                          iSegmentDateKey,
01068                                                          iClassCode,iPartySize)
      ;
01069     // const double lCancellationMeasure = lCancellationChronometer.elapsed();
01070
01071     // DEBUG
01072     // STDAIR_LOG_DEBUG ("Booking cancellation: "
01073     //                   << lCancellationMeasure << " - "
01074     //                   << lAIRINV_ServiceContext.display());
01075
01076     return isCancellationSuccessful;
01077   }
01078
01079   // ////////////////////////////////////////////////////////////////////
01080   bool AIRINV_Service::cancel (const
      stdair::BookingClassID_T& iClassID,
01081                                const stdair::PartySize_T& iPartySize) {
01082     bool isCancelSuccessful = false;
01083
01084     // Delegate the booking to the dedicated command
01085     stdair::BasChronometer lCancelChronometer; lCancelChronometer.start();
01086     isCancelSuccessful = InventoryManager::cancel (
      iClassID, iPartySize);
01087     // const double lCancelMeasure = lCancelChronometer.elapsed();
01088
01089     // DEBUG
01090     // STDAIR_LOG_DEBUG ("Booking cancel: " << lCancelMeasure << " - "
01091     //                   << lAIRINV_ServiceContext.display());
01092
01093     return isCancelSuccessful;
01094   }
01095
01096   // ////////////////////////////////////////////////////////////////////
01097   void AIRINV_Service::takeSnapshots (const
      stdair::AirlineCode_T& iAirlineCode,
01098                                       const stdair::DateTime_T& iSnapshotTime)
      {
01099
01100     if (_airinvServiceContext == NULL) {
01101       throw stdair::NonInitialisedServiceException ("The AirInv service "
01102                                                     "has not been initialised")
      ;
01103     }
01104     assert (_airinvServiceContext != NULL);
01105     AIRINV_ServiceContext& lAIRINV_ServiceContext = *
      _airinvServiceContext;
01106
01107     // TODO: Retrieve the corresponding inventory.
01108     stdair::STDAIR_Service& lSTDAIR_Service =
01109       lAIRINV_ServiceContext.getSTDAIR_Service();
01110     stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
01111
01112     const stdair::InventoryList_T lInventoryList =
01113       stdair::BomManager::getList<stdair::Inventory> (lBomRoot);
01114     for (stdair::InventoryList_T::const_iterator itInv = lInventoryList.begin()
      ;
01115          itInv != lInventoryList.end(); ++itInv) {
01116       const stdair::Inventory* lInv_ptr = *itInv;
01117       assert (lInv_ptr != NULL);
01118
01119       InventoryManager::takeSnapshots (*lInv_ptr
      , iSnapshotTime);
01120     }
01121   }
01122
01123   // ////////////////////////////////////////////////////////////////////
01124   void AIRINV_Service::optimise (const
      stdair::AirlineCode_T& iAirlineCode,
```

```
01125                                 const stdair::KeyDescription_T& iFDDescription
      ,
01126                                 const stdair::DateTime_T& iRMEventTime) {
01127
01128     if (_airinvServiceContext == NULL) {
01129       throw stdair::NonInitialisedServiceException ("The AirInv service "
01130                                                     "has not been initialised")
      ;
01131     }
01132     assert (_airinvServiceContext != NULL);
01133     AIRINV_ServiceContext& lAIRINV_ServiceContext = *
      _airinvServiceContext;
01134
01135     // Retrieve the corresponding inventory & flight-date
01136     stdair::STDAIR_Service& lSTDAIR_Service =
01137       lAIRINV_ServiceContext.getSTDAIR_Service();
01138     stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
01139     stdair::Inventory* lInventory_ptr =
01140       stdair::BomManager::getObjectPtr<stdair::Inventory> (lBomRoot,
      iAirlineCode);
01141     if (lInventory_ptr == NULL) {
01142       std::ostringstream oErrorMessage;
01143       oErrorMessage << "The Inventory '" << iAirlineCode
01144                     << "', can not be retrieved.";
01145       STDAIR_LOG_ERROR (oErrorMessage.str());
01146       throw InventoryNotFoundException (oErrorMessage
      .str());
01147     }
01148     assert (lInventory_ptr != NULL);
01149     stdair::FlightDate* lFlightDate_ptr =
01150       stdair::BomManager::getObjectPtr<stdair::FlightDate> (*lInventory_ptr,
01151                                                             iFDDescription);
01152     if (lFlightDate_ptr == NULL) {
01153       std::ostringstream oErrorMessage;
01154       oErrorMessage << "The flight date '" << iFDDescription
01155                     << "', can not be retrieved in the '"
01156                     << iAirlineCode << "' inventory.";
01157       STDAIR_LOG_ERROR (oErrorMessage.str());
01158       throw FlightDateNotFoundException (
      oErrorMessage.str());
01159     }
01160     assert (lFlightDate_ptr != NULL);
01161
01162     const stdair::UnconstrainingMethod& lUnconstrainingMethod =
01163       lInventory_ptr->getUnconstrainingMethod();
01164     const stdair::ForecastingMethod& lForecastingMethod =
01165       lInventory_ptr->getForecastingMethod();
01166     const stdair::PreOptimisationMethod& lPreOptimisationMethod =
01167       lInventory_ptr->getPreOptimisationMethod();
01168     const stdair::OptimisationMethod& lOptimisationMethod =
01169       lInventory_ptr->getOptimisationMethod();
01170     const stdair::PartnershipTechnique& lPartnershipTechnique =
01171       lInventory_ptr->getPartnershipTechnique();
01172
01173     // Retrieve the RMOL service.
01174     RMOL::RMOL_Service& lRMOL_Service = lAIRINV_ServiceContext.getRMOL_Service(
      );
01175
01176     // Optimise the flight-date.
01177     bool isOptimised = lRMOL_Service.optimise (*lFlightDate_ptr, iRMEventTime,
01178                                                lUnconstrainingMethod,
01179                                                lForecastingMethod,
01180                                                lPreOptimisationMethod,
01181                                                lOptimisationMethod,
01182                                                lPartnershipTechnique);
01183
01184     const stdair::OptimisationMethod::EN_OptimisationMethod&
      lENOptimisationMethod = lOptimisationMethod.getMethod();
01185     const bool isEMSRb =
01186       (lENOptimisationMethod == stdair::OptimisationMethod::LEG_BASED_EMSR_B);
01187     // Update the inventory with the new controls.
01188     // updateBookingControls uses bid price vector to set
01189     // the authorization level. But EMSRb sets directly the
01190     // authorization level and does not compute the bid price vector.
01191     // So if EMSRb is used, do not call updateBookingControls.
01192     if (isOptimised == true && isEMSRb == false) {
01193       InventoryManager::updateBookingControls (*lFlightDate_ptr);
01194       InventoryManager::recalculateAvailability (*lFlightDate_ptr);
01195     }
01196   }
01197 }
```

23.247 airinv/service/AIRINV_ServiceContext.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <airinv/basic/BasConst_AIRINV_Service.hpp>
#include <airinv/service/AIRINV_ServiceContext.hpp>
```

Namespaces

- namespace AIRINV

23.248 AIRINV_ServiceContext.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // AirInv
00008 #include <airinv/basic/BasConst_AIRINV_Service.hpp
      >
00009 #include <airinv/service/AIRINV_ServiceContext.hpp
      >
00010
00011 namespace AIRINV {
00012
00013   // ////////////////////////////////////////////////////////////////////
00014   AIRINV_ServiceContext::AIRINV_ServiceContext()
00015     : _ownStdairService (false), _ownSEVMGRService (true),
00016       _airlineCode (DEFAULT_AIRLINE_CODE) {
00017   }
00018
00019   // ////////////////////////////////////////////////////////////////////
00020   AIRINV_ServiceContext::
00021   AIRINV_ServiceContext (const stdair::AirlineCode_T& iAirlineCode)
00022     : _ownStdairService (false), _ownSEVMGRService (true),
00023       _airlineCode (iAirlineCode) {
00024   }
00025
00026   // ////////////////////////////////////////////////////////////////////
00027   AIRINV_ServiceContext::AIRINV_ServiceContext (const AIRINV_ServiceContext&)
00028     : _ownStdairService (false), _ownSEVMGRService (true),
00029       _airlineCode (DEFAULT_AIRLINE_CODE) {
00030   }
00031
00032   // ////////////////////////////////////////////////////////////////////
00033   AIRINV_ServiceContext::~AIRINV_ServiceContext() {
00034   }
00035
00036   // ////////////////////////////////////////////////////////////////////
00037   const std::string AIRINV_ServiceContext::shortDisplay() const {
00038     std::ostringstream oStr;
00039     oStr << "AIRINV_ServiceContext[" << _airlineCode
00040          << "] -- Owns StdAir service: " << _ownStdairService;
00041     return oStr.str();
00042   }
00043
00044   // ////////////////////////////////////////////////////////////////////
00045   const std::string AIRINV_ServiceContext::display() const {
00046     std::ostringstream oStr;
00047     oStr << shortDisplay();
00048     return oStr.str();
00049   }
00050
00051   // ////////////////////////////////////////////////////////////////////
00052   const std::string AIRINV_ServiceContext::describe() const {
00053     return shortDisplay();
00054   }
00055
00056   // ////////////////////////////////////////////////////////////////////
00057   void AIRINV_ServiceContext::reset() {
00058
00059     // The shared_ptr<>::reset() method drops the refcount by one.
00060     // If the count result is dropping to zero, the resource pointed to
00061     // by the shared_ptr<> will be freed.
00062
```

```
00063     // Reset the stdair shared pointer
00064     _stdairService.reset();
00065
00066     // Reset the rmol shared pointer
00067     _rmolService.reset();
00068
00069     // Reset the airrac shared pointer
00070     _airracService.reset();
00071
00072     // Reset the sevmgr shared pointer
00073     _sevmgrService.reset();
00074   }
00075
00076 }
```

23.249 airinv/service/AIRINV_ServiceContext.hpp File Reference

```
#include <string>
#include <boost/shared_ptr.hpp>
#include <stdair/stdair_service_types.hpp>
#include <stdair/service/ServiceAbstract.hpp>
#include <rmol/RMOL_Types.hpp>
#include <airrac/AIRRAC_Types.hpp>
#include <sevmgr/SEVMGR_Types.hpp>
#include <airinv/AIRINV_Types.hpp>
```

Classes

- class AIRINV::AIRINV_ServiceContext

  Class holding the context of the AirInv services.

Namespaces

- namespace AIRINV

23.250 AIRINV_ServiceContext.hpp

```
00001 #ifndef __AIRINV_SVC_AIRINVSERVICECONTEXT_HPP
00002 #define __AIRINV_SVC_AIRINVSERVICECONTEXT_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // Boost
00010 #include <boost/shared_ptr.hpp>
00011 // StdAir
00012 #include <stdair/stdair_service_types.hpp>
00013 #include <stdair/service/ServiceAbstract.hpp>
00014 // RMOL
00015 #include <rmol/RMOL_Types.hpp>
00016 // AIRRAC
00017 #include <airrac/AIRRAC_Types.hpp>
00018 // SEvMgr
00019 #include <sevmgr/SEVMGR_Types.hpp>
00020 // AirInv
00021 #include <airinv/AIRINV_Types.hpp>
00022
00023 namespace AIRINV {
00024
00028   class AIRINV_ServiceContext : public
      stdair::ServiceAbstract {
00034     friend class AIRINV_Service;
00035     friend class FacAirinvServiceContext;
00036
00037   private:
00038     // ////////////////// Getters ////////////////////
00042     stdair::AirlineCode_T getAirlineCode() const {
```

```
00043       return _airlineCode;
00044     }
00045
00049     stdair::STDAIR_ServicePtr_T getSTDAIR_ServicePtr() const {
00050       return _stdairService;
00051     }
00052
00056     stdair::STDAIR_Service& getSTDAIR_Service() const {
00057       assert (_stdairService != NULL);
00058       return *_stdairService;
00059     }
00060
00064     const bool getOwnStdairServiceFlag() const {
00065       return _ownStdairService;
00066     }
00067
00071     RMOL::RMOL_Service& getRMOL_Service() const {
00072       assert (_rmolService != NULL);
00073       return *_rmolService;
00074     }
00075
00079     AIRRAC::AIRRAC_Service& getAIRRAC_Service() const {
00080       assert (_airracService != NULL);
00081       return *_airracService;
00082     }
00083
00087     SEVMGR::SEVMGR_ServicePtr_T getSEVMGR_ServicePtr() const {
00088       return _sevmgrService;
00089     }
00090
00094     SEVMGR::SEVMGR_Service& getSEVMGR_Service() const {
00095       assert (_sevmgrService != NULL);
00096       return *_sevmgrService;
00097     }
00098
00102     const bool getOwnSEVMGRServiceFlag() const {
00103       return _ownSEVMGRService;
00104     }
00105
00106
00107   private:
00108     // ////////////////// Setters ////////////////////
00112     void setAirlineCode (const stdair::AirlineCode_T& iAirlineCode) {
00113       _airlineCode = iAirlineCode;
00114     }
00115
00119     void setSTDAIR_Service (stdair::STDAIR_ServicePtr_T ioSTDAIR_ServicePtr,
00120                             const bool iOwnStdairService) {
00121       _stdairService = ioSTDAIR_ServicePtr;
00122       _ownStdairService = iOwnStdairService;
00123     }
00124
00128     void setRMOL_Service (RMOL::RMOL_ServicePtr_T ioRMOL_ServicePtr) {
00129       _rmolService = ioRMOL_ServicePtr;
00130     }
00131
00135     void setAIRRAC_Service (AIRRAC::AIRRAC_ServicePtr_T ioAIRRAC_ServicePtr) {
00136       _airracService = ioAIRRAC_ServicePtr;
00137     }
00138
00142     void setSEVMGR_Service (SEVMGR::SEVMGR_ServicePtr_T ioSEVMGR_ServicePtr,
00143                             const bool iOwnSEVMGRService) {
00144       _sevmgrService = ioSEVMGR_ServicePtr;
00145       _ownSEVMGRService = iOwnSEVMGRService;
00146     }
00147
00148   private:
00149     // ////////////////////// Display Methods /////////////////////
00153     const std::string shortDisplay() const;
00154
00158     const std::string display() const;
00159
00163     const std::string describe() const;
00164
00165
00166   private:
00168
00171     AIRINV_ServiceContext (const stdair::AirlineCode_T&);
00175     AIRINV_ServiceContext();
00179     AIRINV_ServiceContext (const AIRINV_ServiceContext&);
00180
00184     ~AIRINV_ServiceContext();
00185
00189     void reset();
00190
00191
00192   private:
```

```
00193     // ///////////////// Children ///////////////
00197     stdair::STDAIR_ServicePtr_T _stdairService;
00198
00202     bool _ownStdairService;
00203
00207     RMOL::RMOL_ServicePtr_T _rmolService;
00208
00212     SEVMGR::SEVMGR_ServicePtr_T _sevmgrService;
00213
00217     bool _ownSEVMGRService;
00218
00222     AIRRAC::AIRRAC_ServicePtr_T _airracService;
00223
00224   private:
00225     // ////////////// Attributes ////////////
00230     stdair::AirlineCode_T _airlineCode;
00231   };
00232
00233 }
00234 #endif // __AIRINV_SVC_AIRINVSERVICECONTEXT_HPP
```

23.251 airinv/service/ServiceAbstract.cpp File Reference

`#include <airinv/service/ServiceAbstract.hpp>`

Namespaces

- namespace AIRINV

23.252 ServiceAbstract.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // AIRINV
00005 #include <airinv/service/ServiceAbstract.hpp>
00006
00007 namespace AIRINV {
00008
00009 }
```

23.253 airinv/service/ServiceAbstract.hpp File Reference

`#include <iosfwd>`

Classes

- class AIRINV::ServiceAbstract

Namespaces

- namespace AIRINV

Functions

- template<class charT , class traits >
  std::basic_ostream< charT,
  traits > & operator<< (std::basic_ostream< charT, traits > &ioOut, const AIRINV::ServiceAbstract &i-
  Service)

- template<class charT , class traits >
  std::basic_istream< charT,
  traits > & operator>> (std::basic_istream< charT, traits > &ioIn, AIRINV::ServiceAbstract &ioService)

23.253.1 Function Documentation

23.253.1.1 template<class charT , class traits > std::basic_ostream<charT, traits>& operator<< (std::basic_ostream< charT, traits > & *ioOut,* const AIRINV::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 42 of file ServiceAbstract.hpp.

23.253.1.2 template<class charT , class traits > std::basic_istream<charT, traits>& operator>> (std::basic_istream< charT, traits > & *ioIn,* AIRINV::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 70 of file ServiceAbstract.hpp.

References AIRINV::ServiceAbstract::fromStream().

23.254 ServiceAbstract.hpp

```
00001 #ifndef __AIRINV_SVC_SERVICEABSTRACT_HPP
00002 #define __AIRINV_SVC_SERVICEABSTRACT_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 //#include <sstream>
00010
00011 namespace AIRINV {
00012
00014   class ServiceAbstract {
00015   public:
00016
00018     virtual ~ServiceAbstract() {}
00019
00022     virtual void toStream (std::ostream& ioOut) const {}
00023
00026     virtual void fromStream (std::istream& ioIn) {}
00027
00028   protected:
00030     ServiceAbstract() {}
00031   };
00032 }
00033
00039 template <class charT, class traits>
00040 inline
00041 std::basic_ostream<charT, traits>&
00042 operator<< (std::basic_ostream<charT, traits>& ioOut,
00043             const AIRINV::ServiceAbstract& iService) {
00049   std::basic_ostringstream<charT,traits> ostr;
00050   ostr.copyfmt (ioOut);
00051   ostr.width (0);
00052
00053   // Fill string stream
00054   iService.toStream (ostr);
00055
00056   // Print string stream
00057   ioOut << ostr.str();
00058
00059   return ioOut;
00060 }
00061
00067 template <class charT, class traits>
00068 inline
00069 std::basic_istream<charT, traits>&
00070 operator>> (std::basic_istream<charT, traits>& ioIn,
00071             AIRINV::ServiceAbstract& ioService) {
```

```
00072   // Fill Service object with input stream
00073   ioService.fromStream (ioIn);
00074   return ioIn;
00075 }
00076
00077 #endif // __AIRINV_SVC_SERVICEABSTRACT_HPP
```

23.255 airinv/ui/cmdline/airinv.cpp File Reference

23.256 airinv.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/program_options.hpp>
00013 #include <boost/tokenizer.hpp>
00014 #include <boost/regex.hpp>
00015 #include <boost/swap.hpp>
00016 #include <boost/algorithm/string/case_conv.hpp>
00017 // StdAir
00018 #include <stdair/basic/BasLogParams.hpp>
00019 #include <stdair/basic/BasDBParams.hpp>
00020 #include <stdair/service/Logger.hpp>
00021 #include <stdair/stdair_json.hpp>
00022 // GNU Readline Wrapper
00023 #include <stdair/ui/cmdline/SReadline.hpp>
00024 // AirInv
00025 #include <airinv/AIRINV_Master_Service.hpp>
00026 #include <airinv/config/airinv-paths.hpp>
00027
00028 // //////// Constants //////
00032 const std::string K_AIRINV_DEFAULT_LOG_FILENAME ("airinv.log");
00033
00037 const std::string K_AIRINV_DEFAULT_INVENTORY_FILENAME (STDAIR_SAMPLE_DIR
00038                                                        "/invdump01.csv");
00042 const std::string K_AIRINV_DEFAULT_SCHEDULE_FILENAME (STDAIR_SAMPLE_DIR
00043                                                       "/schedule01.csv");
00047 const std::string K_AIRINV_DEFAULT_OND_FILENAME (STDAIR_SAMPLE_DIR
00048                                                  "/ond01.csv");
00052 const std::string K_AIRINV_DEFAULT_FRAT5_INPUT_FILENAME (STDAIR_SAMPLE_DIR
00053                                                         "/frat5.csv");
00057 const std::string K_AIRINV_DEFAULT_FF_DISUTILITY_INPUT_FILENAME (
      STDAIR_SAMPLE_DIR
00058                                                                 "
      /ffDisutility.csv");
00062 const std::string K_AIRINV_DEFAULT_YIELD_FILENAME (STDAIR_SAMPLE_DIR
00063                                                   "/yieldstore01.csv");
00064
00069 const bool K_AIRINV_DEFAULT_BUILT_IN_INPUT = false;
00070
00075 const bool K_AIRINV_DEFAULT_FOR_SCHEDULE = false;
00076
00080 const int K_AIRINV_EARLY_RETURN_STATUS = 99;
00081
00086 typedef std::vector<std::string> TokenList_T;
00087
00091 struct Command_T {
00092   typedef enum {
00093     NOP = 0,
00094     QUIT,
00095     HELP,
00096     LIST,
00097     DISPLAY,
00098     SELECT,
00099     SELL,
00100     JSON_LIST,
00101     JSON_DISPLAY,
00102     LAST_VALUE
00103   } Type_T;
00104 };
00105
00106 // ///////// Parsing of Options & Configuration /////////
00107 // A helper function to simplify the main part.
00108 template<class T> std::ostream& operator<< (std::ostream& os,
00109                                             const std::vector<T>& v) {
00110   std::copy (v.begin(), v.end(), std::ostream_iterator<T> (std::cout, " "));
00111   return os;
00112 }
```

```
00113
00117 int readConfiguration (int argc, char* argv[],
00118                        bool& ioIsBuiltin, bool& ioIsForSchedule,
00119                        stdair::Filename_T& ioInventoryFilename,
00120                        stdair::Filename_T& ioScheduleInputFilename,
00121                        stdair::Filename_T& ioODInputFilename,
00122                        stdair::Filename_T& ioFRAT5Filename,
00123                        stdair::Filename_T& ioFFDisutilityFilename,
00124                        stdair::Filename_T& ioYieldInputFilename,
00125                        std::string& ioLogFilename) {
00126   // Default for the built-in input
00127   ioIsBuiltin = K_AIRINV_DEFAULT_BUILT_IN_INPUT;
00128
00129   // Default for the inventory or schedule option
00130   ioIsForSchedule = K_AIRINV_DEFAULT_FOR_SCHEDULE;
00131
00132   // Declare a group of options that will be allowed only on command line
00133   boost::program_options::options_description generic ("Generic options");
00134   generic.add_options()
00135     ("prefix", "print installation prefix")
00136     ("version,v", "print version string")
00137     ("help,h", "produce help message");
00138
00139   // Declare a group of options that will be allowed both on command
00140   // line and in config file
00141
00142   boost::program_options::options_description config ("Configuration");
00143   config.add_options()
00144     ("builtin,b",
00145      "The sample BOM tree can be either built-in or parsed from an input file.
       That latter must then be given with the -i/--inventory or -s/--schedule option")
00146     ("for_schedule,f",
00147      "The BOM tree should be built from a schedule file (instead of from an
       inventory dump)")
00148     ("inventory,i",
00149      boost::program_options::value< std::string >(&ioInventoryFilename)->
      default_value(K_AIRINV_DEFAULT_INVENTORY_FILENAME),
00150      "(CSV) input file for the inventory")
00151     ("schedule,s",
00152      boost::program_options::value< std::string >(&ioScheduleInputFilename)->
      default_value(K_AIRINV_DEFAULT_SCHEDULE_FILENAME),
00153      "(CSV) input file for the schedule")
00154     ("ond,o",
00155      boost::program_options::value< std::string >(&ioODInputFilename)->
      default_value(K_AIRINV_DEFAULT_OND_FILENAME),
00156      "(CSV) input file for the O&D")
00157     ("frat5,F",
00158      boost::program_options::value< std::string >(&ioFRAT5Filename)->
      default_value(K_AIRINV_DEFAULT_FRAT5_INPUT_FILENAME),
00159      "(CSV) input file for the FRAT5 Curve")
00160     ("ff_disutility,D",
00161      boost::program_options::value< std::string >(&ioFFDisutilityFilename)->
      default_value(K_AIRINV_DEFAULT_FF_DISUTILITY_INPUT_FILENAME),
00162      "(CSV) input file for the FF disutility Curve")
00163     ("yield,y",
00164      boost::program_options::value< std::string >(&ioYieldInputFilename)->
      default_value(K_AIRINV_DEFAULT_YIELD_FILENAME),
00165      "(CSV) input file for the yield")
00166     ("log,l",
00167      boost::program_options::value< std::string >(&ioLogFilename)->
      default_value(K_AIRINV_DEFAULT_LOG_FILENAME),
00168      "Filename for the logs")
00169     ;
00170
00171   // Hidden options, will be allowed both on command line and
00172   // in config file, but will not be shown to the user.
00173   boost::program_options::options_description hidden ("Hidden options");
00174   hidden.add_options()
00175     ("copyright",
00176      boost::program_options::value< std::vector<std::string> >(),
00177      "Show the copyright (license)");
00178
00179   boost::program_options::options_description cmdline_options;
00180   cmdline_options.add(generic).add(config).add(hidden);
00181
00182   boost::program_options::options_description config_file_options;
00183   config_file_options.add(config).add(hidden);
00184   boost::program_options::options_description visible ("Allowed options");
00185   visible.add(generic).add(config);
00186
00187   boost::program_options::positional_options_description p;
00188   p.add ("copyright", -1);
00189
00190   boost::program_options::variables_map vm;
00191   boost::program_options::
00192     store (boost::program_options::command_line_parser (argc, argv).
00193            options (cmdline_options).positional(p).run(), vm);
```

```
00194
00195   std::ifstream ifs ("airinv.cfg");
00196   boost::program_options::store (parse_config_file (ifs, config_file_options),
00197                                  vm);
00198   boost::program_options::notify (vm);
00199
00200   if (vm.count ("help")) {
00201     std::cout << visible << std::endl;
00202     return K_AIRINV_EARLY_RETURN_STATUS;
00203   }
00204
00205   if (vm.count ("version")) {
00206     std::cout << PACKAGE_NAME << ", version " << PACKAGE_VERSION
      << std::endl;
00207     return K_AIRINV_EARLY_RETURN_STATUS;
00208   }
00209
00210   if (vm.count ("prefix")) {
00211     std::cout << "Installation prefix: " << PREFIXDIR << std::endl;
00212     return K_AIRINV_EARLY_RETURN_STATUS;
00213   }
00214
00215   if (vm.count ("builtin")) {
00216     ioIsBuiltin = true;
00217   }
00218   const std::string isBuiltinStr = (ioIsBuiltin == true)?"yes":"no";
00219   std::cout << "The BOM should be built-in? " << isBuiltinStr << std::endl;
00220
00221   if (vm.count ("for_schedule")) {
00222     ioIsForSchedule = true;
00223   }
00224   const std::string isForScheduleStr = (ioIsForSchedule == true)?"yes":"no";
00225   std::cout << "The BOM should be built from schedule? " << isForScheduleStr
00226             << std::endl;
00227
00228   if (ioIsBuiltin == false) {
00229
00230     if (ioIsForSchedule == false) {
00231       // The BOM tree should be built from parsing an inventory dump
00232       if (vm.count ("inventory")) {
00233         ioInventoryFilename = vm["inventory"].as< std::string >();
00234         std::cout << "Input inventory filename is: " << ioInventoryFilename
00235                   << std::endl;
00236
00237       } else {
00238         // The built-in option is not selected. However, no inventory dump
00239         // file is specified
00240         std::cerr << "Either one among the -b/--builtin, -i/--inventory or "
00241                   << " -f/--for_schedule and -s/--schedule options "
00242                   << "must be specified" << std::endl;
00243       }
00244
00245     } else {
00246       // The BOM tree should be built from parsing a schedule (and O&D) file
00247       if (vm.count ("schedule")) {
00248         ioScheduleInputFilename = vm["schedule"].as< std::string >();
00249         std::cout << "Input schedule filename is: " << ioScheduleInputFilename
00250                   << std::endl;
00251
00252       } else {
00253         // The built-in option is not selected. However, no schedule file
00254         // is specified
00255         std::cerr << "Either one among the -b/--builtin, -i/--inventory or "
00256                   << " -f/--for_schedule and -s/--schedule options "
00257                   << "must be specified" << std::endl;
00258       }
00259
00260       if (vm.count ("ond")) {
00261         ioODInputFilename = vm["ond"].as< std::string >();
00262         std::cout << "Input O&D filename is: " << ioODInputFilename <<
      std::endl;
00263       }
00264
00265       if (vm.count ("frat5")) {
00266         ioFRAT5Filename = vm["frat5"].as< std::string >();
00267         std::cout << "FRAT5 input filename is: " << ioFRAT5Filename <<
      std::endl;
00268
00269       }
00270
00271       if (vm.count ("ff_disutility")) {
00272         ioFFDisutilityFilename = vm["ff_disutility"].as< std::string >();
00273         std::cout << "FF disutility input filename is: "
00274                   << ioFFDisutilityFilename << std::endl;
00275
00276       }
00277
```

```
00278       if (vm.count ("yield")) {
00279         ioYieldInputFilename = vm["yield"].as< std::string >();
00280         std::cout << "Input yield filename is: " << ioYieldInputFilename <<
      std::endl;
00281       }
00282     }
00283   }
00284
00285   if (vm.count ("log")) {
00286     ioLogFilename = vm["log"].as< std::string >();
00287     std::cout << "Log filename is: " << ioLogFilename << std::endl;
00288   }
00289
00290   return 0;
00291 }
00292
00293 // //////////////////////////////////////////////////////////////////
00294 void initReadline (swift::SReadline& ioInputReader) {
00295
00296   // Prepare the list of my own completers
00297   std::vector<std::string> Completers;
00298
00299   // The following is supported:
00300   // - "identifiers"
00301   // - special identifier %file - means to perform a file name completion
00302   Completers.push_back ("help");
00303   Completers.push_back ("list %airline_code %flight_number");
00304   Completers.push_back ("select %airline_code %flight_number %flight_date");
00305   Completers.push_back ("display");
00306   Completers.push_back ("sell %booking_class %party_size %origin %destination")
      ;
00307   Completers.push_back ("quit");
00308   Completers.push_back ("json_list");
00309   Completers.push_back ("json_display");
00310
00311
00312   // Now register the completers.
00313   // Actually it is possible to re-register another set at any time
00314   ioInputReader.RegisterCompletions (Completers);
00315 }
00316
00317 // //////////////////////////////////////////////////////////////////
00318 Command_T::Type_T extractCommand (TokenList_T& ioTokenList) {
00319   Command_T::Type_T oCommandType = Command_T::LAST_VALUE;
00320
00321   // Interpret the user input
00322   if (ioTokenList.empty() == false) {
00323     TokenList_T::iterator itTok = ioTokenList.begin();
00324     std::string lCommand (*itTok);
00325     boost::algorithm::to_lower (lCommand);
00326
00327     if (lCommand == "help") {
00328       oCommandType = Command_T::HELP;
00329
00330     } else if (lCommand == "list") {
00331       oCommandType = Command_T::LIST;
00332
00333     } else if (lCommand == "display") {
00334       oCommandType = Command_T::DISPLAY;
00335
00336     } else if (lCommand == "select") {
00337       oCommandType = Command_T::SELECT;
00338
00339     } else if (lCommand == "sell") {
00340       oCommandType = Command_T::SELL;
00341
00342     } else if (lCommand == "json_list") {
00343       oCommandType = Command_T::JSON_LIST;
00344
00345     } else if (lCommand == "json_display") {
00346       oCommandType = Command_T::JSON_DISPLAY;
00347
00348     } else if (lCommand == "quit") {
00349       oCommandType = Command_T::QUIT;
00350     }
00351
00352     // Remove the first token (the command), as the corresponding information
00353     // has been extracted in the form of the returned command type enumeration
00354     ioTokenList.erase (itTok);
00355
00356   } else {
00357     oCommandType = Command_T::NOP;
00358   }
00359
00360   return oCommandType;
00361 }
00362
```

```
00363 // ////////////////////////////////////////////////////////////////////
00364 void parseFlightKey (const TokenList_T& iTokenList,
00365                      stdair::AirlineCode_T& ioAirlineCode,
00366                      stdair::FlightNumber_T& ioFlightNumber) {
00367   // Interpret the user input
00368   if (iTokenList.empty() == false) {
00369
00370     // Read the airline code
00371     TokenList_T::const_iterator itTok = iTokenList.begin();
00372     if (itTok->empty() == false) {
00373       ioAirlineCode = *itTok;
00374       boost::algorithm::to_upper (ioAirlineCode);
00375     }
00376
00377     // Read the flight-number
00378     ++itTok;
00379     if (itTok != iTokenList.end()) {
00380
00381       if (itTok->empty() == false) {
00382         try {
00383
00384           ioFlightNumber = boost::lexical_cast<stdair::FlightNumber_T> (*itTok)
      ;
00385
00386         } catch (boost::bad_lexical_cast& eCast) {
00387           std::cerr << "The flight number ('" << *itTok
00388                     << "') cannot be understood. "
00389                     << "The default value (all) is kept."
00390                     << std::endl;
00391           return;
00392         }
00393       }
00394
00395     } else {
00396       return;
00397     }
00398   }
00399 }
00400
00401 // ////////////////////////////////////////////////////////////////////
00402 void parseFlightDateKey (const TokenList_T& iTokenList,
00403                          stdair::AirlineCode_T& ioAirlineCode,
00404                          stdair::FlightNumber_T& ioFlightNumber,
00405                          stdair::Date_T& ioDepartureDate) {
00406   //
00407   const std::string kMonthStr[12] = {"Jan", "Feb", "Mar", "Apr", "May", "Jun",
00408                                      "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"};
00409   //
00410   unsigned short ioDepartureDateYear = ioDepartureDate.year();
00411   unsigned short ioDepartureDateMonth = ioDepartureDate.month();
00412   std::string ioDepartureDateMonthStr = kMonthStr[ioDepartureDateMonth-1];
00413   unsigned short ioDepartureDateDay = ioDepartureDate.day();
00414
00415   // Interpret the user input
00416   if (iTokenList.empty() == false) {
00417
00418     // Read the airline code
00419     TokenList_T::const_iterator itTok = iTokenList.begin();
00420     if (itTok->empty() == false) {
00421       ioAirlineCode = *itTok;
00422       boost::algorithm::to_upper (ioAirlineCode);
00423     }
00424
00425     // Read the flight-number
00426     ++itTok;
00427     if (itTok != iTokenList.end()) {
00428
00429       if (itTok->empty() == false) {
00430         try {
00431
00432           ioFlightNumber = boost::lexical_cast<stdair::FlightNumber_T> (*itTok)
      ;
00433
00434         } catch (boost::bad_lexical_cast& eCast) {
00435           std::cerr << "The flight number ('" << *itTok
00436                     << "') cannot be understood. "
00437                     << "The default value (all) is kept."
00438                     << std::endl;
00439           return;
00440         }
00441       }
00442
00443     } else {
00444       return;
00445     }
00446
00447     // Read the year for the departure date
```

```
00448     ++itTok;
00449     if (itTok != iTokenList.end()) {
00450
00451       if (itTok->empty() == false) {
00452         try {
00453
00454           ioDepartureDateYear = boost::lexical_cast<unsigned short> (*itTok);
00455           if (ioDepartureDateYear < 100) {
00456             ioDepartureDateYear += 2000;
00457           }
00458
00459         } catch (boost::bad_lexical_cast& eCast) {
00460           std::cerr << "The year of the flight departure date ('" << *itTok
00461                     << "') cannot be understood. The default value ("
00462                     << ioDepartureDateYear << ") is kept. " << std::endl;
00463           return;
00464         }
00465       }
00466
00467     } else {
00468       return;
00469     }
00470
00471     // Read the month for the departure date
00472     ++itTok;
00473     if (itTok != iTokenList.end()) {
00474
00475       if (itTok->empty() == false) {
00476         try {
00477
00478           const boost::regex lMonthRegex ("^(\\d{1,2})$");
00479           const bool isMonthANumber = regex_match (*itTok, lMonthRegex);
00480
00481           if (isMonthANumber == true) {
00482             const unsigned short lMonth =
00483               boost::lexical_cast<unsigned short> (*itTok);
00484             if (lMonth > 12) {
00485               throw boost::bad_lexical_cast();
00486             }
00487             ioDepartureDateMonthStr = kMonthStr[lMonth-1];
00488
00489           } else {
00490             const std::string lMonthStr (*itTok);
00491             if (lMonthStr.size() < 3) {
00492               throw boost::bad_lexical_cast();
00493             }
00494             std::string lMonthStr1 (lMonthStr.substr (0, 1));
00495             boost::algorithm::to_upper (lMonthStr1);
00496             std::string lMonthStr23 (lMonthStr.substr (1, 2));
00497             boost::algorithm::to_lower (lMonthStr23);
00498             ioDepartureDateMonthStr = lMonthStr1 + lMonthStr23;
00499           }
00500
00501         } catch (boost::bad_lexical_cast& eCast) {
00502           std::cerr << "The month of the flight departure date ('" << *itTok
00503                     << "') cannot be understood. The default value ("
00504                     << ioDepartureDateMonthStr << ") is kept. " << std::endl;
00505           return;
00506         }
00507       }
00508
00509     } else {
00510       return;
00511     }
00512
00513     // Read the day for the departure date
00514     ++itTok;
00515     if (itTok != iTokenList.end()) {
00516
00517       if (itTok->empty() == false) {
00518         try {
00519
00520           ioDepartureDateDay = boost::lexical_cast<unsigned short> (*itTok);
00521
00522         } catch (boost::bad_lexical_cast& eCast) {
00523           std::cerr << "The day of the flight departure date ('" << *itTok
00524                     << "') cannot be understood. The default value ("
00525                     << ioDepartureDateDay << ") is kept. " << std::endl;
00526           return;
00527         }
00528       }
00529
00530     } else {
00531       return;
00532     }
00533
00534     // Re-compose the departure date
```

```
00535     std::ostringstream lDepartureDateStr;
00536     lDepartureDateStr << ioDepartureDateYear << "-" << ioDepartureDateMonthStr
00537                       << "-" << ioDepartureDateDay;
00538
00539     try {
00540
00541       ioDepartureDate =
00542         boost::gregorian::from_simple_string (lDepartureDateStr.str());
00543
00544     } catch (boost::gregorian::bad_month& eCast) {
00545       std::cerr << "The flight departure date ('" << lDepartureDateStr.str()
00546                 << "') cannot be understood. The default value ("
00547                 << ioDepartureDate << ") is kept. " << std::endl;
00548       return;
00549     }
00550
00551   }
00552 }
00553
00554 // //////////////////////////////////////////////////////////////////
00555 void parseBookingClassKey (const TokenList_T& iTokenList,
00556                            stdair::ClassCode_T& ioBookingClass,
00557                            stdair::PartySize_T& ioPartySize,
00558                            stdair::AirportCode_T& ioOrigin,
00559                            stdair::AirportCode_T& ioDestination) {
00560   // Interpret the user input
00561   if (iTokenList.empty() == false) {
00562
00563     // Read the booking class
00564     TokenList_T::const_iterator itTok = iTokenList.begin();
00565     if (itTok->empty() == false) {
00566       ioBookingClass = *itTok;
00567       boost::algorithm::to_upper (ioBookingClass);
00568     }
00569
00570     // Read the party size
00571     ++itTok;
00572     if (itTok != iTokenList.end()) {
00573
00574       if (itTok->empty() == false) {
00575         try {
00576
00577           ioPartySize = boost::lexical_cast<stdair::PartySize_T> (*itTok);
00578
00579         } catch (boost::bad_lexical_cast& eCast) {
00580           std::cerr << "The party size ('" << *itTok
00581                     << "') cannot be understood. The default value ("
00582                     << ioPartySize << ") is kept." << std::endl;
00583           return;
00584         }
00585       }
00586
00587     } else {
00588       return;
00589     }
00590
00591     // Read the origin
00592     ++itTok;
00593     if (itTok != iTokenList.end()) {
00594
00595       if (itTok->empty() == false) {
00596         ioOrigin = *itTok;
00597         boost::algorithm::to_upper (ioOrigin);
00598       }
00599
00600     } else {
00601       return;
00602     }
00603
00604     // Read the destination
00605     ++itTok;
00606     if (itTok != iTokenList.end()) {
00607
00608       if (itTok->empty() == false) {
00609         ioDestination = *itTok;
00610         boost::algorithm::to_upper (ioDestination);
00611       }
00612
00613     } else {
00614       return;
00615     }
00616   }
00617 }
00618
00619 // //////////////////////////////////////////////////////////////////
00620 std::string toString (const TokenList_T& iTokenList) {
00621   std::ostringstream oStr;
```

```
00622
00623   // Re-create the string with all the tokens, trimmed by read-line
00624   unsigned short idx = 0;
00625   for (TokenList_T::const_iterator itTok = iTokenList.begin();
00626        itTok != iTokenList.end(); ++itTok, ++idx) {
00627     if (idx != 0) {
00628       oStr << " ";
00629     }
00630     oStr << *itTok;
00631   }
00632
00633   return oStr.str();
00634 }
00635
00636 // ////////////////////////////////////////////////////////
00637 TokenList_T extractTokenList (const TokenList_T& iTokenList,
00638                               const std::string& iRegularExpression) {
00639   TokenList_T oTokenList;
00640
00641   // Re-create the string with all the tokens (which had been trimmed
00642   // by read-line)
00643   const std::string lFullLine = toString (iTokenList);
00644
00645   // See the caller for the regular expression
00646   boost::regex expression (iRegularExpression);
00647
00648   std::string::const_iterator start = lFullLine.begin();
00649   std::string::const_iterator end = lFullLine.end();
00650
00651   boost::match_results<std::string::const_iterator> what;
00652   boost::match_flag_type flags = boost::match_default | boost::format_sed;
00653   regex_search (start, end, what, expression, flags);
00654
00655   // Put the matched strings in the list of tokens to be returned back
00656   // to the caller
00657   const unsigned short lMatchSetSize = what.size();
00658   for (unsigned short matchIdx = 1; matchIdx != lMatchSetSize; ++matchIdx) {
00659     const std::string lMatchedString (std::string (what[matchIdx].first,
00660                                                    what[matchIdx].second));
00661     //if (lMatchedString.empty() == false) {
00662     oTokenList.push_back (lMatchedString);
00663     //}
00664   }
00665
00666   // DEBUG
00667   // std::cout << "After (token list): " << oTokenList << std::endl;
00668
00669   return oTokenList;
00670 }
00671
00672 // ////////////////////////////////////////////////////////
00673 TokenList_T extractTokenListForFlight (const TokenList_T& iTokenList) {
00680   const std::string lRegEx ("^([[:alpha:]]{2,3})?"
00681                             "[[:space:]]*([[:digit:]]{1,4})?$");
00682
00683   //
00684   const TokenList_T& oTokenList = extractTokenList (iTokenList, lRegEx);
00685   return oTokenList;
00686 }
00687
00688 // ////////////////////////////////////////////////////////
00689 TokenList_T extractTokenListForFlightDate (const TokenList_T& iTokenList) {
00700   const std::string lRegEx("^([[:alpha:]]{2,3})?"
00701                            "[[:space:]]*([[:digit:]]{1,4})?"
00702                            "[/ ]*"
00703                            "([[:digit:]]{2,4})?[/-]?[[:space:]]*"
00704                            "([[:alpha:]]{3}|[[:digit:]]{1,2})?[/-]?[[:space:]]*
     "
00705                            "([[:digit:]]{1,2})?$");
00706
00707   //
00708   const TokenList_T& oTokenList = extractTokenList (iTokenList, lRegEx);
00709   return oTokenList;
00710 }
00711
00712 // ////////////////////////////////////////////////////////
00713 TokenList_T extractTokenListForClass (const TokenList_T& iTokenList) {
00722   const std::string lRegEx ("^([[:alpha:]])?"
00723                             "[[:space:]]*([[:digit:]]{1,3})?"
00724                             "[[:space:]]*([[:alpha:]]{3})?"
00725                             "[[:space:]]*([[:alpha:]]{3})?$");
00726
00727   //
00728   const TokenList_T& oTokenList = extractTokenList (iTokenList, lRegEx);
00729   return oTokenList;
00730 }
00731
```

```
00732
00733 // ///////// M A I N ////////////
00734 int main (int argc, char* argv[]) {
00735
00736   // State whether the BOM tree should be built-in or parsed from an
00737   // input file
00738   bool isBuiltin;
00739   bool isForSchedule;
00740
00741   // Input file names
00742   stdair::Filename_T lInventoryFilename;
00743   stdair::Filename_T lScheduleInputFilename;
00744   stdair::Filename_T lODInputFilename;
00745   stdair::Filename_T lFRAT5InputFilename;
00746   stdair::Filename_T lFFDisutilityInputFilename;
00747   stdair::Filename_T lYieldInputFilename;
00748
00749   // Readline history
00750   const unsigned int lHistorySize (100);
00751   const std::string lHistoryFilename ("airinv.hist");
00752   const std::string lHistoryBackupFilename ("airinv.hist.bak");
00753
00754   // Default parameters for the interactive session
00755   stdair::AirlineCode_T lLastInteractiveAirlineCode;
00756   stdair::FlightNumber_T lLastInteractiveFlightNumber;
00757   stdair::Date_T lLastInteractiveDate;
00758   stdair::AirlineCode_T lInteractiveAirlineCode;
00759   stdair::FlightNumber_T lInteractiveFlightNumber;
00760   stdair::Date_T lInteractiveDate;
00761   stdair::AirportCode_T lInteractiveOrigin;
00762   stdair::AirportCode_T lInteractiveDestination;
00763   stdair::ClassCode_T lInteractiveBookingClass;
00764   stdair::PartySize_T lInteractivePartySize;
00765
00766   // Parameters for the sale
00767   std::string lSegmentDateKey;
00768
00769   // Output log File
00770   stdair::Filename_T lLogFilename;
00771
00772   // Call the command-line option parser
00773   const int lOptionParserStatus =
00774     readConfiguration (argc, argv, isBuiltin, isForSchedule, lInventoryFilename
      ,
00775                        lScheduleInputFilename, lODInputFilename,
00776                        lFRAT5InputFilename, lFFDisutilityInputFilename,
00777                        lYieldInputFilename, lLogFilename);
00778
00779   if (lOptionParserStatus == K_AIRINV_EARLY_RETURN_STATUS) {
00780     return 0;
00781   }
00782
00783   // Set the log parameters
00784   std::ofstream logOutputFile;
00785   // Open and clean the log outputfile
00786   logOutputFile.open (lLogFilename.c_str());
00787   logOutputFile.clear();
00788
00789   // Initialise the inventory service
00790   const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
00791   AIRINV::AIRINV_Master_Service airinvService (
      lLogParams);
00792
00793   // DEBUG
00794   STDAIR_LOG_DEBUG ("Welcome to AirInv");
00795
00796   // Check wether or not a (CSV) input file should be read
00797   if (isBuiltin == true) {
00798
00799     // Build the sample BOM tree for RMOL
00800     airinvService.buildSampleBom();
00801
00802     // Update the default parameters for the following interactive session
00803     lInteractiveAirlineCode = "BA";
00804     lInteractiveFlightNumber = 9;
00805     lInteractiveDate = stdair::Date_T (2011, 06, 10);
00806     lInteractiveBookingClass = "Q";
00807     lInteractivePartySize = 2;
00808     lInteractiveOrigin = "LHR";
00809     lInteractiveDestination = "SYD";
00810
00811   } else {
00812     if (isForSchedule == true) {
00813       // Build the BOM tree from parsing a schedule file (and O&D list)
00814       stdair::ScheduleFilePath lScheduleFilePath (lScheduleInputFilename);
00815       stdair::ODFilePath lODFilePath (lODInputFilename);
00816       stdair::FRAT5FilePath lFRAT5FilePath (lFRAT5InputFilename);
```

```
00817       stdair::FFDisutilityFilePath lFFDisutilityFilePath (
      lFFDisutilityInputFilename);
00818       AIRRAC::YieldFilePath lYieldFilePath (lYieldInputFilename);
00819       airinvService.parseAndLoad (lScheduleFilePath, lODFilePath,
00820                                   lFRAT5FilePath, lFFDisutilityFilePath,
00821                                   lYieldFilePath);
00822
00823       // Update the default parameters for the following interactive session
00824       lInteractiveAirlineCode = "SQ";
00825       lInteractiveFlightNumber = 11;
00826       lInteractiveDate = stdair::Date_T (2010, 01, 15);
00827       lInteractiveBookingClass = "Y";
00828       lInteractivePartySize = 2;
00829       lInteractiveOrigin = "SIN";
00830       lInteractiveDestination = "BKK";
00831
00832     } else {
00833       // Build the BOM tree from parsing an inventory dump file
00834       AIRINV::InventoryFilePath lInventoryFilePath (
      lInventoryFilename);
00835       airinvService.parseAndLoad (lInventoryFilePath);
00836
00837       // Update the default parameters for the following interactive session
00838       lInteractiveAirlineCode = "SV";
00839       lInteractiveFlightNumber = 5;
00840       lInteractiveDate = stdair::Date_T (2010, 03, 11);
00841       lInteractiveBookingClass = "Y";
00842       lInteractivePartySize = 2;
00843       lInteractiveOrigin = "KBP";
00844       lInteractiveDestination = "JFK";
00845     }
00846   }
00847
00848   // Save the last state
00849   lLastInteractiveAirlineCode = lInteractiveAirlineCode;
00850   lLastInteractiveFlightNumber = lInteractiveFlightNumber;
00851   lLastInteractiveDate = lInteractiveDate;
00852
00853   // DEBUG
00854   STDAIR_LOG_DEBUG ("=====================================================");
00855   STDAIR_LOG_DEBUG ("=       Beginning of the interactive session       =");
00856   STDAIR_LOG_DEBUG ("=====================================================");
00857
00858   // Initialise the GNU readline wrapper
00859   swift::SReadline lReader (lHistoryFilename, lHistorySize);
00860   initReadline (lReader);
00861
00862   // Now we can ask user for a line
00863   std::string lUserInput;
00864   bool EndOfInput (false);
00865   Command_T::Type_T lCommandType (Command_T::NOP);
00866
00867   while (lCommandType != Command_T::QUIT && EndOfInput == false) {
00868     // Prompt
00869     std::ostringstream oPromptStr;
00870     oPromptStr << "airinv "
00871                << lInteractiveAirlineCode << lInteractiveFlightNumber
00872                << " / " << lInteractiveDate
00873                << "> ";
00874     // Call read-line, which will fill the list of tokens
00875     TokenList_T lTokenListByReadline;
00876     lUserInput = lReader.GetLine (oPromptStr.str(), lTokenListByReadline,
00877                                   EndOfInput);
00878
00879     // The history can be saved to an arbitrary file at any time
00880     lReader.SaveHistory (lHistoryBackupFilename);
00881
00882     // The end-of-input typically corresponds to a CTRL-D typed by the user
00883     if (EndOfInput) {
00884       std::cout << std::endl;
00885       break;
00886     }
00887
00888     // Interpret the user input
00889     lCommandType = extractCommand (lTokenListByReadline);
00890
00891     switch (lCommandType) {
00892
00893       // ////////////////////////////// Help ////////////////////////
00894     case Command_T::HELP: {
00895       std::cout << std::endl;
00896       std::cout << "Commands: " << std::endl;
00897       std::cout << " help" << "\t\t" << "Display this help" << std::endl;
00898       std::cout << " quit" << "\t\t" << "Quit the application" << std::endl;
00899       std::cout << " list" << "\t\t"
00900                 << "List airlines, flights and departure dates" << std::endl;
00901       std::cout << " select" << "\t\t"
```

```
00902                 << "Select a flight-date to become the current one"
00903                 << std::endl;
00904       std::cout << " display" << "\t"
00905                 << "Display the current flight-date" << std::endl;
00906       std::cout << " sell" << "\t\t"
00907                 << "Make a booking on the current flight-date" << std::endl;
00908       std::cout << " \nDebug Commands" << std::endl;
00909       std::cout << " json_list" << "\t"
00910                 << "List airlines, flights and departure dates in a JSON format
    "
00911                 << std::endl;
00912       std::cout << " json_display" << "\t"
00913                 << "Display the current flight-date in a JSON format"
00914                 << std::endl;
00915       std::cout << std::endl;
00916       break;
00917     }
00918
00919       // ////////////////////////////// Quit ////////////////////////
00920     case Command_T::QUIT: {
00921       break;
00922     }
00923
00924       // ////////////////////////////// List ////////////////////////
00925     case Command_T::LIST: {
00926       //
00927       TokenList_T lTokenList = extractTokenListForFlight (lTokenListByReadline)
    ;
00928
00929       stdair::AirlineCode_T lAirlineCode ("all");
00930       stdair::FlightNumber_T lFlightNumber (0);
00931       // Parse the parameters given by the user, giving default values
00932       // in case the user does not specify some (or all) of them
00933       parseFlightKey (lTokenList, lAirlineCode, lFlightNumber);
00934
00935       //
00936       const std::string lFlightNumberStr = (lFlightNumber ==0)?" (all)":"";
00937       std::cout << "List of flights for "
00938                 << lAirlineCode << " " << lFlightNumber << lFlightNumberStr
00939                 << std::endl;
00940
00941       // DEBUG: Display the flight-date
00942       const std::string& lFlightDateListStr =
00943         airinvService.list (lAirlineCode, lFlightNumber);
00944
00945       if (lFlightDateListStr.empty() == false) {
00946         std::cout << lFlightDateListStr << std::endl;
00947         STDAIR_LOG_DEBUG (lFlightDateListStr);
00948
00949       } else {
00950         std::cerr << "There is no result for "
00951                   << lAirlineCode << " " << lFlightNumber << lFlightNumberStr
00952                   << ". Just type the list command without any parameter "
00953                   << "to see the flight-dates for all the airlines and for all
    "
00954                   << "the flight numbers."
00955                   << std::endl;
00956       }
00957
00958       break;
00959     }
00960
00961       // ////////////////////////////// Select ////////////////////////
00962     case Command_T::SELECT: {
00963       //
00964       TokenList_T lTokenList =
00965         extractTokenListForFlightDate (lTokenListByReadline);
00966
00967       // Check whether the user wants to select the last saved flight-date
00968       if (lTokenList.empty() == false) {
00969         // Read the booking class
00970         TokenList_T::const_iterator itTok = lTokenList.begin();
00971
00972         if (*itTok == "-") {
00973
00974           // Swap the current state with the last state
00975           boost::swap (lInteractiveAirlineCode, lLastInteractiveAirlineCode);
00976           boost::swap (lInteractiveFlightNumber, lLastInteractiveFlightNumber);
00977           boost::swap (lInteractiveDate, lLastInteractiveDate);
00978
00979           break;
00980         }
00981       }
00982
00983       // Parse the parameters given by the user, giving default values
00984       // in case the user does not specify some (or all) of them
00985       parseFlightDateKey (lTokenList, lInteractiveAirlineCode,
```

```
00986                           lInteractiveFlightNumber, lInteractiveDate);
00987
00988       // Check whether the selected flight-date is valid
00989       const bool isFlightDateValid =
00990         airinvService.check (lInteractiveAirlineCode, lInteractiveFlightNumber,
00991                              lInteractiveDate);
00992       if (isFlightDateValid == false) {
00993         std::ostringstream oFDKStr;
00994         oFDKStr << "The " << lInteractiveAirlineCode
00995                 << lInteractiveFlightNumber << " / " << lInteractiveDate
00996                 << " flight-date is not valid. Make sure it exists (e.g.,"
00997                 << " with the list command). The current flight-date is kept"
00998                 << " selected.";
00999         std::cout << oFDKStr.str() << std::endl;
01000         STDAIR_LOG_ERROR (oFDKStr.str());
01001
01002         // Restore the last state
01003         lInteractiveAirlineCode = lLastInteractiveAirlineCode;
01004         lInteractiveFlightNumber = lLastInteractiveFlightNumber;
01005         lInteractiveDate = lLastInteractiveDate;
01006
01007         break;
01008       }
01009
01010       // DEBUG: Display the flight-date selection
01011       std::ostringstream oFDKStr;
01012       oFDKStr << "Selected the " << lInteractiveAirlineCode
01013               << lInteractiveFlightNumber << " / " << lInteractiveDate
01014               << " flight-date";
01015       std::cout << oFDKStr.str() << std::endl;
01016       STDAIR_LOG_DEBUG (oFDKStr.str());
01017
01018       // Save the last state
01019       lLastInteractiveAirlineCode = lInteractiveAirlineCode;
01020       lLastInteractiveFlightNumber = lInteractiveFlightNumber;
01021       lLastInteractiveDate = lInteractiveDate;
01022
01023       break;
01024     }
01025
01026       // ////////////////////////////// Display ////////////////////////
01027     case Command_T::DISPLAY: {
01028       // DEBUG: Display the flight-date
01029       const std::string& lCSVFlightDateDump =
01030         airinvService.csvDisplay (lInteractiveAirlineCode,
01031                                   lInteractiveFlightNumber, lInteractiveDate);
01032       std::cout << lCSVFlightDateDump << std::endl;
01033       STDAIR_LOG_DEBUG (lCSVFlightDateDump);
01034
01035       break;
01036     }
01037
01038       // ////////////////////////////// Sell ////////////////////////
01039     case Command_T::SELL: {
01040       //
01041       TokenList_T lTokenList = extractTokenListForClass (lTokenListByReadline);
01042
01043       // Parse the parameters given by the user, giving default values
01044       // in case the user does not specify some (or all) of them
01045       parseBookingClassKey (lTokenList, lInteractiveBookingClass,
01046                             lInteractivePartySize,
01047                             lInteractiveOrigin, lInteractiveDestination);
01048
01049       // DEBUG: Display the flight-date before the sell
01050       const std::string& lCSVFlightDateDumpBefore =
01051         airinvService.csvDisplay (lInteractiveAirlineCode,
01052                                   lInteractiveFlightNumber, lInteractiveDate);
01053       //std::cout << lCSVFlightDateDumpBefore << std::endl;
01054       STDAIR_LOG_DEBUG (lCSVFlightDateDumpBefore);
01055
01056       // Make a booking
01057       std::ostringstream oSDKStr;
01058       oSDKStr << lInteractiveAirlineCode << ","
01059               << lInteractiveFlightNumber << ","
01060               << lInteractiveDate << ","
01061               << lInteractiveOrigin << "," << lInteractiveDestination;
01062       const std::string lSegmentDateKey (oSDKStr.str());
01063
01064       // Perform the sell
01065       const bool isSellSuccessful =
01066         airinvService.sell (lSegmentDateKey,
01067                             lInteractiveBookingClass, lInteractivePartySize);
01068
01069       // DEBUG
01070       const std::string isSellSuccessfulStr =
01071         (isSellSuccessful == true)?"Yes":"No";
01072       std::ostringstream oSaleStr;
```

```
01073       oSaleStr << "Sale (’" << lSegmentDateKey << "’, "
01074                << lInteractiveBookingClass << ": " << lInteractivePartySize
01075                << ") successful? " << isSellSuccessfulStr;
01076       std::cout << oSaleStr.str() << std::endl;
01077
01078       // DEBUG
01079       STDAIR_LOG_DEBUG (oSaleStr.str());
01080
01081       // DEBUG: Display the flight-date after the sell
01082       const std::string& lCSVFlightDateDumpAfter =
01083         airinvService.csvDisplay (lInteractiveAirlineCode,
01084                                   lInteractiveFlightNumber, lInteractiveDate);
01085       //std::cout << lCSVFlightDateDumpAfter << std::endl;
01086       STDAIR_LOG_DEBUG (lCSVFlightDateDumpAfter);
01087
01088       break;
01089     }
01090
01091       // ////////////////////////////// JSon List ////////////////////////
01092
01093     case Command_T::JSON_LIST: {
01094
01095       //
01096       TokenList_T lTokenList = extractTokenListForFlight (lTokenListByReadline)
      ;
01097
01098       stdair::AirlineCode_T lAirlineCode ("all");
01099       stdair::FlightNumber_T lFlightNumber (0);
01100       // Parse the parameters given by the user, giving default values
01101       // in case the user does not specify some (or all) of them
01102       parseFlightKey (lTokenList, lAirlineCode, lFlightNumber);
01103
01104       //
01105       const std::string lFlightNumberStr = (lFlightNumber ==0)?" (all)":"";
01106       std::cout << "JSON list of flights for "
01107                 << lAirlineCode << " " << lFlightNumber << lFlightNumberStr
01108                 << std::endl;
01109
01110       std::ostringstream lMyCommandJSONstream;
01111       lMyCommandJSONstream << "{\"list\":"
01112                            << "{ \"airline_code\":\"" << lAirlineCode
01113                            << "\",\"flight_number\":\"" << lFlightNumber
01114                            << "\"}}";
01115
01116       const stdair::JSONString lJSONCommandString (lMyCommandJSONstream.str());
01117       const std::string& lFlightDateListJSONStr =
01118         airinvService.jsonHandler (lJSONCommandString);
01119
01120       // Display the flight-date JSON string
01121       std::cout << lFlightDateListJSONStr << std::endl;
01122       STDAIR_LOG_DEBUG (lFlightDateListJSONStr);
01123
01124       break;
01125     }
01126
01127       // ////////////////////////////// JSon Display ////////////////////////
01128
01129     case Command_T::JSON_DISPLAY: {
01130
01131       // Construct the JSON command string for the current parameters (current
01132       // airline code, current flight number and current date)
01133       std::ostringstream lMyCommandJSONstream;
01134       lMyCommandJSONstream << "{\"flight_date\":"
01135                            << "{ \"departure_date\":\"" << lInteractiveDate
01136                            << "\",\"airline_code\":\"" <<
      lInteractiveAirlineCode
01137                            << "\",\"flight_number\":\"" <<
      lInteractiveFlightNumber
01138                            << "\"}}";
01139
01140       // Get the flight-date details in a JSON string
01141       const stdair::JSONString lJSONCommandString (lMyCommandJSONstream.str());
01142       const std::string& lCSVFlightDateDump =
01143         airinvService.jsonHandler (lJSONCommandString);
01144
01145       // Display the flight-date JSON string
01146       std::cout << lCSVFlightDateDump << std::endl;
01147       STDAIR_LOG_DEBUG (lCSVFlightDateDump);
01148
01149       break;
01150     }
01151
01152       // ////////////////////////////// Default / No value ////////////////////////
01153     case Command_T::NOP: {
01154       break;
01155     }
01156
```

```
01157     case Command_T::LAST_VALUE:
01158     default: {
01159       // DEBUG
01160       std::ostringstream oStr;
01161       oStr << "That command is not yet understood: '" << lUserInput
01162            << "' => " << lTokenListByReadline;
01163       STDAIR_LOG_DEBUG (oStr.str());
01164       std::cout << oStr.str() << std::endl;
01165     }
01166     }
01167   }
01168
01169   // DEBUG
01170   STDAIR_LOG_DEBUG ("End of the session. Exiting.");
01171   std::cout << "End of the session. Exiting." << std::endl;
01172
01173   // Close the Log outputFile
01174   logOutputFile.close();
01175
01176   /*
01177     Note: as that program is not intended to be run on a server in
01178     production, it is better not to catch the exceptions. When it
01179     happens (that an exception is thrownded), that way we get the
01180     call stack.
01181   */
01182
01183   return 0;
01184 }
```

23.257 doc/local/authors.doc File Reference

23.258 doc/local/codingrules.doc File Reference

23.259 doc/local/copyright.doc File Reference

23.260 doc/local/documentation.doc File Reference

23.261 doc/local/features.doc File Reference

23.262 doc/local/help_wanted.doc File Reference

23.263 doc/local/howto_release.doc File Reference

23.264 doc/local/index.doc File Reference

23.265 doc/local/installation.doc File Reference

23.266 doc/local/linking.doc File Reference

23.267 doc/local/test.doc File Reference

23.268 doc/local/users_guide.doc File Reference

23.269 doc/local/verification.doc File Reference

23.270 doc/tutorial/tutorial.doc File Reference

23.271 test/airinv/InventoryTestSuite.cpp File Reference

23.272 InventoryTestSuite.cpp

```
00001
00005 // //////////////////////////////////////////////////////////////////////
00006 // Import section
```

```
00007 // ////////////////////////////////////////////////////////////////////
00008 // STL
00009 #include <sstream>
00010 #include <fstream>
00011 #include <string>
00012 // Boost Unit Test Framework (UTF)
00013 #define BOOST_TEST_DYN_LINK
00014 #define BOOST_TEST_MAIN
00015 #define BOOST_TEST_MODULE InventoryTestSuite
00016 #include <boost/test/unit_test.hpp>
00017 // StdAir
00018 #include <stdair/basic/BasLogParams.hpp>
00019 #include <stdair/basic/BasDBParams.hpp>
00020 #include <stdair/basic/BasFileMgr.hpp>
00021 #include <stdair/bom/TravelSolutionStruct.hpp>
00022 #include <stdair/bom/BookingRequestStruct.hpp>
00023 #include <stdair/service/Logger.hpp>
00024 #include <stdair/stdair_exceptions.hpp>
00025 // Airinv
00026 #include <airinv/AIRINV_Types.hpp>
00027 #include <airinv/AIRINV_Master_Service.hpp>
00028 #include <airinv/config/airinv-paths.hpp>
00029
00030 namespace boost_utf = boost::unit_test;
00031
00032 // (Boost) Unit Test XML Report
00033 std::ofstream utfReportStream ("InventoryTestSuite_utfresults.xml");
00034
00038 struct UnitTestConfig {
00040   UnitTestConfig() {
00041     boost_utf::unit_test_log.set_stream (utfReportStream);
00042     boost_utf::unit_test_log.set_format (boost_utf::XML);
00043     boost_utf::unit_test_log.set_threshold_level (boost_utf::log_test_units);
00044     //boost_utf::unit_test_log.set_threshold_level
       (boost_utf::log_successful_tests);
00045   }
00046
00048   ~UnitTestConfig() {
00049   }
00050 };
00051
00052 // ////////////////////////////////////////////////////////////////////
00056 bool testInventoryHelper (const unsigned short iTestFlag,
00057                           const stdair::Filename_T& iInventoryInputFilename,
00058                           const stdair::Filename_T& iScheduleInputFilename,
00059                           const stdair::Filename_T& iODInputFilename,
00060                           const stdair::Filename_T& iFRAT5InputFilename,
00061                           const stdair::Filename_T& iFFDisutilityInputFilename,
00062                           const stdair::Filename_T& iYieldInputFilename,
00063                           const bool isBuiltin,
00064                           const bool isForSchedule) {
00065
00066   // Output log File
00067   std::ostringstream oStr;
00068   oStr << "InventoryTestSuite_" << iTestFlag << ".log";
00069   const stdair::Filename_T lLogFilename (oStr.str());
00070
00071   // Set the log parameters
00072   std::ofstream logOutputFile;
00073   // Open and clean the log outputfile
00074   logOutputFile.open (lLogFilename.c_str());
00075   logOutputFile.clear();
00076
00077   // Initialise the AirInv service object
00078   stdair::BasLogParams lLogParams (stdair::LOG::DEBUG,
00079                                    logOutputFile);
00080
00081   // Initialise the inventory service
00082   AIRINV::AIRINV_Master_Service airinvService (
       lLogParams);
00083
00084   // Parameters for the sale
00085   std::string lSegmentDateKey;
00086   stdair::ClassCode_T lClassCode;
00087   const stdair::PartySize_T lPartySize (2);
00088
00089   // Check wether or not a (CSV) input file should be read
00090   if (isBuiltin == true) {
00091
00092     // Build the default sample BOM tree (filled with inventories) for AirInv
00093     airinvService.buildSampleBom();
00094
00095     // Define a specific segment-date key for the sample BOM tree
00096     lSegmentDateKey = "BA,9,2011-06-10,LHR,SYD";
00097     lClassCode = "Q";
00098
00099   } else {
```

```
00100
00101     if (isForSchedule == true) {
00102       // Build the BOM tree from parsing a schedule file (and O&D list)
00103       stdair::ScheduleFilePath lScheduleFilePath (iScheduleInputFilename);
00104       stdair::ODFilePath lODFilePath (iODInputFilename);
00105       stdair::FRAT5FilePath lFRAT5FilePath (iFRAT5InputFilename);
00106       stdair::FFDisutilityFilePath lFFDisutilityFilePath (
      iFFDisutilityInputFilename);
00107       AIRRAC::YieldFilePath lYieldFilePath (iYieldInputFilename);
00108       airinvService.parseAndLoad (lScheduleFilePath, lODFilePath,
00109                                   lFRAT5FilePath, lFFDisutilityFilePath,
00110                                   lYieldFilePath);
00111
00112       // Define a specific segment-date key for the schedule-based inventory
00113       lSegmentDateKey = "SQ,11,2010-01-15,SIN,BKK";
00114       lClassCode = "Y";
00115
00116     } else {
00117
00118       // Build the BOM tree from parsing an inventory dump file
00119       AIRINV::InventoryFilePath lInventoryFilePath (
      iInventoryInputFilename);
00120       airinvService.parseAndLoad (lInventoryFilePath);
00121
00122       // Define a specific segment-date key for the inventory parsed file
00123       //const std::string lSegmentDateKey ("SV, 5, 2010-03-11, KBP, JFK,
       08:00:00");
00124       lSegmentDateKey =  "SV, 5, 2010-03-11, KBP, JFK, 08:00:00";
00125       lClassCode = "J";
00126     }
00127
00128   }
00129
00130   // Make a booking
00131   const bool hasSaleBeenSuccessful =
00132     airinvService.sell (lSegmentDateKey, lClassCode, lPartySize);
00133
00134   // DEBUG: Display the list of travel solutions
00135   const std::string& lCSVDump = airinvService.csvDisplay();
00136   STDAIR_LOG_DEBUG (lCSVDump);
00137
00138   // Close the log file
00139   logOutputFile.close();
00140
00141   if (hasSaleBeenSuccessful == false) {
00142     STDAIR_LOG_DEBUG ("No sale can be made for '" << lSegmentDateKey
00143                       << "'");
00144   }
00145
00146   return hasSaleBeenSuccessful;
00147
00148 }
00149
00150 // ///////////////// Main: Unit Test Suite //////////////
00151
00152 // Set the UTF configuration (re-direct the output to a specific file)
00153 BOOST_GLOBAL_FIXTURE (UnitTestConfig);
00154
00155 // Start the test suite
00156 BOOST_AUTO_TEST_SUITE (master_test_suite)
00157
00158
00161 BOOST_AUTO_TEST_CASE (airinv_simple_inventory_sell) {
00162
00163   // Input file name
00164   const stdair::Filename_T lInventoryInputFilename (STDAIR_SAMPLE_DIR
00165                                                     "/invdump01.csv");
00166
00167   // State whether the BOM tree should be built-in or parsed from an input file
00168   const bool isBuiltin = false;
00169   // State whether the BOM tree should be built from a schedule file (instead
       of from an inventory dump)
00170   const bool isForSchedule = false;
00171
00172   // Try sell a default segment.
00173   bool hasTestBeenSuccessful = false;
00174   BOOST_CHECK_NO_THROW (hasTestBeenSuccessful =
00175                         testInventoryHelper (0, lInventoryInputFilename,
00176                                              " ", " ", " ", " ", " ", isBuiltin
      , isForSchedule));
00177   BOOST_CHECK_EQUAL (hasTestBeenSuccessful, true);
00178
00179 }
00180
00184 BOOST_AUTO_TEST_CASE (airinv_simple_inventory_sell_built_in) {
00185
00186   // State whether the BOM tree should be built-in or parsed from an input file
```

```
00187   const bool isBuiltin = true;
00188   // State whether the BOM tree should be built from a schedule file (instead
       of from an inventory dump)
00189   const bool isForSchedule = false;
00190
00191   // Try sell a default segment.
00192   bool hasTestBeenSuccessful = false;
00193   BOOST_CHECK_NO_THROW (hasTestBeenSuccessful =
00194                         testInventoryHelper (1, " ", " ", " ", " ", " ", " ",
00195                                              isBuiltin, isForSchedule));
00196   BOOST_CHECK_EQUAL (hasTestBeenSuccessful, true);
00197
00198 }
00199
00203 BOOST_AUTO_TEST_CASE (airinv_simple_inventory_sell_schedule) {
00204
00205   // Input file names
00206   const stdair::Filename_T lScheduleInputFilename (STDAIR_SAMPLE_DIR
00207                                                    "/schedule01.csv");
00208   const stdair::Filename_T lODInputFilename (STDAIR_SAMPLE_DIR
00209                                              "/ond01.csv");
00210   const stdair::Filename_T lFRAT5InputFilename (STDAIR_SAMPLE_DIR
00211                                                 "/frat5.csv");
00212   const stdair::Filename_T lFFDisutilityInputFilename (STDAIR_SAMPLE_DIR
00213                                                        "/ffDisutility.csv");
00214   const stdair::Filename_T lYieldInputFilename (STDAIR_SAMPLE_DIR
00215                                                 "/yieldstore01.csv");
00216
00217   // State whether the BOM tree should be built-in or parsed from an input file
00218   const bool isBuiltin = false;
00219   // State whether the BOM tree should be built from a schedule file (instead
       of from an inventory dump)
00220   const bool isForSchedule = true;
00221
00222   // Try sell a default segment.
00223   bool hasTestBeenSuccessful = false;
00224   BOOST_CHECK_NO_THROW (hasTestBeenSuccessful =
00225                         testInventoryHelper (2, " ",
00226                                              lScheduleInputFilename,
00227                                              lODInputFilename,
00228                                              lFRAT5InputFilename,
00229                                              lFFDisutilityInputFilename,
00230                                              lYieldInputFilename,
00231                                              isBuiltin, isForSchedule));
00232   BOOST_CHECK_EQUAL (hasTestBeenSuccessful, true);
00233
00234 }
00235
00240 BOOST_AUTO_TEST_CASE (airinv_error_inventory_input_file) {
00241
00242   // Inventory input file name
00243   const stdair::Filename_T lMissingInventoryFilename (STDAIR_SAMPLE_DIR
00244                                                       "/missingFile.csv");
00245
00246   // State whether the BOM tree should be built-in or parsed from an input file
00247   const bool isBuiltin = false;
00248   // State whether the BOM tree should be built from a schedule file (instead
       of from an inventory dump)
00249   const bool isForSchedule = false;
00250
00251   // Try sell a default segment.
00252   BOOST_CHECK_THROW (testInventoryHelper (3, lMissingInventoryFilename,
00253                                           " ", " ", " ", " ", " ", isBuiltin,
      isForSchedule),
00254                      AIRINV::InventoryInputFileNotFoundException
      );
00255
00256 }
00257
00262 BOOST_AUTO_TEST_CASE (airinv_error_schedule_input_file) {
00263
00264   // Schedule input file name
00265   const stdair::Filename_T lMissingScheduleFilename (STDAIR_SAMPLE_DIR
00266                                                      "/missingFile.csv");
00267   const stdair::Filename_T lFRAT5InputFilename (STDAIR_SAMPLE_DIR
00268                                                "/frat5.csv");
00269   const stdair::Filename_T lFFDisutilityInputFilename (STDAIR_SAMPLE_DIR
00270                                                        "/ffDisutility.csv");
00271
00272   // State whether the BOM tree should be built-in or parsed from an input file
00273   const bool isBuiltin = false;
00274   // State whether the BOM tree should be built from a schedule file (instead
       of from an inventory dump)
00275   const bool isForSchedule = true;
00276
00277   // Try sell a default segment.
00278   BOOST_CHECK_THROW (testInventoryHelper (4, " ", lMissingScheduleFilename,
```

```
00279                                         " ", lFRAT5InputFilename,
00280                                         lFFDisutilityInputFilename, " ",
00281                                         isBuiltin, isForSchedule),
00282                       AIRINV::ScheduleInputFileNotFoundException
     );
00283
00284 }
00285
00290 BOOST_AUTO_TEST_CASE (airinv_error_yield_input_file) {
00291
00292   // Input file names
00293   const stdair::Filename_T lScheduleInputFilename (STDAIR_SAMPLE_DIR
00294                                                    "/schedule01.csv");
00295   const stdair::Filename_T lODInputFilename (STDAIR_SAMPLE_DIR
00296                                              "/ond01.csv");
00297   const stdair::Filename_T lFRAT5InputFilename (STDAIR_SAMPLE_DIR
00298                                                "/frat5.csv");
00299   const stdair::Filename_T lFFDisutilityInputFilename (STDAIR_SAMPLE_DIR
00300                                                        "/ffDisutility.csv");
00301   const stdair::Filename_T lYieldInputFilename (STDAIR_SAMPLE_DIR
00302                                                 "/missingFile.csv");
00303
00304   // State whether the BOM tree should be built-in or parsed from an input file
00305   const bool isBuiltin = false;
00306   // State whether the BOM tree should be built from a schedule file (instead
      of from an inventory dump)
00307   const bool isForSchedule = true;
00308
00309   // Try sell a default segment.
00310   BOOST_CHECK_THROW (testInventoryHelper (5, " ",
00311                                           lScheduleInputFilename,
00312                                           lODInputFilename,
00313                                           lFRAT5InputFilename,
00314                                           lFFDisutilityInputFilename,
00315                                           lYieldInputFilename,
00316                                           isBuiltin, isForSchedule),
00317                      AIRRAC::YieldInputFileNotFoundException);
00318
00319 }
00320
00325 BOOST_AUTO_TEST_CASE (airinv_error_flight_date_duplication) {
00326
00327   // Input file names
00328   const stdair::Filename_T lScheduleInputFilename (STDAIR_SAMPLE_DIR
00329                                                    "/scheduleError01.csv");
00330   const stdair::Filename_T lODInputFilename (STDAIR_SAMPLE_DIR
00331                                              "/ond01.csv");
00332   const stdair::Filename_T lFRAT5InputFilename (STDAIR_SAMPLE_DIR
00333                                                "/frat5.csv");
00334   const stdair::Filename_T lFFDisutilityInputFilename (STDAIR_SAMPLE_DIR
00335                                                        "/ffDisutility.csv");
00336   const stdair::Filename_T lYieldInputFilename (STDAIR_SAMPLE_DIR
00337                                                 "/missingFile.csv");
00338
00339   // State whether the BOM tree should be built-in or parsed from an input file
00340   const bool isBuiltin = false;
00341   // State whether the BOM tree should be built from a schedule file (instead
      of from an inventory dump)
00342   const bool isForSchedule = true;
00343
00344   // Try sell a default segment.
00345   BOOST_CHECK_THROW (testInventoryHelper (6, " ",
00346                                           lScheduleInputFilename,
00347                                           lODInputFilename,
00348                                           lFRAT5InputFilename,
00349                                           lFFDisutilityInputFilename,
00350                                           lYieldInputFilename,
00351                                           isBuiltin, isForSchedule),
00352                      AIRINV::FlightDateDuplicationException
     );
00353
00354 }
00355
00360 BOOST_AUTO_TEST_CASE (airinv_error_schedule_parsing_failed) {
00361
00362   // Input file names
00363   const stdair::Filename_T lScheduleInputFilename (STDAIR_SAMPLE_DIR
00364                                                    "/scheduleError02.csv");
00365   const stdair::Filename_T lODInputFilename (STDAIR_SAMPLE_DIR
00366                                              "/ond01.csv");
00367   const stdair::Filename_T lFRAT5InputFilename (STDAIR_SAMPLE_DIR
00368                                                "/frat5.csv");
00369   const stdair::Filename_T lFFDisutilityInputFilename (STDAIR_SAMPLE_DIR
00370                                                        "/ffDisutility.csv");
00371   const stdair::Filename_T lYieldInputFilename (STDAIR_SAMPLE_DIR
00372                                                 "/yieldstore01.csv");
00373
```

```
00374   // State whether the BOM tree should be built-in or parsed from an input file
00375   const bool isBuiltin = false;
00376   // State whether the BOM tree should be built from a schedule file (instead
      of from an inventory dump)
00377   const bool isForSchedule = true;
00378
00379   // Try sell a default segment.
00380   BOOST_CHECK_THROW (testInventoryHelper (7, " ",
00381                                           lScheduleInputFilename,
00382                                           lODInputFilename,
00383                                           lFRAT5InputFilename,
00384                                           lFFDisutilityInputFilename,
00385                                           lYieldInputFilename,
00386                                           isBuiltin, isForSchedule),
00387                      AIRINV::ScheduleFileParsingFailedException
     );
00388
00389 }
00390
00391 // End the test suite
00392 BOOST_AUTO_TEST_SUITE_END()
00393
00394
```

23.273 test/airinv/InventoryTestSuite.hpp File Reference

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

Classes

- class InventoryTestSuite

Functions

- CPPUNIT_TEST_SUITE_REGISTRATION (InventoryTestSuite)

23.273.1 Function Documentation

23.273.1.1 CPPUNIT_TEST_SUITE_REGISTRATION (InventoryTestSuite)

23.274 InventoryTestSuite.hpp

```
00001 // STL
00002 #include <iosfwd>
00003 // CPPUNIT
00004 #include <cppunit/extensions/HelperMacros.h>
00005
00007 class InventoryTestSuite : public CppUnit::TestFixture {
00008   CPPUNIT_TEST_SUITE (InventoryTestSuite);
00009   CPPUNIT_TEST (simpleInventory);
00010   //  CPPUNIT_TEST (errorCase);
00011   CPPUNIT_TEST_SUITE_END ();
00012 public:
00013
00015   void simpleInventory();
00016
00018   //  void errorCase ();
00019
00021   InventoryTestSuite ();
00022
00023 private:
00025   void simpleInventoryHelper();
00026
00027 protected:
00028   std::stringstream _describeKey;
00029 };
00030
00031 CPPUNIT_TEST_SUITE_REGISTRATION (
      InventoryTestSuite);
```