AirRAC

1.00.0

Generated by Doxygen 1.6.1

Wed May 22 19:39:53 2013

Contents

1 AirRAC Documentation

1.1 Getting Started

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

1.2 AirRAC at SourceForge

- Project page
- Download AirRAC
- Open a ticket for a bug or feature
- Mailing lists

- Forums
  - Discuss about Development issues
  - Ask for Help
  - Discuss AirRAC

1.3 AirRAC 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 AirRAC

1.6 About AirRAC

AirRAC is a C++ library of airline revenue accounting classes and functions, mainly targeting simulation purposes. N

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

AirRAC 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 AirRAC 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 AirRAC.

2 People

2.1 Project Admins (and Developers)

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

2.2 Retired Developers

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

2.3 Contributors

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

2.4 Distribution Maintainers

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

Note:

(N) - Amadeus employees.

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.

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

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

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

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

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

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

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

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

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

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

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

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

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

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

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

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under

the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

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

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

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

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

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

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

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

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

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received

copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

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

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

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

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

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

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

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

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

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

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

4.3.1 NO WARRANTY

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

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

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 AirRAC 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 AirRAC 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.
 *
 * ------------------------------------------------------------------------
 *
 * AirRAC - C++ Simulated Revenue Accounting (RAC) System Library
 *
 * Copyright (C) 2009-2011  (\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 AirRAC is given below sorted in different categories. Many more features and functions exist and for these we refer to the reference documentation.

6.1 Yield calculation

- Calculation of yields from statistics on tickets/coupons

6.2 Yield rule engine

- Yield rules: storage, engine, management

6.3 Yield retrieval

- Retrieval of yields for specific booking requests or product assesment

6.4 Other features

- CSV input file parsing
- Memory handling

7 Make a Difference

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

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

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

8 Make a new release

8.1 Introduction

This document describes briefly the recommended procedure of releasing a new version of AirRAC 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://airrac.git.sourceforge.net/gitroot/airrac/airrac airracgit
cd airracgit
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/airracgit
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 airrac.spec
```

8.4 Commit and publish the release branch

Commit the new release:

```
cd ~/dev/sim/airracgit
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/airracgit
git checkout trunk
vi ChangeLog
vi airrac.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/airracgit
git checkout 0.5.0
rm -rf build && mkdir -p build
cd build
cmake -DCMAKE_INSTALL_PREFIX=/home/user/dev/deliveries/airrac-0.5.0 \
 -DCMAKE_BUILD_TYPE:STRING=Debug -DINSTALL_DOC:BOOL=ON ..
make check && make dist
```

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

8.7 Generation the RPM packages

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

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

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 `AirRAC'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, `airrac-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/airracgit
git checkout 0.5.0
rsync -aiv doc/html/ doc/latex/refman.pdf joe,airrac@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 airrac-announce@lists.sourceforge.net (see https://lists.sourceforge.net/lists/listinfo/airrac-announce for the archives)

9 Installation

9.1 Table of Contents

- Specifying the System Type
- Sharing Defaults
- Defining Variables
- 'cmake' Invocation

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 airrac-devel airrac-doc
```

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

9.3 AirRAC Requirements

AirRAC 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 AirRAC.

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 files 'CMakeFiles' containing compiler output (useful mainly for debugging 'cmake').

It can also use an optional file (typically called 'config.cache' and enabled with '--cache-file=config.cache' or simply '-C') that saves the results of its tests to speed up reconfiguring. Caching is disabled by default to prevent problems with accidental use of stale cache files.

If you need to do unusual things to compile the package, please try to figure out how 'configure' could check whether to do them, and mail diffs or instructions to the address given in the 'README' so they can be considered for the next release. If you are using the cache, and at some point 'config.cache' contains results you don't want to keep, you may remove or edit it.

The file 'CMakeLists.txt' is used to create the 'Makefile' files.

The simplest way to compile this package is:

1. 'cd' to the directory containing the package's source code and type './cmake ..' to configure the package for your system. Running 'cmake' is generally fast. While running, it prints some messages telling which features it is checking for.
2. Type 'make' to compile the package.
3. Optionally, type 'make check'to run any self-tests that come with the package, generally using the just-built uninstalled binaries.
4. Type 'make install' to install the programs and any data files and documentation. When installing into a prefix owned by root, it is recommended that the package be configured and built as a regular user, and only the 'make install' phase executed with root privileges.
5. You can remove the program binaries and object files from the source code directory by typing 'make clean'. To also remove the files that 'configure' created (so you can compile the package for a different kind of computer), type 'make distclean'. There is also a 'make maintainer-clean' target, but that is intended mainly for the package's developers. If you use it, you may have to get all sorts of other programs in order to regenerate files that came with the distribution.
6. Often, you can also type 'make uninstall' to remove the installed files again. In practice, not all packages have tested that uninstallation works correctly, even though it is required by the GNU Coding Standards.

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=PREFIX', where PREFIX must be an absolute file name.

You can specify separate installation prefixes for architecture-specific files and architecture-independent files. If you pass the option '--exec-prefix=PREFIX' to 'configure', the package uses PREFIX as the prefix for installing programs and libraries. Documentation and other data files still use the regular prefix.

In addition, if you use an unusual directory layout you can give options like '--bindir=DIR' to specify different values for particular

kinds of files. Run 'configure --help' for a list of the directories you can set and what kinds of files go in them. In general, the default for these options is expressed in terms of '${prefix}', so that specifying just '--prefix' will affect all of the other directory specifications that were not explicitly provided.

The most portable way to affect installation locations is to pass the correct locations to 'configure'; however, many packages provide one or both of the following shortcuts of passing variable assignments to the 'make install' command line to change installation locations without having to reconfigure or recompile.

The first method involves providing an override variable for each affected directory. For example, 'make install prefix=/alternate/directory' will choose an alternate location for all directory configuration variables that were expressed in terms of '${prefix}'. Any directories that were specified during 'configure', but not in terms of '${prefix}', must each be overridden at install time for the entire installation to be relocated. The approach of makefile variable overrides for each directory variable is required by the GNU Coding Standards, and ideally causes no recompilation. However, some platforms have known limitations with the semantics of shared libraries that end up requiring recompilation when using this method, particularly noticeable in packages that use GNU Libtool.

The second method involves providing the 'DESTDIR' variable. For example, 'make install DESTDIR=/alternate/directory' will prepend '/alternate/directory' before all installation names. The approach of 'DESTDIR' overrides is not required by the GNU Coding Standards, and does not work on platforms that have drive letters. On the other hand, it does better at avoiding recompilation issues, and works well even when some directory options were not specified in terms of '${prefix}' at 'configure' time.

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=PREFIX' or '--program-suffix=SUFFIX'.

Some packages pay attention to '--enable-FEATURE' options to 'configure', where FEATURE indicates an optional part of the package. They may also pay attention to '--with-PACKAGE' options, where PACKAGE is something like 'gnu-as' or 'x' (for the X Window System). The 'README' should mention any '--enable-' and '--with-' options that the package recognizes.

For packages that use the X Window System, 'configure' can usually find the X include and library files automatically, but if it doesn't, you can use the 'configure' options '--x-includes=DIR' and '--x-libraries=DIR' to specify their locations.

Some packages offer the ability to configure how verbose the execution of 'make' will be. For these packages, running './configure --enable-silent-rules' sets the default to minimal output, which can be overridden with 'make V=1'; while running './configure

--disable-silent-rules' sets the default to verbose, which can be overridden with 'make V=0'.

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 'configure', and exit.
- '--help=short', '--help=recursive' print a summary of the options unique to this package's 'configure', and exit. The 'short' variant lists options used only in the top level, while the 'recursive' variant lists options also present in any nested packages.
- '--version', '-V' print the version of Autoconf used to generate the 'configure' script, and exit.
- '--cache-file=FILE' enable the cache: use and save the results of the tests in FILE, traditionally 'config.cache'. FILE defaults to '/dev/null' to disable caching.

- '--config-cache', '-C' alias for '--cache-file=config.cache'.
- '--quiet', '--silent', '-q' do not print messages saying which checks are being made. To suppress all normal output, redirect it to '/dev/null' (any error messages will still be shown).
- '--srcdir=DIR' look for the package's source code in directory DIR. Usually 'configure' can determine that directory automatically.
- '--prefix=DIR' use DIR as the installation prefix.

  See also:

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

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

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

The 'cmake' script produces an ouput like this:

```
cmake -DCMAKE_INSTALL_PREFIX=/home/user/dev/deliveries/airrac-0.5.0 \
 -DLIB_SUFFIX=64 -DCMAKE_BUILD_TYPE:STRING=Debug -DINSTALL_DOC:BOOL=ON \
 -DWITH_STDAIR_PREFIX=/home/user/dev/deliveries/stdair-stable ..
-- 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: fd0a80b436abd00facc362505699501b2e7acf58 trunk
-- Requires Boost-1.41
-- Boost version: 1.46.0
-- Found the following Boost libraries:
--   program_options
--   date_time
--   iostreams
--   serialization
--   filesystem
--   unit_test_framework
-- Found Boost version: 1.46.0
-- Found BoostWrapper: /usr/include (found suitable version "1.46.0", required is "1.41")
-- Requires MySQL without specifying any version
-- Using mysql-config: /usr/bin/mysql_config
-- Found MySQL: /usr/lib64/mysql/libmysqlclient.so (found version "5.5.14")
-- 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 (found suitable version "3.0.0", required is "3.0")
-- Found SOCIMySQL: /usr/lib64/libsoci_mysql.so (found suitable version "3.0.0", required is "3.0")
-- Found SOCI with MySQL back-end support version: 3.0.0
-- Requires StdAir-0.35
-- Found StdAir version: 0.36.2
-- Requires Doxygen without specifying any version
-- Found Doxygen: /usr/bin/doxygen
```

```
-- Found DoxygenWrapper: /usr/bin/doxygen (found version "1.7.4")
-- Found Doxygen version: 1.7.4
-- Had to set the linker language for 'airraclib' to CXX
-- Test 'YieldTestSuite' to be built with 'YieldTestSuite.cpp'
--
-- ============================================================
-- ---------------------------------
-- ---     Project Information    ---
-- ---------------------------------
-- PROJECT_NAME ................... : airrac
-- PACKAGE_PRETTY_NAME ............ : AirRAC
-- PACKAGE ........................ : airrac
-- PACKAGE_NAME ................... : AIRRAC
-- PACKAGE_VERSION ................ : 0.5.0
-- GENERIC_LIB_VERSION ............ : 0.5.0
-- GENERIC_LIB_SOVERSION .......... : 99.99
--
-- ---------------------------------
-- ---     Build Configuration    ---
-- ---------------------------------
-- Modules to build ............... : airrac
-- Libraries to build ............. : airraclib
-- Binaries to build .............. : airrac
-- Modules to test ................ : airrac
-- Binaries to test ............... : YieldTestSuitetst
--
-- * Module ....................... : airrac
--   + Layers to be built ......... : .;basic;bom;factory;command;service
--   + Dependencies on other layers :
--   + Libraries to be built ...... : airraclib
--   + Executables to be built .... : airrac
--   + Test to be checked ......... : YieldTestSuitetst
--
-- BUILD_SHARED_LIBS .............. : ON
-- CMAKE_BUILD_TYPE ............... : Debug
-- CMAKE_MODULE_PATH .............. : /home/user/dev/sim/airrac/airracgithub/config/
-- CMAKE_INSTALL_PREFIX ........... : /home/user/dev/deliveries/airrac-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/user/dev/deliveries/airrac-0.5.0/lib64
-- INSTALL_BIN_DIR ................ : /home/user/dev/deliveries/airrac-0.5.0/bin
-- INSTALL_INCLUDE_DIR ............ : /home/user/dev/deliveries/airrac-0.5.0/include
-- INSTALL_DATA_DIR ............... : /home/user/dev/deliveries/airrac-0.5.0/share
-- INSTALL_SAMPLE_DIR ............. : /home/user/dev/deliveries/airrac-0.5.0/share/airrac/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/user/dev/sim/airrac/airracgithub/README
-- CPACK_RESOURCE_FILE_LICENSE .... : /home/user/dev/sim/airrac/airracgithub/COPYING
-- CPACK_GENERATOR ................ : TBZ2
-- CPACK_DEBIAN_PACKAGE_DEPENDS ... :
-- CPACK_SOURCE_GENERATOR ......... : TBZ2;TGZ
-- CPACK_SOURCE_PACKAGE_FILE_NAME . : airrac-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 .. : program_options;date_time;iostreams;serialization;filesystem;unit_te
--   - Boost required libraries ... : optimized;/usr/lib64/libboost_iostreams-mt.so;debug;/usr/lib64/libbo
--
-- * MySQL:
--   - MYSQL_VERSION ..... : 5.5.14
--   - MYSQL_INCLUDE_DIR ..... : /usr/include/mysql
--   - MYSQL_LIBRARIES ..... : /usr/lib64/mysql/libmysqlclient.so
--
-- * SOCI:
--   - SOCI_VERSION ..... : 3.0.0
--   - SOCI_INCLUDE_DIR ..... : /usr/include/soci
--   - SOCIMYSQL_INCLUDE_DIR ..... : /usr/include/soci
--   - SOCI_LIBRARIES ..... : /usr/lib64/libsoci_core.so
--   - SOCIMYSQL_LIBRARIES ..... : /usr/lib64/libsoci_mysql.so
--
-- * StdAir:
--   - STDAIR_VERSION ..... : 0.36.2
--   - STDAIR_BINARY_DIRS ..... : /home/user/dev/deliveries/stdair-0.36.2/bin
--   - STDAIR_EXECUTABLES ..... : stdair
--   - STDAIR_LIBRARY_DIRS ..... : /home/user/dev/deliveries/stdair-0.36.2/lib64
--   - STDAIR_LIBRARIES ..... : stdairlib;stdairuicllib
--   - STDAIR_INCLUDE_DIRS ..... : /home/user/dev/deliveries/stdair-0.36.2/include
--   - STDAIR_SAMPLE_DIR ..... : /home/user/dev/deliveries/stdair-0.36.2/share/stdair/samples
--
-- Change a value with: cmake -D<Variable>=<Value>
-- ================================================================
--
-- Configuring done
-- Generating done
-- Build files have been written to: /home/user/dev/sim/airrac/airracgithub/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_airrac
[ 90%] Built target airraclib
[100%] Built target YieldTestSuitetst
Test project /home/user/dev/sim/airrac/airracgithub/build/test/airrac
    Start 1: YieldTestSuitetst
1/1 Test #1: YieldTestSuitetst .................   Passed    0.03 sec

100% tests passed, 0 tests failed out of 1

Total Test time (real) =   0.23 sec
[100%] Built target check_airractst
[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/airracgit
rm -rf build && mkdir build
cd build
```

to remove everything.

10 Linking with AirRAC

10.1 Table of Contents

10.2 Introduction

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

10.3 Using the pkg-config command

'pkg-config' is a helper tool used when compiling applications and libraries. It helps you insert the correct compiler and linker options. The syntax of the 'pkg-config' is as follows:

```
pkg-config <options> <library_name>
```

For instance, assuming that you need to compile an AirRAC based program 'my_prog.cpp', you should use the following command:

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

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

10.4 Using the airrac-config script

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

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

```
airrac-config --help
```

If the 'airrac-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
```

10.5 M4 macro for the GNU Autotools

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

- 'AirRAC_VERSION' (e.g., defined to 0.23.0)
- 'AirRAC_CFLAGS' (e.g., defined to '-I${prefix}/include')
- 'AirRAC_LIBS' (e.g., defined to '-L${prefix}/lib -lairrac')

10.6 Using AirRAC with dynamic linking

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

11 Test Rules

This section describes how the functionality of the AirRAC library should be verified. In the 'test/airrac' subdirectory, test source files are provided. All functionality should be tested using these test source files.

11.1 The Test Source Files

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

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

The test source file should be placed in the `'test/airrac'` subdirectory and should have a name ending with `'TestSuite.cpp'`.

11.2 The Reference File

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

11.3 Testing AirRAC Library

One can compile and execute all test programs from the `'test/airrac'` sub-directory by typing:

```
% make check
```

after successful compilation of the AirRAC library.

12 Users Guide

12.1 Table of Contents

12.2 Introduction

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

12.3 Get Started

12.3.1 Get the AirRAC library

12.3.2 Build the AirRAC project

To run the configuration script the first time, go to the top directory (where the AirRAC package has been un-packed), and issue either of the following two commands, depending on whether the AirRAC project has been checked out from the Subversion repository or downloaded as a tar-ball package from the Sourceforge Web site:

- `./autogen.sh`
- `./configure`

12.3.3 Build and Run the Tests

12.3.4 Install the AirRAC Project (Binaries, Documentation)

12.4 Exploring the Predefined BOM Tree

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

12.4.1 Yield Rule Engine BOM Tree

- `AIRRAC::YieldRuleStruct`

12.5 Extending the BOM Tree

13 Supported Systems

13.1 Table of Contents

13.2 Introduction

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

13.3 AirRAC 0.1.x

13.3.1 Linux Systems

13.3.1.1 Fedora Core 4 with ATLAS

- **Platform**: Intel Pentium 4
- **Operating System**: Fedora Core 4 (x86)
- **Compiler**: g++ (GCC) 4.0.2 20051125
- **AirRAC release**: 0.1.0
- **External Libraries**: From FC4 distribution:
    - `fftw3.i386-3.0.1-3`
    - `fftw3-devel.i386-3.0.1-3`
    - `atlas-sse2.i386-3.6.0-8.fc4`
    - `atlas-sse2-devel.i386-3.6.0-8.fc4`
    - `blas.i386-3.0-35.fc4`
    - `lapack.i386-3.0-35.fc4`
- **Tests Status**: All tests PASSED
- **Comments**: AirRAC configured with:

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

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

13.3.1.2 Gentoo Linux with ACML

- **Platform**: AMD Sempron 3000+
- **Operating System**: Gentoo Linux 2006.0 (x86 arch)
- **Compiler(s)**: g++ (GCC) 3.4.5
- **AirRAC release**: 0.1.1
- **External Libraries**: Compiled and installed from portage tree:
  - `sci-libs/acml-3.0.0`
- **Tests Status**: All tests PASSED
- **Comments**: BLAS and LAPACK libs set by using the following system commands:

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

AirRAC configured with:

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

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

13.3.1.3 Gentoo Linux with ATLAS

- **Platform**: Intel Pentium M Centrino
- **Operating System**: Gentoo Linux 2006.0 (x86)
- **Compiler**: g++ (GCC) 3.4.5
- **AirRAC release**: 0.1.1
- **External Libraries**: Compiled and installed from portage tree:
  - `sci-libs/fftw-3.1`
  - `sci-libs/blas-atlas-3.6.0-r1`
  - `sci-libs/lapack-atlas-3.6.0`
- **Tests Status**: All tests PASSED
- **Comments**: BLAS and LAPACK libs set by using the following system commands:

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

AirRAC configured with:

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

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

13.3.1.4 Gentoo Linux with MKL

- **Platform**: Intel Pentium M Centrino
- **Operating System**: Gentoo Linux 2006.0 (x86 arch)
- **Compiler**: g++ (GCC) 3.4.5
- **AirRAC release**: 0.1.0
- **External Libraries**: Intel Math Kernel Library (MKL) 8.0.1 installed manually in the following directory: `/opt/intel/mkl/8.0.1`
- **Tests Status**: All tests PASSED
- **Comments**: AirRAC configured using the following commands:

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

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

13.3.1.5 Gentoo Linux with NetLib's BLAS and LAPACK

- **Platform**: Intel Pentium M Centrino
- **Operating System**: Gentoo Linux 2006.0 (x86)
- **Compiler**: g++ (GCC) 3.4.5
- **AirRAC release**: 0.1.1
- **External Libraries**: Compiled and installed from portage tree:
  - `sci-libs/fftw-3.1`
  - `sci-libs/blas-reference-19940131-r2`
  - `sci-libs/cblas-reference-20030223`
  - `sci-libs/lapack-reference-3.0-r2`
- **Tests Status**: All tests PASSED
- **Comments**: BLAS and LAPACK libs set by using the following system commands:

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

  AirRAC configured with:

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

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

13.3.1.6 Red Hat Enterprise Linux with AirRAC External

- **Platform**: Intel Pentium 4
- **Operating System**: Red Hat Enterprise Linux AS release 4 (Nahant Update 2)
- **Compiler**: g++ (GCC) 3.4.4 20050721 (Red Hat 3.4.4-2)
- **AirRAC release**: 0.1.0
- **External Libraries**: BLAS, CBLAS, LAPACK and FFTW libraries from AirRAC External 2.1.1 package
- **Tests Status**: All tests PASSED
- **Date**: March 7, 2006
- **Tester**: Erik G. Larsson

13.3.1.7 SUSE Linux 10.0 with NetLib's BLAS and LAPACK

- **Platform**: Intel Pentium 4 CPU 3.20GHz (64-bit)
- **Operating System**: SUSE Linux 10.0 (x86_64)
- **Compiler(s)**: g++ (GCC) 4.0.2
- **AirRAC release**: 0.1.0
- **External Libraries**: BLAS, LAPACK and FFTW libraries installed from OpenSuse 10.0 RPM repository:
  - `blas-3.0-926`
  - `lapack-3.0-926`
  - `fftw3-3.0.1-114`
  - `fftw3-threads-3.0.1-114`
  - `fftw3-devel-3.0.1-114`
- **Tests Status**: All tests PASSED
- **Comments**: AirRAC configured with:

```
% export CXXFLAGS="-m64 -march=nocona -O3 -pipe"
% ./configure --with-lapack="/usr/lib64/liblapack.so.3"
```

- **Date**: March 1, 2006
- **Tester**: Adam Piatyszek (ediap)

13.3.1.8 SUSE Linux 10.0 with MKL

- **Platform**: Intel Pentium 4 CPU 3.20GHz (64-bit)
- **Operating System**: SUSE Linux 10.0 (x86_64)
- **Compiler(s)**: g++ (GCC) 4.0.2
- **AirRAC release**: 0.1.0
- **External Libraries**: Intel Math Kernel Library (MKL) 8.0.1 installed manually in the following directory: `/opt/intel/mkl/8.0.1`
- **Tests Status**: All tests PASSED
- **Comments**: AirRAC configured with:

```
% export CXXFLAGS="-m64 -march=nocona -O3 -pipe"
% export LDFLAGS="-L/opt/intel/mkl/8.0.1/lib/em64t"
% export CPPFLAGS="-I/opt/intel/mkl/8.0.1/include"
% ./configure
```

- **Date**: March 1, 2006
- **Tester**: Adam Piatyszek (ediap)

13.3.2 Windows Systems

13.3.2.1 Microsoft Windows XP with Cygwin

- **Platform**: AMD Sempron 3000+
- **Operating System**: Microsoft Windows XP SP2, Cygwin 1.5.19-4
- **Compiler(s)**: g++ (GCC) 3.4.4 (cygming special)
- **AirRAC release**: 0.1.1
- **External Libraries**: Installed from Cygwin's repository:
  - `fftw-3.0.1-2`
  - `fftw-dev-3.0.1-1`
  - `lapack-3.0-4`
- **Tests Status**: All tests PASSED
- **Comments**: Only static library can be built. AirRAC configured with:

```
% ./configure
```

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

13.3.2.2 Microsoft Windows XP with Cygwin and ATLAS

- **Platform**: AMD Sempron 3000+
- **Operating System**: Microsoft Windows XP SP2, Cygwin 1.5.19-4
- **Compiler(s)**: g++ (GCC) 3.4.4 (cygming special)
- **AirRAC release**: 0.1.1
- **External Libraries**: Installed from Cygwin's repository:
  - `fftw-3.0.1-2`
  - `fftw-dev-3.0.1-1`

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

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

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

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

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

13.3.2.3 Microsoft Windows XP with Cygwin and ACML

- **Platform**: AMD Sempron 3000+
- **Operating System**: Microsoft Windows XP SP2, Cygwin 1.5.19-4
- **Compiler(s)**: g++ (GCC) 3.4.4 (cygming special)
- **AirRAC release**: 0.1.2
- **External Libraries**: ACML version 3.1.0 (`acml3.1.0-32-win32-g77.exe`) installed into a default directory, i.e. `"c:\Program Files\AMD\acml3.1.0"`
- **Tests Status**: All tests PASSED
- **Comments**: Only static library can be built. AirRAC configured with:

```
% export LDFLAGS="-L/cygdrive/c/Progra~1/AMD/acml3.1.0/gnu32/lib"
% export CPPFLAGS="-I/cygdrive/c/Progra~1/AMD/acml3.1.0/gnu32/include"
% ./configure --enable-debug
```

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

13.3.2.4 Microsoft Windows XP with MinGW, MSYS and ACML

- **Platform**: AMD Sempron 3000+
- **Operating System**: Microsoft Windows XP SP2, MinGW 5.0.2, MSYS 1.0.10
- **Compiler(s)**: g++ (GCC) 3.4.4 (mingw special)
- **AirRAC release**: 0.1.2
- **External Libraries**: ACML version 3.1.0 (`acml3.1.0-32-win32-g77.exe`) installed into a default directory, i.e. `"c:\Program Files\AMD\acml3.1.0"`
- **Tests Status**: All tests PASSED
- **Comments**: Only static library can be built. AirRAC configured with:

```
% export LDFLAGS="-L/c/Progra~1/AMD/acml3.1.0/gnu32/lib"
% export CPPFLAGS="-I/c/Progra~1/AMD/acml3.1.0/gnu32/include"
% ./configure --enable-debug
```

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

13.3.2.5 Microsoft Windows XP with MinGW, MSYS and AirRAC External

- **Platform**: AMD Sempron 3000+
- **Operating System**: Microsoft Windows XP SP2, MinGW 5.0.2, MSYS 1.0.10
- **Compiler(s)**: g++ (GCC) 3.4.4 (mingw special)
- **AirRAC release**: 0.1.5
- **External Libraries**: BLAS, CBLAS, LAPACK and FFTW libraries from AirRAC External 2.2.0 package
- **Tests Status**: All tests PASSED
- **Comments**: Only static library can be built. AirRAC configured with:

```
% export LDFLAGS="-L/usr/local/lib"
% export CPPFLAGS="-I/usr/local/include"
% export CXXFLAGS="-Wall -O3 -march=athlon-tbird -pipe"
% ./configure --disable-html-doc
```

- **Date**: August 11, 2006
- **Tester**: Adam Piatyszek (ediap)

13.3.2.6 Microsoft Windows XP with MS Visual C++ and Intel MKL

- **Platform**: AMD Sempron 3000+
- **Operating System**: Microsoft Windows XP SP2
- **Compiler(s)**: Microsoft Visual C++ 2005 .NET
- **AirRAC release**: 0.1.5
- **External Libraries**: Intel Math Kernel Library (MKL) 8.1 installed manually in the following directory: `"C:\Program Files\Intel\MKL\8.1"`
- **Tests Status**: Not fully tested. Some AirRAC based programs compiled and run with success.
- **Comments**: Only static library can be built. AirRAC built by opening the `"win32\airrac.vcproj"` project file in MSVC++ and executing `"Build -> Build Solution"` command from menu.
- **Date**: August 11, 2006
- **Tester**: Adam Piatyszek (ediap)

13.3.3 Unix Systems

13.3.3.1 SunOS 5.9 with AirRAC External

- **Platform**: SUNW, Sun-Blade-100 (SPARC)
- **Operating System**: SunOS 5.9 Generic_112233-10
- **Compiler(s)**: g++ (GCC) 3.4.5
- **AirRAC release**: 0.1.2
- **External Libraries**: BLAS, CBLAS, LAPACK and FFTW libraries from AirRAC External 2.1.1 package. The following configuration command has been used:

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

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

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

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

14 AirRAC Supported Systems (Previous Releases)

14.1 AirRAC 3.9.1

14.2 AirRAC 3.9.0

14.3 AirRAC 3.8.1

15 Tutorials

15.1 Table of Contents

15.2 Introduction

This page contains some tutorial examples that will help you getting started using AirRAC. Most examples show how to construct some simple business objects, i.e., instances of the so-named Business Object Model (BOM).

15.2.1 Preparing the AirRAC Project for Development

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

15.3 Build a Predefined BOM Tree

A few steps:

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

15.3.1 Instanciate the BOM Root Object

First, a BOM root object (i.e., a root for all the classes in the project) is instantiated by the `airrac::AIRRAC_ServiceContext` context object, when the `airrac::AIRRAC_Service` is itself instantiated. The corresponding `AirRAC` type (class) is `airrac::BomRoot`.

In the following sample, that object is named `ioBomRoot`, and is given as input/output parameter of the `airrac::CmdBomManager::buildSampleBom()` method:

15.3.2 Instanciate the (Airline) Inventory Object

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

Thanks to that key, an airline inventory object, i.e. of type (class) `airrac::Inventory`, can be instantiated. Let us name that airline inventory object `lBAInv`.

15.3.3 Link the Inventory Object with the BOM Root

Then, both objects have to be linked: the airline inventory object (`airrac::Inventory`) has to be linked with the root of the BOM tree (`airrac::BomRoot`). That operation is as simple as using the `airrac::FacBomManager::addToListAndMap()` method:

15.3.4 Build Another Airline Inventory

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

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

15.3.5 Dump The BOM Tree Content

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

... and browsed:

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

15.3.6 Result of the Tutorial Program

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

```
[D]/home/user/dev/sim/airrac/airracgithub/airrac/batches/airrac.cpp:184: Welcome
      to AirRAC
[D]/home/user/dev/sim/airrac/airracgithub/airrac/command/YieldParserHelper.cpp:49
      3: Parsing yield input file: /home/user/dev/deliveries/stdair-0.36.2/share/stdair
      /samples/yieldstore01.csv
[D]/home/user/dev/sim/airrac/airracgithub/airrac/command/YieldParserHelper.cpp:32
      6: YieldRule: 1, SIN-BKK (ALL), DC, [2010-Jan-15/2011-Jan-01] - [00:00:00/23:59:0
      0], Y, 200 EUR, SQ / Y
[D]/home/user/dev/sim/airrac/airracgithub/airrac/command/YieldParserHelper.cpp:32
      6: YieldRule: 2, SIN-HND (ALL), DC, [2010-Jan-15/2011-Jan-01] - [00:00:00/23:59:0
      0], Y, 200 EUR, SQ / Y
[D]/home/user/dev/sim/airrac/airracgithub/airrac/command/YieldParserHelper.cpp:32
      6: YieldRule: 3, SIN-NCE (ALL), DC, [2010-Jan-15/2011-Jan-01] - [00:00:00/23:59:0
      0], Y, 1200 EUR, SQ / Y - AF / YLMN
[D]/home/user/dev/sim/airrac/airracgithub/airrac/command/YieldParserHelper.cpp:32
      6: YieldRule: 4, SIN-BKK (ALL), DC, [2010-Jan-15/2011-Jan-01] - [00:00:00/23:59:0
      0], Y, 300 EUR, SQ / Y
[D]/home/user/dev/sim/airrac/airracgithub/airrac/command/YieldParserHelper.cpp:32
      6: YieldRule: 5, SIN-HND (ALL), DC, [2010-Jan-15/2011-Jan-01] - [00:00:00/23:59:0
      0], Y, 300 EUR, SQ / Y
[D]/home/user/dev/sim/airrac/airracgithub/airrac/command/YieldParserHelper.cpp:32
      6: YieldRule: 6, SIN-NCE (ALL), DC, [2010-Jan-15/2011-Jan-01] - [00:00:00/23:59:0
      0], Y, 1500 EUR, SQ / Y - AF / YLMN
[D]/home/user/dev/sim/airrac/airracgithub/airrac/command/YieldParserHelper.cpp:54
      1: Parsing of yield input file: /home/user/dev/deliveries/stdair-0.36.2/share/std
      air/samples/yieldstore01.csv succeeded
[D]/home/user/dev/sim/airrac/airracgithub/airrac/batches/airrac.cpp:205: BOM tree
      :
================================================================
BomRoot:  -- ROOT --
================================================================
++++++++++++++++++++++++++++++++++++++++++++++++
AirportPair: SIN, BKK
++++++++++++++++++++++++++++++++++++++++++++++++
------------------------------------------
DatePeriod: [2010-Jan-15/2011-Jan-01]
------------------------------------------
******************************************
PosChannel: ALL, DC
******************************************
----------------------------------------
TimePeriod: 00:00:00-23:59:00
----------------------------------------
--------------------------------------
Fare/yield-Features: OW -- Y
--------------------------------------
-------------------------------------
AirlineClassList: SQ Y
-------------------------------------
-------------------------------------
Fare/yield-Features: RT -- Y
--------------------------------------
-------------------------------------
AirlineClassList: SQ Y
-------------------------------------
++++++++++++++++++++++++++++++++++++++++++++++++
AirportPair: SIN, HND
++++++++++++++++++++++++++++++++++++++++++++++++
------------------------------------------
DatePeriod: [2010-Jan-15/2011-Jan-01]
------------------------------------------
******************************************
PosChannel: ALL, DC
```

```
****************************************
----------------------------------------
TimePeriod: 00:00:00-23:59:00
----------------------------------------
----------------------------------------
Fare/yield-Features: OW -- Y
----------------------------------------
----------------------------------------
AirlineClassList: SQ Y
----------------------------------------
----------------------------------------
Fare/yield-Features: RT -- Y
----------------------------------------
----------------------------------------
AirlineClassList: SQ Y
----------------------------------------
++++++++++++++++++++++++++++++++++++++++++++++++
AirportPair: SIN, NCE
++++++++++++++++++++++++++++++++++++++++++++++++
----------------------------------------
DatePeriod: [2010-Jan-15/2011-Jan-01]
----------------------------------------
****************************************
PosChannel: ALL, DC
****************************************
----------------------------------------
TimePeriod: 00:00:00-23:59:00
----------------------------------------
----------------------------------------
Fare/yield-Features: OW -- Y
----------------------------------------
----------------------------------------
AirlineClassList: SQ Y, AF YLMN
----------------------------------------
----------------------------------------
Fare/yield-Features: RT -- Y
----------------------------------------
----------------------------------------
AirlineClassList: SQ Y, AF YLMN
----------------------------------------

[D]/home/user/dev/sim/airrac/airracgithub/airrac/batches/airrac.cpp:210: Travel s
      olutions:
    [0] [0] BA, 9, 2011-06-10, LHR, SYD, 21:45 --- Q, 900, 1 1 1 --- [0] Q:8
```

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

16 Command-Line Test to Demonstrate How To Test the AirRAC 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 YieldTestSuite
```

```
#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/service/Logger.hpp>
// Airrac
#include <airrac/AIRRAC_Service.hpp>
#include <airrac/config/airrac-paths.hpp>

namespace boost_utf = boost::unit_test;

// (Boost) Unit Test XML Report
std::ofstream utfReportStream ("YieldTestSuite_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_tes
      ts);
  }

  ~UnitTestConfig() {
  }
};

// ////////////////////////////////////////////////////////////////////

void testYieldQuoterHelper (const unsigned short iTestFlag,
                            const stdair::Filename_T iYieldInputFilename,
                            const bool isBuiltin) {

  // Output log File
  std::ostringstream oStr;
  oStr << "FQTTestSuite_" << 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 AirRAC service object
  const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG,
                                         logOutputFile);

  // Initialise the AirRAC service object
  AIRRAC::AIRRAC_Service airracService (lLogParams);

  // Build a sample list of travel solutions
  stdair::TravelSolutionList_T lTravelSolutionList;
  airracService.buildSampleTravelSolutions (lTravelSolutionList);

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

    // Build the default sample BOM tree (filled with yields) for AirRAC
    airracService.buildSampleBom();

  } else {

    // Build the BOM tree from parsing the yield input file
    AIRRAC::YieldFilePath lYieldFilePath (iYieldInputFilename);
```

```
    airracService.parseAndLoad (lYieldFilePath);
  }

  // Calculate the yields for the given travel solution
  airracService.calculateYields (lTravelSolutionList);

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

}


// /////////////// 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 (airrac_simple_yield) {

  // Input file name
  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;

  // Try to yieldQuote the sample default list of travel solutions
  BOOST_CHECK_NO_THROW (testYieldQuoterHelper (0, lYieldInputFilename, isBuiltin)
      );

}

BOOST_AUTO_TEST_CASE (airrac_error_parsing_input_file) {

  // Input file name
  const stdair::Filename_T lYieldInputFilename (STDAIR_SAMPLE_DIR "/yieldstoreErr
      or01.csv");

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

  // Try to yield quote the sample default list of travel solutions
  BOOST_CHECK_THROW (testYieldQuoterHelper (1, lYieldInputFilename, isBuiltin),
                     AIRRAC::YieldFileParsingFailedException);
}

BOOST_AUTO_TEST_CASE (airrac_error_missing_input_file) {

  // Input file name
  const stdair::Filename_T lYieldInputFilename (STDAIR_SAMPLE_DIR "/missingFile.c
      sv");

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

  // Try to yield quote the sample default list of travel solutions
  BOOST_CHECK_THROW (testYieldQuoterHelper (2, lYieldInputFilename, isBuiltin),
                     AIRRAC::YieldInputFileNotFoundException);
}

BOOST_AUTO_TEST_CASE (airrac_simple_yield_built_in) {

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

```
  const bool isBuiltin = true;

  // Try to yield quote the sample default list of travel solutions
  BOOST_CHECK_NO_THROW (testYieldQuoterHelper (3, " ", isBuiltin));

}

// End the test suite
BOOST_AUTO_TEST_SUITE_END()

/*!
```

17 Directory Hierarchy

17.1 Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

18 Namespace Index

18.1 Namespace List

Here is a list of all namespaces with brief descriptions:

19 Class Index

19.1 Class Hierarchy

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

20 Class Index

20.1 Class List

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

21 File Index

21.1 File List

Here is a list of all files with brief descriptions:

22 Directory Documentation

22.1 test/airrac/ Directory Reference

Files

- file YieldTestSuite.cpp
- file YieldTestSuite.hpp

22.2 airrac/ Directory Reference

Directories

- directory basic
- directory batches
- directory bom
- directory command
- directory config
- directory factory
- directory service

Files

- file AIRRAC_Service.hpp
- file AIRRAC_Types.hpp

22.3 airrac/basic/ Directory Reference

Files

- file BasConst.cpp
- file BasConst_AIRRAC_Service.hpp
- file BasConst_General.hpp

22.4 airrac/batches/ Directory Reference

Files

- file airrac.cpp

22.5 airrac/bom/ Directory Reference

Files

- file YieldRuleStruct.cpp
- file YieldRuleStruct.hpp

22.6 airrac/command/ Directory Reference

Files

- file YieldManager.cpp
- file YieldManager.hpp
- file YieldParser.cpp
- file YieldParser.hpp
- file YieldParserHelper.cpp
- file YieldParserHelper.hpp
- file YieldRuleGenerator.cpp
- file YieldRuleGenerator.hpp

22.7 airrac/config/ Directory Reference

Files

- file airrac-paths.hpp

22.8 airrac/factory/ Directory Reference

Files

- file FacAirracServiceContext.cpp
- file FacAirracServiceContext.hpp

22.9 airrac/service/ Directory Reference

Files

- file AIRRAC_Service.cpp
- file AIRRAC_ServiceContext.cpp
- file AIRRAC_ServiceContext.hpp

22.10 test/ Directory Reference

Directories

- directory airrac

23 Namespace Documentation

23.1 AIRRAC Namespace Reference

Namespaces

- namespace YieldParserHelper

Classes

- class AIRRAC_Service

  Interface for the AIRRAC Services.

- class AirportPairNotFoundException
- class PosOrChannelNotFoundException
- class FlightDateNotFoundException
- class FlightTimeNotFoundException
- class FeaturesNotFoundException
- class AirlineNotFoundException
- class YieldInputFileNotFoundException
- class YieldFileParsingFailedException
- class QuotingException
- class YieldFilePath
- struct YieldRuleStruct

  Utility Structure for the parsing of Flight-Date structures.

- class YieldManager

  Command wrapping the travel request process.

- class YieldParser

  Class wrapping the parser entry point.

- class YieldFileParser
- class YieldRuleGenerator
- class FacAirracServiceContext

  Factory for the service context.

- class AIRRAC_ServiceContext

  Inner class holding the context for the AIRRAC Service object.

Typedefs

- typedef boost::shared_ptr< AIRRAC_Service > AIRRAC_ServicePtr_T
- typedef unsigned int YieldID_T

Variables

- const std::string DEFAULT_AIRLINE_CODE = "BA"

23.1.1 Typedef Documentation

23.1.1.1 typedef boost::shared_ptr<AIRRAC_Service> AIRRAC::AIRRAC_ServicePtr_T

Definition at line 95 of file AIRRAC_Types.hpp.

23.1.1.2 typedef unsigned int AIRRAC::YieldID_T

ID for the Yield Quote system.

Definition at line 102 of file AIRRAC_Types.hpp.

23.1.2 Variable Documentation

23.1.2.1 const std::string AIRRAC::DEFAULT_AIRLINE_CODE = "BA"

Default airline name for the AIRRAC_Service.

Definition at line 10 of file BasConst.cpp.

23.2 AIRRAC::YieldParserHelper Namespace Reference

Classes

- struct YieldRuleParser
- struct ParserSemanticAction
- struct storeYieldId
- struct storeOrigin
- struct storeDestination
- struct storeTripType
- struct storeDateRangeStart
- struct storeDateRangeEnd
- struct storeStartRangeTime
- struct storeEndRangeTime
- struct storePOS
- struct storeCabinCode
- struct storeChannel
- struct storeYield
- struct storeAirlineCode
- struct storeClass
- struct doEndYield

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

23.2.1 Variable Documentation

23.2.1.1 stdair::int1_p_t AIRRAC::YieldParserHelper::int1_p

Namespaces. 1-digit-integer parser

Definition at line 341 of file YieldParserHelper.cpp.

23.2.1.2 stdair::uint2_p_t AIRRAC::YieldParserHelper::uint2_p

2-digit-integer parser

Definition at line 344 of file YieldParserHelper.cpp.

23.2.1.3 stdair::uint4_p_t AIRRAC::YieldParserHelper::uint4_p

4-digit-integer parser

Definition at line 347 of file YieldParserHelper.cpp.

23.2.1.4 stdair::uint1_4_p_t AIRRAC::YieldParserHelper::uint1_4_p

Up-to-4-digit-integer parser

Definition at line 350 of file YieldParserHelper.cpp.

Referenced by AIRRAC::YieldParserHelper::YieldRuleParser::YieldRuleParser().

23.2.1.5 stdair::hour_p_t AIRRAC::YieldParserHelper::hour_p

Time element parsers.

Definition at line 353 of file YieldParserHelper.cpp.

Referenced by AIRRAC::YieldParserHelper::YieldRuleParser::YieldRuleParser().

23.2.1.6 stdair::minute_p_t AIRRAC::YieldParserHelper::minute_p

Definition at line 354 of file YieldParserHelper.cpp.

Referenced by AIRRAC::YieldParserHelper::YieldRuleParser::YieldRuleParser().

23.2.1.7 stdair::second_p_t AIRRAC::YieldParserHelper::second_p

Definition at line 355 of file YieldParserHelper.cpp.

Referenced by AIRRAC::YieldParserHelper::YieldRuleParser::YieldRuleParser().

23.2.1.8 stdair::year_p_t AIRRAC::YieldParserHelper::year_p

Date element parsers.

Definition at line 358 of file YieldParserHelper.cpp.

Referenced by AIRRAC::YieldParserHelper::YieldRuleParser::YieldRuleParser().

23.2.1.9 stdair::month_p_t AIRRAC::YieldParserHelper::month_p

Definition at line 359 of file YieldParserHelper.cpp.

Referenced by AIRRAC::YieldParserHelper::YieldRuleParser::YieldRuleParser().

23.2.1.10 stdair::day_p_t AIRRAC::YieldParserHelper::day_p

Definition at line 360 of file YieldParserHelper.cpp.

Referenced by AIRRAC::YieldParserHelper::YieldRuleParser::YieldRuleParser().

23.3 stdair Namespace Reference

Forward declarations.

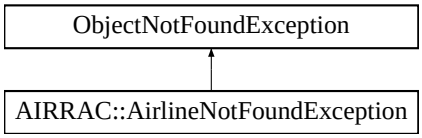
23.3.1 Detailed Description

Forward declarations.

24 Class Documentation

24.1 AIRRAC::AirlineNotFoundException Class Reference

`#include <airrac/AIRRAC_Types.hpp>`Inheritance diagram for AIRRAC::AirlineNotFoundException::



Public Member Functions

- AirlineNotFoundException (const std::string &iWhat)

24.1.1 Detailed Description

Definition at line 54 of file AIRRAC_Types.hpp.

24.1.2 Constructor & Destructor Documentation

24.1.2.1 AIRRAC::AirlineNotFoundException::AirlineNotFoundException (const std::string & *iWhat*) [inline]

Constructor.

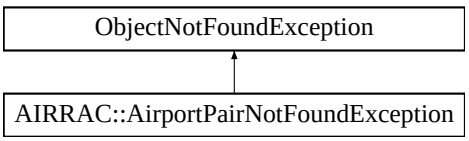
Definition at line 57 of file AIRRAC_Types.hpp.

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

- airrac/AIRRAC_Types.hpp

24.2 AIRRAC::AirportPairNotFoundException Class Reference

`#include <airrac/AIRRAC_Types.hpp>`Inheritance diagram for AIRRAC::AirportPairNotFoundException::



Public Member Functions

- AirportPairNotFoundException (const std::string &iWhat)

24.2.1 Detailed Description

Definition at line 19 of file AIRRAC_Types.hpp.

24.2.2 Constructor & Destructor Documentation

24.2.2.1 AIRRAC::AirportPairNotFoundException::AirportPairNotFoundException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 22 of file AIRRAC_Types.hpp.

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

- airrac/AIRRAC_Types.hpp

24.3 AIRRAC::AIRRAC_Service Class Reference

Interface for the AIRRAC Services.

`#include <airrac/AIRRAC_Service.hpp>`

Public Member Functions

- AIRRAC_Service (const stdair::BasLogParams &)
- AIRRAC_Service (const stdair::BasLogParams &, const stdair::BasDBParams &)
- AIRRAC_Service (stdair::STDAIR_ServicePtr_T ioSTDAIR_ServicePtr)
- void parseAndLoad (const YieldFilePath &iYieldFilename)
- ~AIRRAC_Service ()
- void calculateYields (stdair::TravelSolutionList_T &)
- void updateYields (stdair::BomRoot &)
- void buildSampleBom ()
- void clonePersistentBom ()
- void buildComplementaryLinks (stdair::BomRoot &)
- void buildSampleTravelSolutions (stdair::TravelSolutionList_T &)
- std::string csvDisplay () const
- std::string csvDisplay (const stdair::TravelSolutionList_T &) const

24.3.1 Detailed Description

Interface for the AIRRAC Services.

Definition at line 30 of file AIRRAC_Service.hpp.

24.3.2 Constructor & Destructor Documentation

24.3.2.1 AIRRAC::AIRRAC_Service::AIRRAC_Service (const stdair::BasLogParams & *iLogParams*)

Constructor.

The initAirracService() 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 34 of file AIRRAC_Service.cpp.

24.3.2.2 AIRRAC::AIRRAC_Service::AIRRAC_Service (const stdair::BasLogParams & *iLogParams*, const stdair::BasDBParams & *iDBParams*)

Constructor.

The initAirracService() 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.

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

Definition at line 54 of file AIRRAC_Service.cpp.

24.3.2.3 AIRRAC::AIRRAC_Service::AIRRAC_Service (stdair::STDAIR_ServicePtr_T *ioSTDAIR_ServicePtr*)

Constructor.

The initAirracService() 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 AIRRAC_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 76 of file AIRRAC_Service.cpp.

24.3.2.4 AIRRAC::AIRRAC_Service::~AIRRAC_Service ()

Destructor.

Definition at line 92 of file AIRRAC_Service.cpp.

24.3.3 Member Function Documentation

24.3.3.1 void AIRRAC::AIRRAC_Service::parseAndLoad (const YieldFilePath & *iYieldFilename*)

Parse the yield input file, 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 YieldFilePath& Filename of the input yield file.

Definition at line 170 of file AIRRAC_Service.cpp.

References buildComplementaryLinks(), clonePersistentBom(), and AIRRAC::YieldParser::generateYieldStore().

Referenced by main().

24.3.3.2 void AIRRAC::AIRRAC_Service::calculateYields (stdair::TravelSolutionList_T & *ioTravelSolutionList*)

Calculate/retrieve a yield.

Definition at line 402 of file AIRRAC_Service.cpp.

24.3.3.3 void AIRRAC::AIRRAC_Service::updateYields (stdair::BomRoot & *ioBomRoot*)

Update the yields for booking classes and O&D.

Definition at line 433 of file AIRRAC_Service.cpp.

24.3.3.4 void AIRRAC::AIRRAC_Service::buildSampleBom ()

Build a sample BOM tree.

For now, no object is created: the BOM tree remains empty. In the future, it will hold a sample yield store.

Definition at line 223 of file AIRRAC_Service.cpp.

References buildComplementaryLinks(), and clonePersistentBom().

Referenced by main().

24.3.3.5 void AIRRAC::AIRRAC_Service::clonePersistentBom ()

Clone the persistent sample BOM tree.

Definition at line 280 of file AIRRAC_Service.cpp.

References buildComplementaryLinks().

Referenced by buildSampleBom(), and parseAndLoad().

24.3.3.6 void AIRRAC::AIRRAC_Service::buildComplementaryLinks (stdair::BomRoot & *ioBomRoot*)

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

Note:

Do nothing for now.

Definition at line 326 of file AIRRAC_Service.cpp.

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

24.3.3.7 void AIRRAC::AIRRAC_Service::buildSampleTravelSolutions (stdair::TravelSolutionList_T & *ioTravelSolutionList*)

Build a sample list of travel solutions.

As of now (March 2011), that list is made of the following travel solutions:

- BA9
- LHR-SYD
- 2011-06-10
- Q
- WTP: 900
- Change fee: 20; Non refundable; Saturday night stay

Parameters:

TravelSolutionList_T& Sample list of travel solution structures. It should be given empty. It is altered with the returned sample.

Definition at line 332 of file AIRRAC_Service.cpp.

Referenced by main().

24.3.3.8 std::string AIRRAC::AIRRAC_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 352 of file AIRRAC_Service.cpp.

Referenced by main().

24.3.3.9 std::string AIRRAC::AIRRAC_Service::csvDisplay (const stdair::TravelSolutionList_T & *ioTravelSolutionList*) const

Display (dump in the returned string) the full list of travel solution structures.

Returns:

std::string Output string in which the list of travel solutions is logged/dumped.

Definition at line 380 of file AIRRAC_Service.cpp.

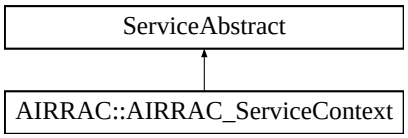
The documentation for this class was generated from the following files:

- airrac/AIRRAC_Service.hpp
- airrac/service/AIRRAC_Service.cpp

24.4 AIRRAC::AIRRAC_ServiceContext Class Reference

Inner class holding the context for the AIRRAC Service object.

`#include <airrac/service/AIRRAC_ServiceContext.hpp>`Inheritance diagram for AIRRAC::AIRRAC_ServiceContext::



Friends

- class AIRRAC_Service
- class FacAirracServiceContext

24.4.1 Detailed Description

Inner class holding the context for the AIRRAC Service object.

Definition at line 25 of file AIRRAC_ServiceContext.hpp.

24.4.2 Friends And Related Function Documentation

24.4.2.1 friend class AIRRAC_Service [friend]

The AIRRAC_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 31 of file AIRRAC_ServiceContext.hpp.

24.4.2.2 friend class FacAirracServiceContext [friend]

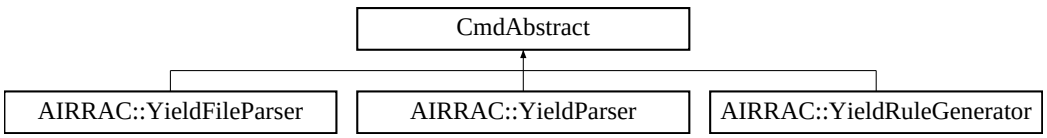
Definition at line 32 of file AIRRAC_ServiceContext.hpp.

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

- airrac/service/AIRRAC_ServiceContext.hpp
- airrac/service/AIRRAC_ServiceContext.cpp

24.5 CmdAbstract Class Reference

Inheritance diagram for CmdAbstract::

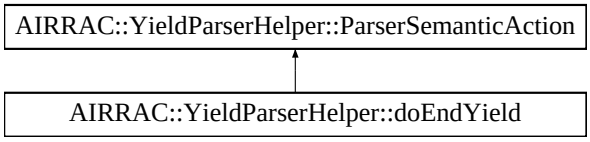


The documentation for this class was generated from the following files:

- airrac/command/YieldParserHelper.hpp
- airrac/command/YieldRuleGenerator.hpp
- airrac/command/YieldParser.hpp

24.6 AIRRAC::YieldParserHelper::doEndYield Struct Reference

`#include <airrac/command/YieldParserHelper.hpp>`Inheritance diagram for AIRRAC::YieldParserHelper::doEndYield::



Public Member Functions

- doEndYield (stdair::BomRoot &, YieldRuleStruct &)
- void operator() (boost::spirit::qi::unused_type, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- stdair::BomRoot & _bomRoot
- YieldRuleStruct & _yieldRule

24.6.1 Detailed Description

Mark the end of the yield-rule parsing.

Definition at line 178 of file YieldParserHelper.hpp.

24.6.2 Constructor & Destructor Documentation

24.6.2.1 AIRRAC::YieldParserHelper::doEndYield::doEndYield (stdair::BomRoot & *ioBomRoot*, YieldRuleStruct & *ioYieldRule*)

Actor Constructor.

Definition at line 314 of file YieldParserHelper.cpp.

24.6.3 Member Function Documentation

24.6.3.1 void AIRRAC::YieldParserHelper::doEndYield::operator() (boost::spirit::qi::unused_type, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 321 of file YieldParserHelper.cpp.

References _bomRoot, AIRRAC::YieldParserHelper::ParserSemanticAction::_yieldRule, and AIRRAC::YieldRuleStruct::describe().

24.6.4 Member Data Documentation

24.6.4.1 stdair::BomRoot& AIRRAC::YieldParserHelper::doEndYield::_bomRoot

Actor Specific Context.

Definition at line 186 of file YieldParserHelper.hpp.

Referenced by operator()().

24.6.4.2 YieldRuleStruct& AIRRAC::YieldParserHelper::ParserSemanticAction::_yieldRule [inherited]

Actor Context.

Definition at line 34 of file YieldParserHelper.hpp.

Referenced by operator()(), AIRRAC::YieldParserHelper::storeClass::operator()(), AIRRAC::YieldParserHelper::storeAirlineCode::operator()(), AIRRAC::YieldParserHelper::storeYield::operator()(), AIRRAC::YieldParserHelper::storeChannel::operator()(), AIRRAC::YieldParserHelper::storeCabinCode::operator()(), AIRRAC::YieldParserHelper::storePOS::operator()(), AIRRAC::YieldParserHelper::storeEndRangeTime::operator()(), AIRRAC::YieldParserHelper::storeStartRangeTime::operator()(), AIRRAC::YieldParserHelper::storeDateRangeEnd::operator(AIRRAC::YieldParserHelper::storeDateRangeStart::operator()(), AIRRAC::YieldParserHelper::storeTripType::operator()(), AIRRAC::YieldParserHelper::storeDestination::operator()(), AIRRAC::YieldParserHelper::storeOrigin::operator()(), and AIRRAC::YieldParserHelper::storeYieldId::operator()().

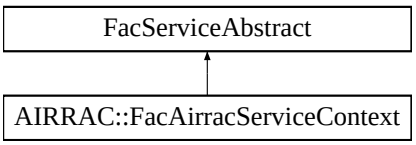
The documentation for this struct was generated from the following files:

- airrac/command/YieldParserHelper.hpp
- airrac/command/YieldParserHelper.cpp

24.7 AIRRAC::FacAirracServiceContext Class Reference

Factory for the service context.

`#include <airrac/factory/FacAirracServiceContext.hpp>`Inheritance diagram for AIRRAC::FacAirracServiceContext::



Public Member Functions

- ~FacAirracServiceContext ()
- AIRRAC_ServiceContext & create ()

Static Public Member Functions

- static FacAirracServiceContext & instance ()

Protected Member Functions

- FacAirracServiceContext ()

24.7.1 Detailed Description

Factory for the service context.

Definition at line 21 of file FacAirracServiceContext.hpp.

24.7.2 Constructor & Destructor Documentation

24.7.2.1 AIRRAC::FacAirracServiceContext::~FacAirracServiceContext ()

Destructor.

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

Definition at line 17 of file FacAirracServiceContext.cpp.

24.7.2.2 AIRRAC::FacAirracServiceContext::FacAirracServiceContext () [inline, protected]

Default Constructor.

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

Definition at line 56 of file FacAirracServiceContext.hpp.

Referenced by instance().

24.7.3 Member Function Documentation

24.7.3.1 FacAirracServiceContext & AIRRAC::FacAirracServiceContext::instance () [static]

Provide the unique instance.

The singleton is instantiated when first used.

Returns:

FacServiceContext&

Definition at line 22 of file FacAirracServiceContext.cpp.

References FacAirracServiceContext().

24.7.3.2 AIRRAC_ServiceContext & AIRRAC::FacAirracServiceContext::create ()

Create a new ServiceContext object.

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

Returns:

ServiceContext& The newly created object.

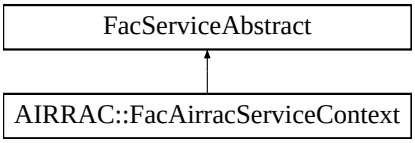
Definition at line 34 of file FacAirracServiceContext.cpp.

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

- airrac/factory/FacAirracServiceContext.hpp
- airrac/factory/FacAirracServiceContext.cpp

24.8 FacServiceAbstract Class Reference

Inheritance diagram for FacServiceAbstract::

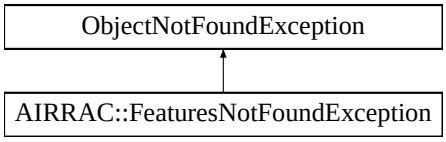


The documentation for this class was generated from the following file:

- airrac/factory/FacAirracServiceContext.hpp

24.9 AIRRAC::FeaturesNotFoundException Class Reference

`#include <airrac/AIRRAC_Types.hpp>`Inheritance diagram for AIRRAC::FeaturesNotFoundException::



Public Member Functions

- FeaturesNotFoundException (const std::string &iWhat)

24.9.1 Detailed Description

Definition at line 47 of file AIRRAC_Types.hpp.

24.9.2 Constructor & Destructor Documentation

24.9.2.1 AIRRAC::FeaturesNotFoundException::FeaturesNotFoundException (const std::string & *iWhat*) `[inline]`

Constructor.

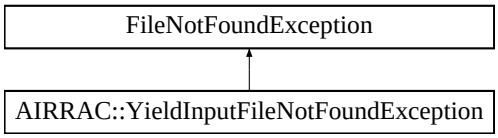
Definition at line 50 of file AIRRAC_Types.hpp.

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

- airrac/AIRRAC_Types.hpp

24.10 FileNotFoundException Class Reference

Inheritance diagram for FileNotFoundException::

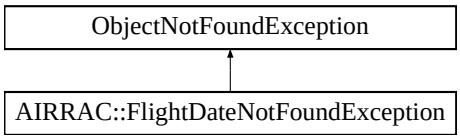


The documentation for this class was generated from the following file:

- airrac/AIRRAC_Types.hpp

24.11 AIRRAC::FlightDateNotFoundException Class Reference

`#include <airrac/AIRRAC_Types.hpp>`Inheritance diagram for AIRRAC::FlightDateNotFoundException::



Public Member Functions

- FlightDateNotFoundException (const std::string &iWhat)

24.11.1 Detailed Description

Definition at line 33 of file AIRRAC_Types.hpp.

24.11.2 Constructor & Destructor Documentation

24.11.2.1 AIRRAC::FlightDateNotFoundException::FlightDateNotFoundException (const std::string & *iWhat*) `[inline]`

Constructor.

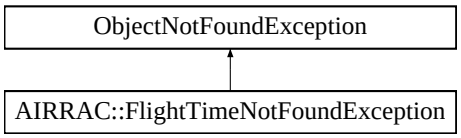
Definition at line 36 of file AIRRAC_Types.hpp.

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

- airrac/AIRRAC_Types.hpp

24.12 AIRRAC::FlightTimeNotFoundException Class Reference

`#include <airrac/AIRRAC_Types.hpp>`Inheritance diagram for AIRRAC::FlightTimeNotFoundException::



Public Member Functions

- FlightTimeNotFoundException (const std::string &iWhat)

24.12.1 Detailed Description

Definition at line 40 of file AIRRAC_Types.hpp.

24.12.2 Constructor & Destructor Documentation

24.12.2.1 AIRRAC::FlightTimeNotFoundException::FlightTimeNotFoundException (const std::string & *iWhat*) `[inline]`

Constructor.

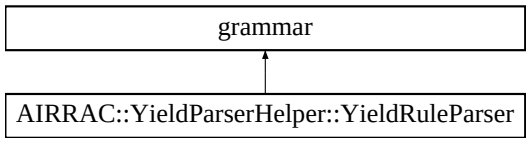
Definition at line 43 of file AIRRAC_Types.hpp.

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

- airrac/AIRRAC_Types.hpp

24.13 grammar Class Reference

Inheritance diagram for grammar::

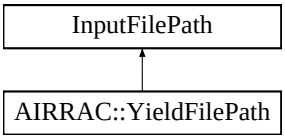


The documentation for this class was generated from the following file:

- airrac/command/YieldParserHelper.cpp

24.14 InputFilePath Class Reference

Inheritance diagram for InputFilePath::

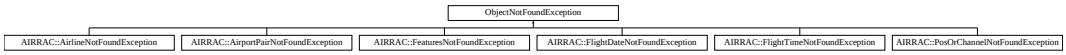


The documentation for this class was generated from the following file:

- airrac/AIRRAC_Types.hpp

24.15 ObjectNotFoundException Class Reference

Inheritance diagram for ObjectNotFoundException::

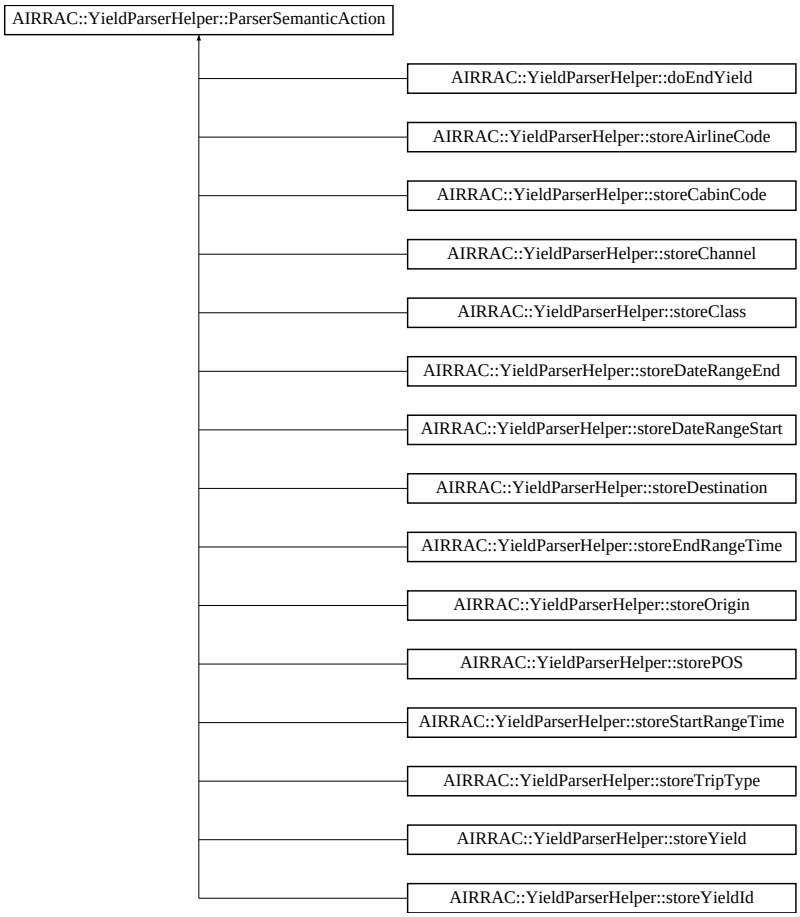


The documentation for this class was generated from the following file:

- airrac/AIRRAC_Types.hpp

24.16 AIRRAC::YieldParserHelper::ParserSemanticAction Struct Reference

`#include <airrac/command/YieldParserHelper.hpp>`Inheritance diagram for AIRRAC::YieldParserHelper::ParserSemanticAction::



Public Member Functions

- ParserSemanticAction (YieldRuleStruct &)

Public Attributes

- YieldRuleStruct & _yieldRule

24.16.1 Detailed Description

Generic Semantic Action (Actor / Functor) for the Yield Parser.

Definition at line 30 of file YieldParserHelper.hpp.

24.16.2 Constructor & Destructor Documentation

24.16.2.1 AIRRAC::YieldParserHelper::ParserSemanticAction::ParserSemanticAction (YieldRuleStruct & *ioYieldRule*)

Actor Constructor.

Definition at line 28 of file YieldParserHelper.cpp.

24.16.3 Member Data Documentation

24.16.3.1 YieldRuleStruct& AIRRAC::YieldParserHelper::ParserSemanticAction::_yieldRule

Actor Context.

Definition at line 34 of file YieldParserHelper.hpp.

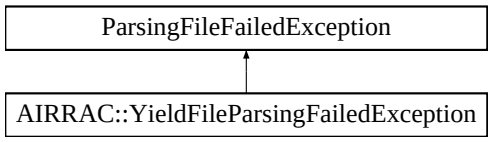
Referenced by AIRRAC::YieldParserHelper::doEndYield::operator()(), AIRRAC::YieldParserHelper::storeClass::operator()(), AIRRAC::YieldParserHelper::storeAirlineCode::operator()(), AIRRAC::YieldParserHelper::storeYield::operator()(), AIRRAC::YieldParserHelper::storeChannel::operator()(), AIRRAC::YieldParserHelper::storeCabinCode::operator()(), AIRRAC::YieldParserHelper::storePOS::operator()(), AIRRAC::YieldParserHelper::storeEndRangeTime::operator()(), AIRRAC::YieldParserHelper::storeStartRangeTime::operator AIRRAC::YieldParserHelper::storeDateRangeEnd::operator()(), AIRRAC::YieldParserHelper::storeDateRangeStart::operator() AIRRAC::YieldParserHelper::storeTripType::operator()(), AIRRAC::YieldParserHelper::storeDestination::operator()(), AIRRAC::YieldParserHelper::storeOrigin::operator()(), and AIRRAC::YieldParserHelper::storeYieldId::operator()().

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

- airrac/command/YieldParserHelper.hpp
- airrac/command/YieldParserHelper.cpp

24.17 ParsingFileFailedException Class Reference

Inheritance diagram for ParsingFileFailedException::

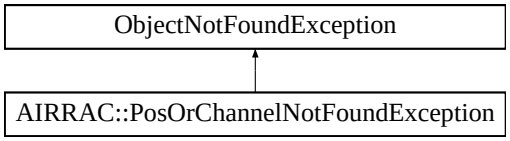


The documentation for this class was generated from the following file:

- airrac/AIRRAC_Types.hpp

24.18 AIRRAC::PosOrChannelNotFoundException Class Reference

`#include <airrac/AIRRAC_Types.hpp>`Inheritance diagram for AIRRAC::PosOrChannelNotFoundException::



Public Member Functions

- PosOrChannelNotFoundException (const std::string &iWhat)

24.18.1 Detailed Description

Definition at line 26 of file AIRRAC_Types.hpp.

24.18.2 Constructor & Destructor Documentation

24.18.2.1 AIRRAC::PosOrChannelNotFoundException::PosOrChannelNotFoundException (const std::string & *iWhat*) [inline]

Constructor.

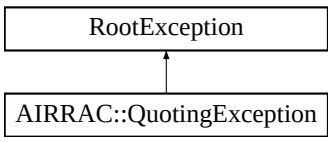
Definition at line 29 of file AIRRAC_Types.hpp.

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

- airrac/AIRRAC_Types.hpp

24.19 AIRRAC::QuotingException Class Reference

`#include <airrac/AIRRAC_Types.hpp>`Inheritance diagram for AIRRAC::QuotingException::



24.19.1 Detailed Description

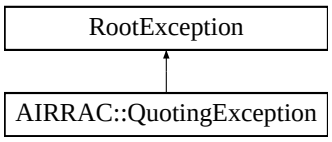
Definition at line 75 of file AIRRAC_Types.hpp.

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

- airrac/AIRRAC_Types.hpp

24.20 RootException Class Reference

Inheritance diagram for RootException::

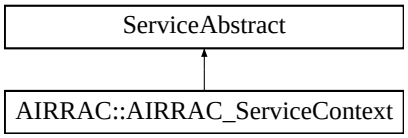


The documentation for this class was generated from the following file:

- airrac/AIRRAC_Types.hpp

24.21 ServiceAbstract Class Reference

Inheritance diagram for ServiceAbstract::

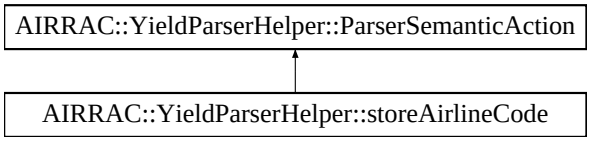


The documentation for this class was generated from the following file:

- airrac/service/AIRRAC_ServiceContext.hpp

24.22 AIRRAC::YieldParserHelper::storeAirlineCode Struct Reference

`#include <airrac/command/YieldParserHelper.hpp>`Inheritance diagram for AIRRAC::YieldParserHelper::storeAirlineCode::



Public Member Functions

- storeAirlineCode (YieldRuleStruct &)
- void operator() (std::vector< char >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- YieldRuleStruct & _yieldRule

24.22.1 Detailed Description

Store the parsed airline code.

Definition at line 158 of file YieldParserHelper.hpp.

24.22.2 Constructor & Destructor Documentation

24.22.2.1 AIRRAC::YieldParserHelper::storeAirlineCode::storeAirlineCode (YieldRuleStruct & *ioYieldRule*)

Actor Constructor.

Definition at line 270 of file YieldParserHelper.cpp.

24.22.3 Member Function Documentation

24.22.3.1 void AIRRAC::YieldParserHelper::storeAirlineCode::operator() (std::vector< char > *iChar*, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 275 of file YieldParserHelper.cpp.

References AIRRAC::YieldParserHelper::ParserSemanticAction::_yieldRule, AIRRAC::YieldRuleStruct::addAirlineCode(), and AIRRAC::YieldRuleStruct::setAirlineCode().

24.22.4 Member Data Documentation

24.22.4.1 YieldRuleStruct& AIRRAC::YieldParserHelper::ParserSemanticAction::_yieldRule [inherited]

Actor Context.

Definition at line 34 of file YieldParserHelper.hpp.

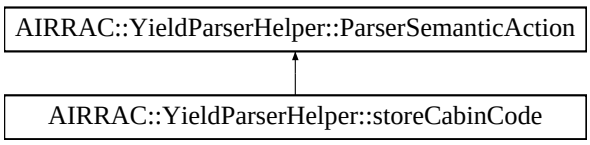
Referenced by AIRRAC::YieldParserHelper::doEndYield::operator()(), AIRRAC::YieldParserHelper::storeClass::operator()(), operator()(), AIRRAC::YieldParserHelper::storeYield::operator()(), AIRRAC::YieldParserHelper::storeChannel::operator()(), AIRRAC::YieldParserHelper::storeCabinCode::operator()(), AIRRAC::YieldParserHelper::storePOS::operator()(), AIRRAC::YieldParserHelper::storeEndRangeTime::operator()(), AIRRAC::YieldParserHelper::storeStartRangeTime::operator AIRRAC::YieldParserHelper::storeDateRangeEnd::operator()(), AIRRAC::YieldParserHelper::storeDateRangeStart::operator(AIRRAC::YieldParserHelper::storeTripType::operator()(), AIRRAC::YieldParserHelper::storeDestination::operator()(), AIRRAC::YieldParserHelper::storeOrigin::operator()(), and AIRRAC::YieldParserHelper::storeYieldId::operator()().

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

- airrac/command/YieldParserHelper.hpp
- airrac/command/YieldParserHelper.cpp

24.23 AIRRAC::YieldParserHelper::storeCabinCode Struct Reference

`#include <airrac/command/YieldParserHelper.hpp>`Inheritance diagram for AIRRAC::YieldParserHelper::storeCabinCode::



Public Member Functions

- storeCabinCode (YieldRuleStruct &)
- void operator() (char, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- YieldRuleStruct & _yieldRule

24.23.1 Detailed Description

Store the cabin code.

Definition at line 128 of file YieldParserHelper.hpp.

24.23.2 Constructor & Destructor Documentation

24.23.2.1 AIRRAC::YieldParserHelper::storeCabinCode::storeCabinCode (YieldRuleStruct & *ioYieldRule*)

Actor Constructor.

Definition at line 212 of file YieldParserHelper.cpp.

24.23.3 Member Function Documentation

24.23.3.1 void AIRRAC::YieldParserHelper::storeCabinCode::operator() (char *iChar*, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 217 of file YieldParserHelper.cpp.

References AIRRAC::YieldParserHelper::ParserSemanticAction::_yieldRule, and AIRRAC::YieldRuleStruct::setCabinCode().

24.23.4 Member Data Documentation

24.23.4.1 YieldRuleStruct& AIRRAC::YieldParserHelper::ParserSemanticAction::_yieldRule [inherited]

Actor Context.

Definition at line 34 of file YieldParserHelper.hpp.

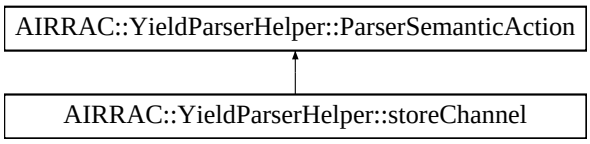
Referenced by AIRRAC::YieldParserHelper::doEndYield::operator()(), AIRRAC::YieldParserHelper::storeClass::operator()(), AIRRAC::YieldParserHelper::storeAirlineCode::operator()(), AIRRAC::YieldParserHelper::storeYield::operator()(), AIRRAC::YieldParserHelper::storeChannel::operator()(), operator()(), AIRRAC::YieldParserHelper::storePOS::operator()(), AIRRAC::YieldParserHelper::storeEndRangeTime::operator()(), AIRRAC::YieldParserHelper::storeStartRangeTime::operator()(), AIRRAC::YieldParserHelper::storeDateRangeEnd::operator()(), AIRRAC::YieldParserHelper::storeDateRangeStart::operator(), AIRRAC::YieldParserHelper::storeTripType::operator()(), AIRRAC::YieldParserHelper::storeDestination::operator()(), AIRRAC::YieldParserHelper::storeOrigin::operator()(), and AIRRAC::YieldParserHelper::storeYieldId::operator()().

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

- airrac/command/YieldParserHelper.hpp
- airrac/command/YieldParserHelper.cpp

24.24 AIRRAC::YieldParserHelper::storeChannel Struct Reference

`#include <airrac/command/YieldParserHelper.hpp>`Inheritance diagram for AIRRAC::YieldParserHelper::storeChannel::



Public Member Functions

- storeChannel (YieldRuleStruct &)
- void operator() (std::vector< char >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- YieldRuleStruct & _yieldRule

24.24.1 Detailed Description

Store the channel distribution.

Definition at line 138 of file YieldParserHelper.hpp.

24.24.2 Constructor & Destructor Documentation

24.24.2.1 AIRRAC::YieldParserHelper::storeChannel::storeChannel (YieldRuleStruct & *ioYieldRule*)

Actor Constructor.

Definition at line 233 of file YieldParserHelper.cpp.

24.24.3 Member Function Documentation

24.24.3.1 void AIRRAC::YieldParserHelper::storeChannel::operator() (std::vector< char > *iChar*, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 238 of file YieldParserHelper.cpp.

References AIRRAC::YieldParserHelper::ParserSemanticAction::_yieldRule, and AIRRAC::YieldRuleStruct::setChannel().

24.24.4 Member Data Documentation

24.24.4.1 YieldRuleStruct& AIRRAC::YieldParserHelper::ParserSemanticAction::_yieldRule [inherited]

Actor Context.

Definition at line 34 of file YieldParserHelper.hpp.

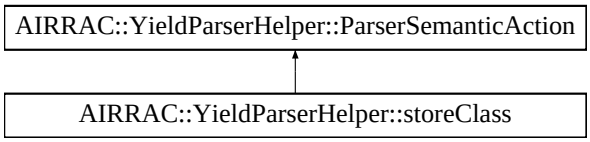
Referenced by AIRRAC::YieldParserHelper::doEndYield::operator()(), AIRRAC::YieldParserHelper::storeClass::operator()(), AIRRAC::YieldParserHelper::storeAirlineCode::operator()(), AIRRAC::YieldParserHelper::storeYield::operator()(), operator()(), AIRRAC::YieldParserHelper::storeCabinCode::operator()(), AIRRAC::YieldParserHelper::storePOS::operator()(), AIRRAC::YieldParserHelper::storeEndRangeTime::operator()(), AIRRAC::YieldParserHelper::storeStartRangeTime::operator AIRRAC::YieldParserHelper::storeDateRangeEnd::operator()(), AIRRAC::YieldParserHelper::storeDateRangeStart::operator(AIRRAC::YieldParserHelper::storeTripType::operator()(), AIRRAC::YieldParserHelper::storeDestination::operator()(), AIRRAC::YieldParserHelper::storeOrigin::operator()(), and AIRRAC::YieldParserHelper::storeYieldId::operator()().

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

- airrac/command/YieldParserHelper.hpp
- airrac/command/YieldParserHelper.cpp

24.25 AIRRAC::YieldParserHelper::storeClass Struct Reference

`#include <airrac/command/YieldParserHelper.hpp>`Inheritance diagram for AIRRAC::YieldParserHelper::storeClass::



Public Member Functions

- storeClass (YieldRuleStruct &)
- void operator() (std::vector< char >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- YieldRuleStruct & _yieldRule

24.25.1 Detailed Description

Store the parsed class.

Definition at line 168 of file YieldParserHelper.hpp.

24.25.2 Constructor & Destructor Documentation

24.25.2.1 AIRRAC::YieldParserHelper::storeClass::storeClass (YieldRuleStruct & *ioYieldRule*)

Actor Constructor.

Definition at line 290 of file YieldParserHelper.cpp.

24.25.3 Member Function Documentation

24.25.3.1 void AIRRAC::YieldParserHelper::storeClass::operator() (std::vector< char > *iChar*, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 295 of file YieldParserHelper.cpp.

References AIRRAC::YieldParserHelper::ParserSemanticAction::_yieldRule, and AIRRAC::YieldRuleStruct::addClassCode().

24.25.4 Member Data Documentation

24.25.4.1 YieldRuleStruct& AIRRAC::YieldParserHelper::ParserSemanticAction::_yieldRule [inherited]

Actor Context.

Definition at line 34 of file YieldParserHelper.hpp.

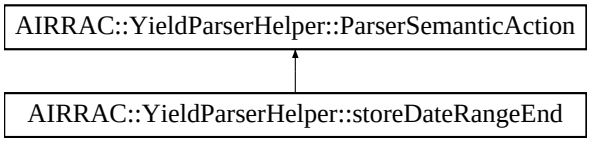
Referenced by AIRRAC::YieldParserHelper::doEndYield::operator()(), operator()(), AIRRAC::YieldParserHelper::storeAirlineCode::operator()(), AIRRAC::YieldParserHelper::storeYield::operator()(), AIRRAC::YieldParserHelper::storeChannel::operator()(), AIRRAC::YieldParserHelper::storeCabinCode::operator()(), AIRRAC::YieldParserHelper::storePOS::operator()(), AIRRAC::YieldParserHelper::storeEndRangeTime::operator()(), AIRRAC::YieldParserHelper::storeStartRangeTime::operator(), AIRRAC::YieldParserHelper::storeDateRangeEnd::operator()(), AIRRAC::YieldParserHelper::storeDateRangeStart::operator(), AIRRAC::YieldParserHelper::storeTripType::operator()(), AIRRAC::YieldParserHelper::storeDestination::operator()(), AIRRAC::YieldParserHelper::storeOrigin::operator()(), and AIRRAC::YieldParserHelper::storeYieldId::operator()().

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

- airrac/command/YieldParserHelper.hpp
- airrac/command/YieldParserHelper.cpp

24.26 AIRRAC::YieldParserHelper::storeDateRangeEnd Struct Reference

`#include <airrac/command/YieldParserHelper.hpp>`Inheritance diagram for AIRRAC::YieldParserHelper::storeDateRangeEnd::



Public Member Functions

- storeDateRangeEnd (YieldRuleStruct &)
- void operator() (boost::spirit::qi::unused_type, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- YieldRuleStruct & _yieldRule

24.26.1 Detailed Description

Store the parsed end of the date range.

Definition at line 88 of file YieldParserHelper.hpp.

24.26.2 Constructor & Destructor Documentation

24.26.2.1 AIRRAC::YieldParserHelper::storeDateRangeEnd::storeDateRangeEnd (YieldRuleStruct & *ioYieldRule*)

Actor Constructor.

Definition at line 129 of file YieldParserHelper.cpp.

24.26.3 Member Function Documentation

24.26.3.1 void AIRRAC::YieldParserHelper::storeDateRangeEnd::operator() (boost::spirit::qi::unused_type, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 134 of file YieldParserHelper.cpp.

References AIRRAC::YieldParserHelper::ParserSemanticAction::_yieldRule, AIRRAC::YieldRuleStruct::calculateDate(), and AIRRAC::YieldRuleStruct::setDateRangeEnd().

24.26.4 Member Data Documentation

24.26.4.1 YieldRuleStruct& AIRRAC::YieldParserHelper::ParserSemanticAction::_yieldRule [inherited]

Actor Context.

Definition at line 34 of file YieldParserHelper.hpp.

Referenced by AIRRAC::YieldParserHelper::doEndYield::operator()(), AIRRAC::YieldParserHelper::storeClass::operator()(), AIRRAC::YieldParserHelper::storeAirlineCode::operator()(), AIRRAC::YieldParserHelper::storeYield::operator()(), AIRRAC::YieldParserHelper::storeChannel::operator()(), AIRRAC::YieldParserHelper::storeCabinCode::operator()(), AIRRAC::YieldParserHelper::storePOS::operator()(), AIRRAC::YieldParserHelper::storeEndRangeTime::operator()(), AIRRAC::YieldParserHelper::storeStartRangeTime::operator

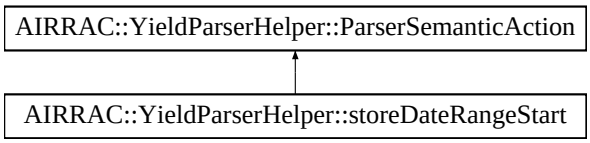
operator()(), AIRRAC::YieldParserHelper::storeDateRangeStart::operator()(), AIRRAC::YieldParserHelper::storeTripType::operator()(), AIRRAC::YieldParserHelper::storeDestination::operator()(), AIRRAC::YieldParserHelper::storeOrigin::operator()(), and AIRRAC::YieldParserHelper::storeYieldId::operator()().

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

- airrac/command/YieldParserHelper.hpp
- airrac/command/YieldParserHelper.cpp

24.27 AIRRAC::YieldParserHelper::storeDateRangeStart Struct Reference

`#include <airrac/command/YieldParserHelper.hpp>`Inheritance diagram for AIRRAC::YieldParserHelper::storeDateRangeStart::



Public Member Functions

- storeDateRangeStart (YieldRuleStruct &)
- void operator() (boost::spirit::qi::unused_type, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- YieldRuleStruct & _yieldRule

24.27.1 Detailed Description

Store the parsed start of the date range.

Definition at line 78 of file YieldParserHelper.hpp.

24.27.2 Constructor & Destructor Documentation

24.27.2.1 AIRRAC::YieldParserHelper::storeDateRangeStart::storeDateRangeStart (YieldRuleStruct & *ioYieldRule*)

Actor Constructor.

Definition at line 113 of file YieldParserHelper.cpp.

24.27.3 Member Function Documentation

24.27.3.1 void AIRRAC::YieldParserHelper::storeDateRangeStart::operator() (boost::spirit::qi::unused_type, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 118 of file YieldParserHelper.cpp.

References AIRRAC::YieldParserHelper::ParserSemanticAction::_yieldRule, AIRRAC::YieldRuleStruct::calculateDate(), and AIRRAC::YieldRuleStruct::setDateRangeStart().

24.27.4 Member Data Documentation

24.27.4.1 YieldRuleStruct& AIRRAC::YieldParserHelper::ParserSemanticAction::_yieldRule [inherited]

Actor Context.

Definition at line 34 of file YieldParserHelper.hpp.

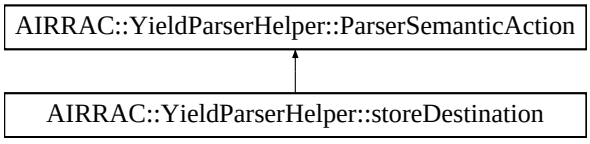
Referenced by AIRRAC::YieldParserHelper::doEndYield::operator()(), AIRRAC::YieldParserHelper::storeClass::operator()(), AIRRAC::YieldParserHelper::storeAirlineCode::operator()(), AIRRAC::YieldParserHelper::storeYield::operator()(), AIRRAC::YieldParserHelper::storeChannel::operator()(), AIRRAC::YieldParserHelper::storeCabinCode::operator()(), AIRRAC::YieldParserHelper::storePOS::operator()(), AIRRAC::YieldParserHelper::storeEndRangeTime::operator()(), AIRRAC::YieldParserHelper::storeStartRangeTime::operator AIRRAC::YieldParserHelper::storeDateRangeEnd::operator()(), operator()(), AIRRAC::YieldParserHelper::storeTripType::operator()(), AIRRAC::YieldParserHelper::storeDestination::operator()(), AIRRAC::YieldParserHelper::storeOrigin::operator()(), and AIRRAC::YieldParserHelper::storeYieldId::operator()().

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

- airrac/command/YieldParserHelper.hpp
- airrac/command/YieldParserHelper.cpp

24.28 AIRRAC::YieldParserHelper::storeDestination Struct Reference

`#include <airrac/command/YieldParserHelper.hpp>`Inheritance diagram for AIRRAC::YieldParserHelper::storeDestination::



Public Member Functions

- storeDestination (YieldRuleStruct &)
- void operator() (std::vector< char >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- YieldRuleStruct & _yieldRule

24.28.1 Detailed Description

Store the parsed destination.

Definition at line 58 of file YieldParserHelper.hpp.

24.28.2 Constructor & Destructor Documentation

24.28.2.1 AIRRAC::YieldParserHelper::storeDestination::storeDestination (YieldRuleStruct & *ioYieldRule*)

Actor Constructor.

Definition at line 75 of file YieldParserHelper.cpp.

24.28.3 Member Function Documentation

24.28.3.1 void AIRRAC::YieldParserHelper::storeDestination::operator() (std::vector< char > *iChar*, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 80 of file YieldParserHelper.cpp.

References AIRRAC::YieldParserHelper::ParserSemanticAction::_yieldRule, and AIRRAC::YieldRuleStruct::setDestination().

24.28.4 Member Data Documentation

24.28.4.1 YieldRuleStruct& AIRRAC::YieldParserHelper::ParserSemanticAction::_yieldRule [inherited]

Actor Context.

Definition at line 34 of file YieldParserHelper.hpp.

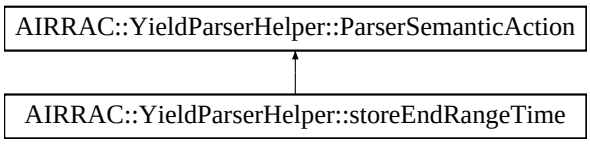
Referenced by AIRRAC::YieldParserHelper::doEndYield::operator()(), AIRRAC::YieldParserHelper::storeClass::operator()(), AIRRAC::YieldParserHelper::storeAirlineCode::operator()(), AIRRAC::YieldParserHelper::storeYield::operator()(), AIRRAC::YieldParserHelper::storeChannel::operator()(), AIRRAC::YieldParserHelper::storeCabinCode::operator()(), AIRRAC::YieldParserHelper::storePOS::operator()(), AIRRAC::YieldParserHelper::storeEndRangeTime::operator()(), AIRRAC::YieldParserHelper::storeStartRangeTime::operator AIRRAC::YieldParserHelper::storeDateRangeEnd::operator()(), AIRRAC::YieldParserHelper::storeDateRangeStart::operator(AIRRAC::YieldParserHelper::storeTripType::operator()(), operator()(), AIRRAC::YieldParserHelper::storeOrigin::operator()(), and AIRRAC::YieldParserHelper::storeYieldId::operator()().

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

- airrac/command/YieldParserHelper.hpp
- airrac/command/YieldParserHelper.cpp

24.29 AIRRAC::YieldParserHelper::storeEndRangeTime Struct Reference

`#include <airrac/command/YieldParserHelper.hpp>`Inheritance diagram for AIRRAC::YieldParserHelper::storeEndRangeTime::



Public Member Functions

- storeEndRangeTime (YieldRuleStruct &)
- void operator() (boost::spirit::qi::unused_type, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- YieldRuleStruct & _yieldRule

24.29.1 Detailed Description

Store the parsed end start range time.

Definition at line 108 of file YieldParserHelper.hpp.

24.29.2 Constructor & Destructor Documentation

24.29.2.1 AIRRAC::YieldParserHelper::storeEndRangeTime::storeEndRangeTime (YieldRuleStruct & *ioYieldRule*)

Actor Constructor.

Definition at line 168 of file YieldParserHelper.cpp.

24.29.3 Member Function Documentation

24.29.3.1 void AIRRAC::YieldParserHelper::storeEndRangeTime::operator() (boost::spirit::qi::unused_type, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 173 of file YieldParserHelper.cpp.

References AIRRAC::YieldRuleStruct::_itSeconds, AIRRAC::YieldParserHelper::ParserSemanticAction::_yieldRule, AIRRAC::YieldRuleStruct::calculateTime(), and AIRRAC::YieldRuleStruct::setTimeRangeEnd().

24.29.4 Member Data Documentation

24.29.4.1 YieldRuleStruct& AIRRAC::YieldParserHelper::ParserSemanticAction::_yieldRule [inherited]

Actor Context.

Definition at line 34 of file YieldParserHelper.hpp.

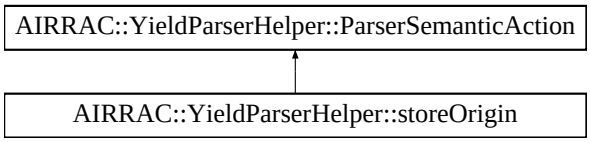
Referenced by AIRRAC::YieldParserHelper::doEndYield::operator()(), AIRRAC::YieldParserHelper::storeClass::operator()(), AIRRAC::YieldParserHelper::storeAirlineCode::operator()(), AIRRAC::YieldParserHelper::storeYield::operator()(), AIRRAC::YieldParserHelper::storeChannel::operator()(), AIRRAC::YieldParserHelper::storeCabinCode::operator()(), AIRRAC::YieldParserHelper::storePOS::operator()(), operator()(), AIRRAC::YieldParserHelper::storeStartRangeTime::operator()(), AIRRAC::YieldParserHelper::storeDateRangeEnd::operator()(), AIRRAC::YieldParserHelper::storeDateRangeStart::operator()(), AIRRAC::YieldParserHelper::storeTripType::operator()(), AIRRAC::YieldParserHelper::storeDestination::operator()(), AIRRAC::YieldParserHelper::storeOrigin::operator()(), and AIRRAC::YieldParserHelper::storeYieldId::operator()().

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

- airrac/command/YieldParserHelper.hpp
- airrac/command/YieldParserHelper.cpp

24.30 AIRRAC::YieldParserHelper::storeOrigin Struct Reference

`#include <airrac/command/YieldParserHelper.hpp>`Inheritance diagram for AIRRAC::YieldParserHelper::storeOrigin::



Public Member Functions

- storeOrigin (YieldRuleStruct &)
- void operator() (std::vector< char >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- YieldRuleStruct & _yieldRule

24.30.1 Detailed Description

Store the parsed origin.

Definition at line 48 of file YieldParserHelper.hpp.

24.30.2 Constructor & Destructor Documentation

24.30.2.1 AIRRAC::YieldParserHelper::storeOrigin::storeOrigin (YieldRuleStruct & *ioYieldRule*)

Actor Constructor.

Definition at line 59 of file YieldParserHelper.cpp.

24.30.3 Member Function Documentation

24.30.3.1 void AIRRAC::YieldParserHelper::storeOrigin::operator() (std::vector< char > *iChar*, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 64 of file YieldParserHelper.cpp.

References AIRRAC::YieldParserHelper::ParserSemanticAction::_yieldRule, and AIRRAC::YieldRuleStruct::setOrigin().

24.30.4 Member Data Documentation

24.30.4.1 YieldRuleStruct& AIRRAC::YieldParserHelper::ParserSemanticAction::_yieldRule [inherited]

Actor Context.

Definition at line 34 of file YieldParserHelper.hpp.

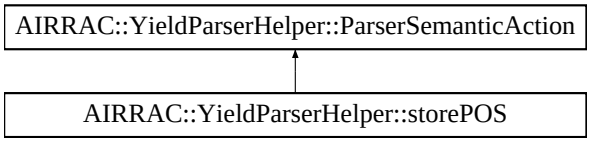
Referenced by AIRRAC::YieldParserHelper::doEndYield::operator()(), AIRRAC::YieldParserHelper::storeClass::operator()(), AIRRAC::YieldParserHelper::storeAirlineCode::operator()(), AIRRAC::YieldParserHelper::storeYield::operator()(), AIRRAC::YieldParserHelper::storeChannel::operator()(), AIRRAC::YieldParserHelper::storeCabinCode::operator()(), AIRRAC::YieldParserHelper::storePOS::operator()(), AIRRAC::YieldParserHelper::storeEndRangeTime::operator()(), AIRRAC::YieldParserHelper::storeStartRangeTime::operator AIRRAC::YieldParserHelper::storeDateRangeEnd::operator()(), AIRRAC::YieldParserHelper::storeDateRangeStart::operator() AIRRAC::YieldParserHelper::storeTripType::operator()(), AIRRAC::YieldParserHelper::storeDestination::operator()(), operator()(), and AIRRAC::YieldParserHelper::storeYieldId::operator()().

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

- airrac/command/YieldParserHelper.hpp
- airrac/command/YieldParserHelper.cpp

24.31 AIRRAC::YieldParserHelper::storePOS Struct Reference

`#include <airrac/command/YieldParserHelper.hpp>`Inheritance diagram for AIRRAC::YieldParserHelper::storePOS::



Public Member Functions

- storePOS (YieldRuleStruct &)
- void operator() (std::vector< char >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- YieldRuleStruct & _yieldRule

24.31.1 Detailed Description

Store the parsed customer point_of_sale.

Definition at line 118 of file YieldParserHelper.hpp.

24.31.2 Constructor & Destructor Documentation

24.31.2.1 AIRRAC::YieldParserHelper::storePOS::storePOS (YieldRuleStruct & *ioYieldRule*)

Actor Constructor.

Definition at line 186 of file YieldParserHelper.cpp.

24.31.3 Member Function Documentation

24.31.3.1 void AIRRAC::YieldParserHelper::storePOS::operator() (std::vector< char > *iChar*, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 191 of file YieldParserHelper.cpp.

References AIRRAC::YieldParserHelper::ParserSemanticAction::_yieldRule, AIRRAC::YieldRuleStruct::getDestination(), AIRRAC::YieldRuleStruct::getOrigin(), and AIRRAC::YieldRuleStruct::setPOS().

24.31.4 Member Data Documentation

24.31.4.1 YieldRuleStruct& AIRRAC::YieldParserHelper::ParserSemanticAction::_yieldRule [inherited]

Actor Context.

Definition at line 34 of file YieldParserHelper.hpp.

Referenced by AIRRAC::YieldParserHelper::doEndYield::operator()(), AIRRAC::YieldParserHelper::storeClass::operator()(), AIRRAC::YieldParserHelper::storeAirlineCode::operator()(), AIRRAC::YieldParserHelper::storeYield::operator()(), AIRRAC::YieldParserHelper::storeChannel::operator()(), AIRRAC::YieldParserHelper::storeCabinCode::operator()(), operator()(), AIRRAC::YieldParserHelper::storeEndRangeTime::operator()(), AIRRAC::YieldParserHelper::storeStartRangeTime::operator()(), AIRRAC::YieldParserHelper::storeDateRangeEnd::operator()(), AIRRAC::YieldParserHelper::storeDateRangeStart::operator()

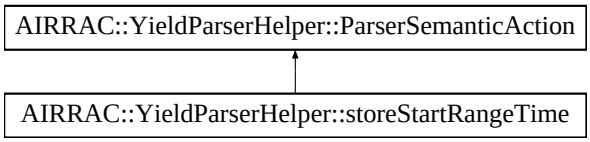
AIRRAC::YieldParserHelper::storeTripType::operator()(), AIRRAC::YieldParserHelper::storeDestination::operator()(), AIRRAC::YieldParserHelper::storeOrigin::operator()(), and AIRRAC::YieldParserHelper::storeYieldId::operator()().

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

- airrac/command/YieldParserHelper.hpp
- airrac/command/YieldParserHelper.cpp

24.32 AIRRAC::YieldParserHelper::storeStartRangeTime Struct Reference

`#include <airrac/command/YieldParserHelper.hpp>`Inheritance diagram for AIRRAC::YieldParserHelper::storeStartRangeTime::



Public Member Functions

- storeStartRangeTime (YieldRuleStruct &)
- void operator() (boost::spirit::qi::unused_type, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- YieldRuleStruct & _yieldRule

24.32.1 Detailed Description

Store the parsed start range time.

Definition at line 98 of file YieldParserHelper.hpp.

24.32.2 Constructor & Destructor Documentation

24.32.2.1 AIRRAC::YieldParserHelper::storeStartRangeTime::storeStartRangeTime (YieldRuleStruct & *ioYieldRule*)

Actor Constructor.

Definition at line 150 of file YieldParserHelper.cpp.

24.32.3 Member Function Documentation

24.32.3.1 void AIRRAC::YieldParserHelper::storeStartRangeTime::operator() (boost::spirit::qi::unused_type, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 155 of file YieldParserHelper.cpp.

References AIRRAC::YieldRuleStruct::_itSeconds, AIRRAC::YieldParserHelper::ParserSemanticAction::_yieldRule, AIRRAC::YieldRuleStruct::calculateTime(), and AIRRAC::YieldRuleStruct::setTimeRangeStart().

24.32.4 Member Data Documentation

24.32.4.1 YieldRuleStruct& AIRRAC::YieldParserHelper::ParserSemanticAction::_yieldRule [inherited]

Actor Context.

Definition at line 34 of file YieldParserHelper.hpp.

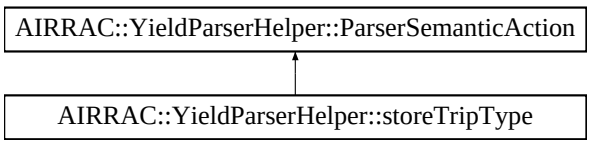
Referenced by AIRRAC::YieldParserHelper::doEndYield::operator()(), AIRRAC::YieldParserHelper::storeClass::operator()(), AIRRAC::YieldParserHelper::storeAirlineCode::operator()(), AIRRAC::YieldParserHelper::storeYield::operator()(), AIRRAC::YieldParserHelper::storeChannel::operator()(), AIRRAC::YieldParserHelper::storeCabinCode::operator()(), AIRRAC::YieldParserHelper::storePOS::operator()(), AIRRAC::YieldParserHelper::storeEndRangeTime::operator()(), operator()(), AIRRAC::YieldParserHelper::storeDateRangeEnd::operator()(), AIRRAC::YieldParserHelper::storeDateRangeStart::operator()(), AIRRAC::YieldParserHelper::storeTripType::operator()(), AIRRAC::YieldParserHelper::storeDestination::operator()(), AIRRAC::YieldParserHelper::storeOrigin::operator()(), and AIRRAC::YieldParserHelper::storeYieldId::operator()().

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

- airrac/command/YieldParserHelper.hpp
- airrac/command/YieldParserHelper.cpp

24.33 AIRRAC::YieldParserHelper::storeTripType Struct Reference

`#include <airrac/command/YieldParserHelper.hpp>`Inheritance diagram for AIRRAC::YieldParserHelper::storeTripType::



Public Member Functions

- storeTripType (YieldRuleStruct &)
- void operator() (std::vector< char >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- YieldRuleStruct & _yieldRule

24.33.1 Detailed Description

Store the parsed customer trip type.

Definition at line 68 of file YieldParserHelper.hpp.

24.33.2 Constructor & Destructor Documentation

24.33.2.1 AIRRAC::YieldParserHelper::storeTripType::storeTripType (YieldRuleStruct & *ioYieldRule*)

Actor Constructor.

Definition at line 91 of file YieldParserHelper.cpp.

24.33.3 Member Function Documentation

24.33.3.1 void AIRRAC::YieldParserHelper::storeTripType::operator() (std::vector< char > *iChar*, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 96 of file YieldParserHelper.cpp.

References AIRRAC::YieldParserHelper::ParserSemanticAction::_yieldRule, and AIRRAC::YieldRuleStruct::setTripType().

24.33.4 Member Data Documentation

24.33.4.1 YieldRuleStruct& AIRRAC::YieldParserHelper::ParserSemanticAction::_yieldRule [inherited]

Actor Context.

Definition at line 34 of file YieldParserHelper.hpp.

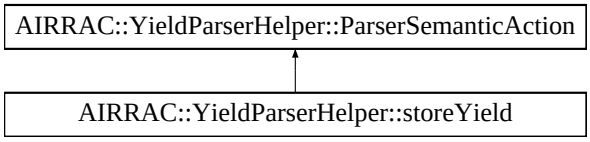
Referenced by AIRRAC::YieldParserHelper::doEndYield::operator()(), AIRRAC::YieldParserHelper::storeClass::operator()(), AIRRAC::YieldParserHelper::storeAirlineCode::operator()(), AIRRAC::YieldParserHelper::storeYield::operator()(), AIRRAC::YieldParserHelper::storeChannel::operator()(), AIRRAC::YieldParserHelper::storeCabinCode::operator()(), AIRRAC::YieldParserHelper::storePOS::operator()(), AIRRAC::YieldParserHelper::storeEndRangeTime::operator()(), AIRRAC::YieldParserHelper::storeStartRangeTime::operator AIRRAC::YieldParserHelper::storeDateRangeEnd::operator()(), AIRRAC::YieldParserHelper::storeDateRangeStart::operator(operator()(), AIRRAC::YieldParserHelper::storeDestination::operator()(), AIRRAC::YieldParserHelper::storeOrigin::operator()(), and AIRRAC::YieldParserHelper::storeYieldId::operator()().

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

- airrac/command/YieldParserHelper.hpp
- airrac/command/YieldParserHelper.cpp

24.34 AIRRAC::YieldParserHelper::storeYield Struct Reference

`#include <airrac/command/YieldParserHelper.hpp>`Inheritance diagram for AIRRAC::YieldParserHelper::storeYield::



Public Member Functions

- storeYield (YieldRuleStruct &)
- void operator() (double, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- YieldRuleStruct & _yieldRule

24.34.1 Detailed Description

Store the parsed yield value.

Definition at line 148 of file YieldParserHelper.hpp.

24.34.2 Constructor & Destructor Documentation

24.34.2.1 AIRRAC::YieldParserHelper::storeYield::storeYield (YieldRuleStruct & *ioYieldRule*)

Actor Constructor.

Definition at line 254 of file YieldParserHelper.cpp.

24.34.3 Member Function Documentation

24.34.3.1 void AIRRAC::YieldParserHelper::storeYield::operator() (double *iYield*, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 259 of file YieldParserHelper.cpp.

References AIRRAC::YieldParserHelper::ParserSemanticAction::_yieldRule, and AIRRAC::YieldRuleStruct::setYield().

24.34.4 Member Data Documentation

24.34.4.1 YieldRuleStruct& AIRRAC::YieldParserHelper::ParserSemanticAction::_yieldRule [inherited]

Actor Context.

Definition at line 34 of file YieldParserHelper.hpp.

Referenced by AIRRAC::YieldParserHelper::doEndYield::operator()(), AIRRAC::YieldParserHelper::storeClass::operator()(), AIRRAC::YieldParserHelper::storeAirlineCode::operator()(),

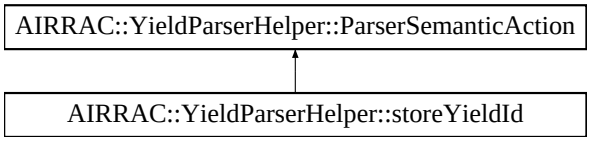
operator()(), AIRRAC::YieldParserHelper::storeChannel::operator()(), AIRRAC::YieldParserHelper::storeCabinCode::operator()(), AIRRAC::YieldParserHelper::storePOS::operator()(), AIRRAC::YieldParserHelper::storeEndRangeTime::operator()(), AIRRAC::YieldParserHelper::storeStartRangeTime::operator AIRRAC::YieldParserHelper::storeDateRangeEnd::operator()(), AIRRAC::YieldParserHelper::storeDateRangeStart::operator() AIRRAC::YieldParserHelper::storeTripType::operator()(), AIRRAC::YieldParserHelper::storeDestination::operator()(), AIRRAC::YieldParserHelper::storeOrigin::operator()(), and AIRRAC::YieldParserHelper::storeYieldId::operator()().

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

- airrac/command/YieldParserHelper.hpp
- airrac/command/YieldParserHelper.cpp

24.35 AIRRAC::YieldParserHelper::storeYieldId Struct Reference

`#include <airrac/command/YieldParserHelper.hpp>`Inheritance diagram for AIRRAC::YieldParserHelper::storeYieldId::



Public Member Functions

- storeYieldId (YieldRuleStruct &)
- void operator() (unsigned int, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- YieldRuleStruct & _yieldRule

24.35.1 Detailed Description

Store the parsed yield Id.

Definition at line 38 of file YieldParserHelper.hpp.

24.35.2 Constructor & Destructor Documentation

24.35.2.1 AIRRAC::YieldParserHelper::storeYieldId::storeYieldId (YieldRuleStruct & *ioYieldRule*)

Actor Constructor.

Definition at line 34 of file YieldParserHelper.cpp.

24.35.3 Member Function Documentation

24.35.3.1 void AIRRAC::YieldParserHelper::storeYieldId::operator() (unsigned int *iYieldId*, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 39 of file YieldParserHelper.cpp.

References AIRRAC::YieldRuleStruct::_itSeconds, AIRRAC::YieldParserHelper::ParserSemanticAction::_yieldRule, AIRRAC::YieldRuleStruct::clearAirlineCodeList(), AIRRAC::YieldRuleStruct::clearClassCodeList(), AIRRAC::YieldRuleStruct::setAirlineCode(), AIRRAC::YieldRuleStruct::setClassCode(), and AIRRAC::YieldRuleStruct::setYieldID().

24.35.4 Member Data Documentation

24.35.4.1 YieldRuleStruct& AIRRAC::YieldParserHelper::ParserSemanticAction::_yieldRule [inherited]

Actor Context.

Definition at line 34 of file YieldParserHelper.hpp.

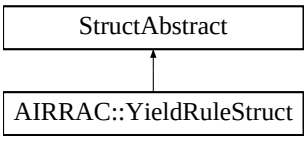
Referenced by AIRRAC::YieldParserHelper::doEndYield::operator()(), AIRRAC::YieldParserHelper::storeClass::operator()(), AIRRAC::YieldParserHelper::storeAirlineCode::operator()(), AIRRAC::YieldParserHelper::storeYield::operator()(), AIRRAC::YieldParserHelper::storeChannel::operator()(), AIRRAC::YieldParserHelper::storeCabinCode::operator()(), AIRRAC::YieldParserHelper::storePOS::operator()(), AIRRAC::YieldParserHelper::storeEndRangeTime::operator()(), AIRRAC::YieldParserHelper::storeStartRangeTime::operator(), AIRRAC::YieldParserHelper::storeDateRangeEnd::operator()(), AIRRAC::YieldParserHelper::storeDateRangeStart::operator(), AIRRAC::YieldParserHelper::storeTripType::operator()(), AIRRAC::YieldParserHelper::storeDestination::operator()(), AIRRAC::YieldParserHelper::storeOrigin::operator()(), and operator()().

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

- airrac/command/YieldParserHelper.hpp
- airrac/command/YieldParserHelper.cpp

24.36 StructAbstract Class Reference

Inheritance diagram for StructAbstract::

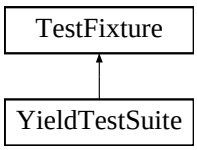


The documentation for this class was generated from the following file:

- airrac/bom/YieldRuleStruct.hpp

24.37 TestFixture Class Reference

Inheritance diagram for TestFixture::

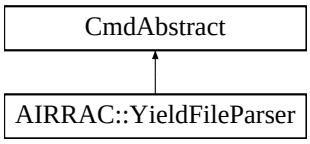


The documentation for this class was generated from the following file:

- test/airrac/YieldTestSuite.hpp

24.38 AIRRAC::YieldFileParser Class Reference

`#include <airrac/command/YieldParserHelper.hpp>`Inheritance diagram for AIRRAC::YieldFileParser::



Public Member Functions

- YieldFileParser (stdair::BomRoot &, const stdair::Filename_T &iYieldInputFilename)
- void generateYieldStore ()

24.38.1 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 202 of file YieldParserHelper.hpp.

24.38.2 Constructor & Destructor Documentation

24.38.2.1 AIRRAC::YieldFileParser::YieldFileParser (stdair::BomRoot &, const stdair::Filename_T & *iYieldInputFilename*)

Constructor.

24.38.3 Member Function Documentation

24.38.3.1 void AIRRAC::YieldFileParser::generateYieldStore ()

Parse the yield store input file.

Definition at line 529 of file YieldParserHelper.cpp.

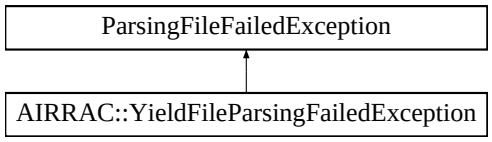
Referenced by AIRRAC::YieldParser::generateYieldStore().

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

- airrac/command/YieldParserHelper.hpp
- airrac/command/YieldParserHelper.cpp

24.39 AIRRAC::YieldFileParsingFailedException Class Reference

`#include <airrac/AIRRAC_Types.hpp>`Inheritance diagram for AIRRAC::YieldFileParsingFailedException::



Public Member Functions

- YieldFileParsingFailedException (const std::string &iWhat)

24.39.1 Detailed Description

Definition at line 68 of file AIRRAC_Types.hpp.

24.39.2 Constructor & Destructor Documentation

24.39.2.1 AIRRAC::YieldFileParsingFailedException::YieldFileParsingFailedException (const std::string & *iWhat*) `[inline]`

Constructor.

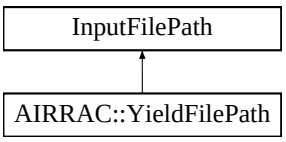
Definition at line 71 of file AIRRAC_Types.hpp.

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

- airrac/AIRRAC_Types.hpp

24.40 AIRRAC::YieldFilePath Class Reference

`#include <airrac/AIRRAC_Types.hpp>`Inheritance diagram for AIRRAC::YieldFilePath::



Public Member Functions

- YieldFilePath (const stdair::Filename_T &iFilename)

24.40.1 Detailed Description

Yield input file.

Definition at line 82 of file AIRRAC_Types.hpp.

24.40.2 Constructor & Destructor Documentation

24.40.2.1 AIRRAC::YieldFilePath::YieldFilePath (const stdair::Filename_T & *iFilename*) [inline, explicit]

Constructor.

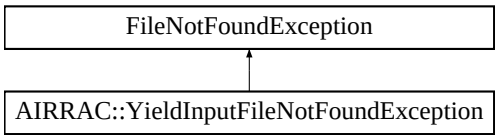
Definition at line 87 of file AIRRAC_Types.hpp.

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

- airrac/AIRRAC_Types.hpp

24.41 AIRRAC::YieldInputFileNotFoundException Class Reference

`#include <airrac/AIRRAC_Types.hpp>`Inheritance diagram for AIRRAC::YieldInputFileNotFoundException::



Public Member Functions

- YieldInputFileNotFoundException (const std::string &iWhat)

24.41.1 Detailed Description

Definition at line 61 of file AIRRAC_Types.hpp.

24.41.2 Constructor & Destructor Documentation

24.41.2.1 AIRRAC::YieldInputFileNotFoundException::YieldInputFileNotFoundException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 64 of file AIRRAC_Types.hpp.

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

- airrac/AIRRAC_Types.hpp

24.42 AIRRAC::YieldManager Class Reference

Command wrapping the travel request process.

`#include <airrac/command/YieldManager.hpp>`

Friends

- class AIRRAC_Service

24.42.1 Detailed Description

Command wrapping the travel request process.

Definition at line 23 of file YieldManager.hpp.

24.42.2 Friends And Related Function Documentation

24.42.2.1 friend class AIRRAC_Service [friend]

Only the AIRRAC_Service may access to the methods of that class.

Definition at line 27 of file YieldManager.hpp.

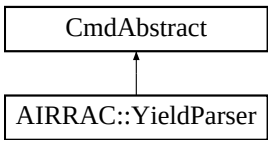
The documentation for this class was generated from the following files:

- airrac/command/YieldManager.hpp
- airrac/command/YieldManager.cpp

24.43 AIRRAC::YieldParser Class Reference

Class wrapping the parser entry point.

`#include <airrac/command/YieldParser.hpp>`Inheritance diagram for AIRRAC::YieldParser::



Static Public Member Functions

- static void generateYieldStore (const YieldFilePath &, stdair::BomRoot &)

24.43.1 Detailed Description

Class wrapping the parser entry point.

Definition at line 25 of file YieldParser.hpp.

24.43.2 Member Function Documentation

24.43.2.1 void AIRRAC::YieldParser::generateYieldStore (const YieldFilePath & *iYieldFilename*, stdair::BomRoot & *ioBomRoot*) [static]

Parse the CSV file describing an airline yield store, and generates the corresponding data model in memory. It can then be used, for instance in a simulator.

Parameters:

const YieldFilePath& The file-name of the CSV-formatted yield input file.

stdair::BomRoot& Root of the BOM tree.

Definition at line 16 of file YieldParser.cpp.

References AIRRAC::YieldFileParser::generateYieldStore().

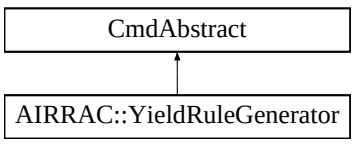
Referenced by AIRRAC::AIRRAC_Service::parseAndLoad().

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

- airrac/command/YieldParser.hpp
- airrac/command/YieldParser.cpp

24.44 AIRRAC::YieldRuleGenerator Class Reference

`#include <airrac/command/YieldRuleGenerator.hpp>`Inheritance diagram for AIRRAC::YieldRuleGenerator::



Friends

- class YieldFileParser
- struct YieldParserHelper::doEndYield
- class YieldParser

24.44.1 Detailed Description

Class handling the generation / instantiation of the Yield BOM.

Definition at line 32 of file YieldRuleGenerator.hpp.

24.44.2 Friends And Related Function Documentation

24.44.2.1 friend class YieldFileParser [friend]

Definition at line 36 of file YieldRuleGenerator.hpp.

24.44.2.2 friend struct YieldParserHelper::doEndYield [friend]

Definition at line 37 of file YieldRuleGenerator.hpp.

24.44.2.3 friend class YieldParser [friend]

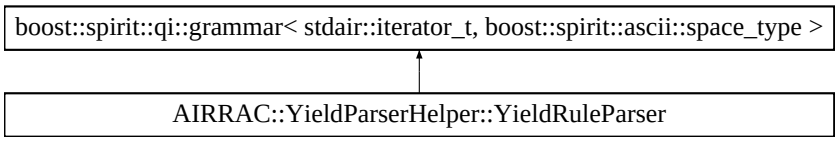
Definition at line 38 of file YieldRuleGenerator.hpp.

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

- airrac/command/YieldRuleGenerator.hpp
- airrac/command/YieldRuleGenerator.cpp

24.45 AIRRAC::YieldParserHelper::YieldRuleParser Struct Reference

Inheritance diagram for AIRRAC::YieldParserHelper::YieldRuleParser::



Public Member Functions

- YieldRuleParser (stdair::BomRoot &ioBomRoot, YieldRuleStruct &ioYieldRule)

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 > yield_rule
- boost::spirit::qi::rule< stdair::iterator_t, boost::spirit::ascii::space_type > yield_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 > tripType
- 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 > point_of_sale
- 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 > yield
- 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 > yield_rule_end
- stdair::BomRoot & _bomRoot
- YieldRuleStruct & _yieldRule

24.45.1 Detailed Description

Yields: yieldID; OriginCity; DestinationCity; DateRangeStart; DateRangeEnd; DepartureTimeRangeStart; DepartureTimeRangeEnd; Yield; AirlineCode; Class

1; LHR; JFK; 2008-06-01; 2009-12-31; 00:00; 23:59; 4200.0; BA; A;

YieldID (Integer) OriginCity (3-char airport code) DestinationCity (3-char airport code) DateRangeStart (yyyy-mm-dd) DateRangeEnd (yyyy-mm-dd) DepartureTimeRangeStart (hh:mm) DepartureTimeRangeEnd (hh:mm) Yield (Double) AirlineCodeList (List of 2-char airline code) ClassList (List of 1-char class code) Grammar for the Yield-Rule parser.

Definition at line 387 of file YieldParserHelper.cpp.

24.45.2 Constructor & Destructor Documentation

24.45.2.1 AIRRAC::YieldParserHelper::YieldRuleParser::YieldRuleParser (stdair::BomRoot & *ioBomRoot*, YieldRuleStruct & *ioYieldRule*) `[inline]`

Definition at line 391 of file YieldParserHelper.cpp.

References _bomRoot, AIRRAC::YieldRuleStruct::_itDay, AIRRAC::YieldRuleStruct::_itHours, AIRRAC::YieldRuleStruct::_itMinutes, AIRRAC::YieldRuleStruct::_itMonth, AIRRAC::YieldRuleStruct::_itSeconds, AIRRAC::YieldRuleStruct::_itYear, _yieldRule, cabinCode, channel, comments, date, dateRangeEnd, dateRangeStart, AIRRAC::YieldParserHelper::day_p, destination, AIRRAC::YieldParserHelper::hour_p, AIRRAC::YieldParserHelper::minute_p, AIRRAC::YieldParserHelper::month_p, origin, point_of_sale, AIRRAC::YieldParserHelper::second_p, segment, start, time, timeRangeEnd, timeRangeStart, tripType, AIRRAC::YieldParserHelper::uint1_4_p, AIRRAC::YieldParserHelper::year_p, yield, yield_id, yield_rule, and yield_rule_end.

24.45.3 Member Data Documentation

24.45.3.1 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRRAC::YieldParserHelper::YieldRuleParser::start

Definition at line 487 of file YieldParserHelper.cpp.

Referenced by YieldRuleParser().

24.45.3.2 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRRAC::YieldParserHelper::YieldRuleParser::comments

Definition at line 487 of file YieldParserHelper.cpp.

Referenced by YieldRuleParser().

24.45.3.3 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRRAC::YieldParserHelper::YieldRuleParser::yield_rule

Definition at line 487 of file YieldParserHelper.cpp.

Referenced by YieldRuleParser().

24.45.3.4 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRRAC::YieldParserHelper::YieldRuleParser::yield_id

Definition at line 487 of file YieldParserHelper.cpp.

Referenced by YieldRuleParser().

24.45.3.5 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRRAC::YieldParserHelper::YieldRuleParser::origin

Definition at line 487 of file YieldParserHelper.cpp.

Referenced by YieldRuleParser().

24.45.3.6 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRRAC::YieldParserHelper::YieldRuleParser::destination

Definition at line 487 of file YieldParserHelper.cpp.

Referenced by YieldRuleParser().

24.45.3.7 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRRAC::YieldParserHelper::YieldRuleParser::tripType

Definition at line 487 of file YieldParserHelper.cpp.

Referenced by YieldRuleParser().

24.45.3.8 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRRAC::YieldParserHelper::YieldRuleParser::dateRangeStart

Definition at line 487 of file YieldParserHelper.cpp.

Referenced by YieldRuleParser().

24.45.3.9 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRRAC::YieldParserHelper::YieldRuleParser::dateRangeEnd

Definition at line 487 of file YieldParserHelper.cpp.

Referenced by YieldRuleParser().

24.45.3.10 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRRAC::YieldParserHelper::YieldRuleParser::date

Definition at line 487 of file YieldParserHelper.cpp.

Referenced by YieldRuleParser().

24.45.3.11 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRRAC::YieldParserHelper::YieldRuleParser::timeRangeStart

Definition at line 487 of file YieldParserHelper.cpp.

Referenced by YieldRuleParser().

24.45.3.12 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRRAC::YieldParserHelper::YieldRuleParser::timeRangeEnd

Definition at line 487 of file YieldParserHelper.cpp.

Referenced by YieldRuleParser().

24.45.3.13 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRRAC::YieldParserHelper::YieldRuleParser::time

Definition at line 487 of file YieldParserHelper.cpp.

Referenced by YieldRuleParser().

24.45.3.14 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRRAC::YieldParserHelper::YieldRuleParser::point_of_sale

Definition at line 487 of file YieldParserHelper.cpp.

Referenced by YieldRuleParser().

24.45.3.15 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRRAC::YieldParserHelper::YieldRuleParser::cabinCode

Definition at line 487 of file YieldParserHelper.cpp.

Referenced by YieldRuleParser().

24.45.3.16 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRRAC::YieldParserHelper::YieldRuleParser::channel

Definition at line 487 of file YieldParserHelper.cpp.

Referenced by YieldRuleParser().

24.45.3.17 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRRAC::YieldParserHelper::YieldRuleParser::yield

Definition at line 487 of file YieldParserHelper.cpp.

Referenced by YieldRuleParser().

24.45.3.18 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRRAC::YieldParserHelper::YieldRuleParser::segment

Definition at line 487 of file YieldParserHelper.cpp.

Referenced by YieldRuleParser().

24.45.3.19 boost::spirit::qi::rule<stdair::iterator_t, boost::spirit::ascii::space_type> AIRRAC::YieldParserHelper::YieldRuleParser::yield_rule_end

Definition at line 487 of file YieldParserHelper.cpp.

Referenced by YieldRuleParser().

24.45.3.20 stdair::BomRoot& AIRRAC::YieldParserHelper::YieldRuleParser::_bomRoot

Definition at line 493 of file YieldParserHelper.cpp.

Referenced by YieldRuleParser().

24.45.3.21 YieldRuleStruct& AIRRAC::YieldParserHelper::YieldRuleParser::_yieldRule

Definition at line 494 of file YieldParserHelper.cpp.

Referenced by YieldRuleParser().

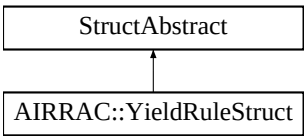
The documentation for this struct was generated from the following file:

- airrac/command/YieldParserHelper.cpp

24.46 AIRRAC::YieldRuleStruct Struct Reference

Utility Structure for the parsing of Flight-Date structures.

`#include <airrac/bom/YieldRuleStruct.hpp>`Inheritance diagram for AIRRAC::YieldRuleStruct::



Public Member Functions

- YieldRuleStruct ()
- ~YieldRuleStruct ()
- AIRRAC::YieldID_T getYieldID () const
- stdair::AirportCode_T getOrigin () const
- stdair::AirportCode_T getDestination () const
- stdair::TripType_T getTripType () const
- stdair::Date_T getDateRangeStart () const
- stdair::Date_T getDateRangeEnd () const
- stdair::Duration_T getTimeRangeStart () const
- stdair::Duration_T getTimeRangeEnd () const

- stdair::CabinCode_T getCabinCode () const
- const stdair::CityCode_T getPOS () const
- stdair::ChannelLabel_T getChannel () const
- stdair::YieldValue_T getYield () const
- stdair::AirlineCode_T getAirlineCode () const
- stdair::ClassCode_T getClassCode () const
- const unsigned int getAirlineListSize () const
- const unsigned int getClassCodeListSize () const
- stdair::AirlineCodeList_T getAirlineList () const
- stdair::ClassList_StringList_T getClassCodeList () const
- stdair::Date_T calculateDate () const
- stdair::Duration_T calculateTime () const
- const std::string describe () const
- void setYieldID (const AIRRAC::YieldID_T iYieldID)
- void setOrigin (const stdair::AirportCode_T &iOrigin)
- void setDestination (const stdair::AirportCode_T &iDestination)
- void setTripType (const stdair::TripType_T &iTripType)
- void setDateRangeStart (const stdair::Date_T &iDateRangeStart)
- void setDateRangeEnd (const stdair::Date_T &iDateRangeEnd)
- void setTimeRangeStart (const stdair::Duration_T &iTimeRangeStart)
- void setTimeRangeEnd (const stdair::Duration_T &iTimeRangeEnd)
- void setCabinCode (const stdair::CabinCode_T &iCabinCode)
- void setPOS (const stdair::CityCode_T &iPOS)
- void setChannel (const stdair::ChannelLabel_T &iChannel)
- void setYield (const stdair::YieldValue_T &iYield)
- void setAirlineCode (const stdair::AirlineCode_T &iAirlineCode)
- void setClassCode (const stdair::ClassCode_T &iClassCode)
- void clearAirlineCodeList ()
- void clearClassCodeList ()
- void addAirlineCode (const stdair::AirlineCode_T &iAirlineCode)
- void addClassCode (const stdair::ClassCode_T &iClassCode)

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

24.46.1 Detailed Description

Utility Structure for the parsing of Flight-Date structures.

Definition at line 24 of file YieldRuleStruct.hpp.

24.46.2 Constructor & Destructor Documentation

24.46.2.1 AIRRAC::YieldRuleStruct::YieldRuleStruct ()

Constructor.

Definition at line 17 of file YieldRuleStruct.cpp.

24.46.2.2 AIRRAC::YieldRuleStruct::~YieldRuleStruct ()

Destructor.

Definition at line 34 of file YieldRuleStruct.cpp.

24.46.3 Member Function Documentation

24.46.3.1 AIRRAC::YieldID_T AIRRAC::YieldRuleStruct::getYieldID () const [inline]

Get the yield ID.

Definition at line 40 of file YieldRuleStruct.hpp.

24.46.3.2 stdair::AirportCode_T AIRRAC::YieldRuleStruct::getOrigin () const [inline]

Get the origin.

Definition at line 45 of file YieldRuleStruct.hpp.

Referenced by AIRRAC::YieldParserHelper::storePOS::operator()().

24.46.3.3 stdair::AirportCode_T AIRRAC::YieldRuleStruct::getDestination () const [inline]

Get the destination.

Definition at line 50 of file YieldRuleStruct.hpp.

Referenced by AIRRAC::YieldParserHelper::storePOS::operator()().

24.46.3.4 stdair::TripType_T AIRRAC::YieldRuleStruct::getTripType () const [inline]

Get the trip type.

Definition at line 55 of file YieldRuleStruct.hpp.

24.46.3.5 stdair::Date_T AIRRAC::YieldRuleStruct::getDateRangeStart () const [inline]

Get the date range start.

Definition at line 60 of file YieldRuleStruct.hpp.

24.46.3.6 stdair::Date_T AIRRAC::YieldRuleStruct::getDateRangeEnd () const [inline]

Get the date range end.

Definition at line 65 of file YieldRuleStruct.hpp.

24.46.3.7 stdair::Duration_T AIRRAC::YieldRuleStruct::getTimeRangeStart () const [inline]

Get the time range start.

Definition at line 70 of file YieldRuleStruct.hpp.

24.46.3.8 stdair::Duration_T AIRRAC::YieldRuleStruct::getTimeRangeEnd () const [inline]

Get the time range end.

Definition at line 75 of file YieldRuleStruct.hpp.

24.46.3.9 stdair::CabinCode_T AIRRAC::YieldRuleStruct::getCabinCode () const [inline]

Get the cabin code.

Definition at line 80 of file YieldRuleStruct.hpp.

24.46.3.10 const stdair::CityCode_T AIRRAC::YieldRuleStruct::getPOS () const [inline]

Get the point-of-sale.

Definition at line 85 of file YieldRuleStruct.hpp.

24.46.3.11 stdair::ChannelLabel_T AIRRAC::YieldRuleStruct::getChannel () const [inline]

Get the channel.

Definition at line 90 of file YieldRuleStruct.hpp.

24.46.3.12 stdair::YieldValue_T AIRRAC::YieldRuleStruct::getYield () const [inline]

Get the yield.

Definition at line 95 of file YieldRuleStruct.hpp.

24.46.3.13 stdair::AirlineCode_T AIRRAC::YieldRuleStruct::getAirlineCode () const [inline]

Get the airline code.

Definition at line 100 of file YieldRuleStruct.hpp.

24.46.3.14 stdair::ClassCode_T AIRRAC::YieldRuleStruct::getClassCode () const [inline]

Get the class code.

Definition at line 105 of file YieldRuleStruct.hpp.

24.46.3.15 const unsigned int AIRRAC::YieldRuleStruct::getAirlineListSize () const [inline]

Get the size of the airline code list.

Definition at line 110 of file YieldRuleStruct.hpp.

24.46.3.16 const unsigned int AIRRAC::YieldRuleStruct::getClassCodeListSize () const [inline]

Get the size of the class code list.

Definition at line 115 of file YieldRuleStruct.hpp.

24.46.3.17 stdair::AirlineCodeList_T AIRRAC::YieldRuleStruct::getAirlineList () const [inline]

Get the airline code list.

Definition at line 120 of file YieldRuleStruct.hpp.

24.46.3.18 stdair::ClassList_StringList_T AIRRAC::YieldRuleStruct::getClassCodeList () const [inline]

Get the class code list.

Definition at line 125 of file YieldRuleStruct.hpp.

24.46.3.19 stdair::Date_T AIRRAC::YieldRuleStruct::calculateDate () const

Calculate the date from the staging details.

Definition at line 38 of file YieldRuleStruct.cpp.

References _itDay, _itMonth, and _itYear.

Referenced by AIRRAC::YieldParserHelper::storeDateRangeEnd::operator()(), and AIRRAC::YieldParserHelper::storeDateRangeStart::operator()().

24.46.3.20 stdair::Duration_T AIRRAC::YieldRuleStruct::calculateTime () const

Calculate the time from the staging details.

Definition at line 44 of file YieldRuleStruct.cpp.

References _itHours, _itMinutes, and _itSeconds.

Referenced by AIRRAC::YieldParserHelper::storeEndRangeTime::operator()(), and AIRRAC::YieldParserHelper::storeStartRangeTime::operator()().

24.46.3.21 const std::string AIRRAC::YieldRuleStruct::describe () const

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

Definition at line 52 of file YieldRuleStruct.cpp.

Referenced by AIRRAC::YieldParserHelper::doEndYield::operator()().

24.46.3.22 void AIRRAC::YieldRuleStruct::setYieldID (const AIRRAC::YieldID_T *iYieldID*) [inline]

Set the yield ID.

Definition at line 143 of file YieldRuleStruct.hpp.

Referenced by AIRRAC::YieldParserHelper::storeYieldId::operator()().

24.46.3.23 void AIRRAC::YieldRuleStruct::setOrigin (const stdair::AirportCode_T & *iOrigin*) [inline]

Set the origin.

Definition at line 148 of file YieldRuleStruct.hpp.

Referenced by AIRRAC::YieldParserHelper::storeOrigin::operator()().

24.46.3.24 void AIRRAC::YieldRuleStruct::setDestination (const stdair::AirportCode_T & *iDestination*) [inline]

Set the destination.

Definition at line 153 of file YieldRuleStruct.hpp.

Referenced by AIRRAC::YieldParserHelper::storeDestination::operator()().

24.46.3.25 void AIRRAC::YieldRuleStruct::setTripType (const stdair::TripType_T & *iTripType*) [inline]

Set the trip type.

Definition at line 158 of file YieldRuleStruct.hpp.

Referenced by AIRRAC::YieldParserHelper::storeTripType::operator()().

24.46.3.26 void AIRRAC::YieldRuleStruct::setDateRangeStart (const stdair::Date_T & *iDateRangeStart*) [inline]

Set the date range start.

Definition at line 163 of file YieldRuleStruct.hpp.

Referenced by AIRRAC::YieldParserHelper::storeDateRangeStart::operator()().

24.46.3.27 void AIRRAC::YieldRuleStruct::setDateRangeEnd (const stdair::Date_T & *iDateRangeEnd*) [inline]

Set the date range end.

Definition at line 168 of file YieldRuleStruct.hpp.

Referenced by AIRRAC::YieldParserHelper::storeDateRangeEnd::operator()().

24.46.3.28 void AIRRAC::YieldRuleStruct::setTimeRangeStart (const stdair::Duration_T & *iTimeRangeStart*) [inline]

Set the time range start.

Definition at line 173 of file YieldRuleStruct.hpp.

Referenced by AIRRAC::YieldParserHelper::storeStartRangeTime::operator()().

24.46.3.29 void AIRRAC::YieldRuleStruct::setTimeRangeEnd (const stdair::Duration_T & *iTimeRangeEnd*) [inline]

Set the time range end.

Definition at line 178 of file YieldRuleStruct.hpp.

Referenced by AIRRAC::YieldParserHelper::storeEndRangeTime::operator()().

24.46.3.30 void AIRRAC::YieldRuleStruct::setCabinCode (const stdair::CabinCode_T & *iCabinCode*) [inline]

Set the cabin code.

Definition at line 183 of file YieldRuleStruct.hpp.

Referenced by AIRRAC::YieldParserHelper::storeCabinCode::operator()().

24.46.3.31 void AIRRAC::YieldRuleStruct::setPOS (const stdair::CityCode_T & *iPOS*) [inline]

Set the point-of-sale.

Definition at line 188 of file YieldRuleStruct.hpp.

Referenced by AIRRAC::YieldParserHelper::storePOS::operator()().

24.46.3.32 void AIRRAC::YieldRuleStruct::setChannel (const stdair::ChannelLabel_T & *iChannel*) [inline]

Set the channel.

Definition at line 193 of file YieldRuleStruct.hpp.

Referenced by AIRRAC::YieldParserHelper::storeChannel::operator()().

24.46.3.33 void AIRRAC::YieldRuleStruct::setYield (const stdair::YieldValue_T & *iYield*) [inline]

Set the yield.

Definition at line 198 of file YieldRuleStruct.hpp.

Referenced by AIRRAC::YieldParserHelper::storeYield::operator()().

24.46.3.34 void AIRRAC::YieldRuleStruct::setAirlineCode (const stdair::AirlineCode_T & *iAirlineCode*) [inline]

Set the airline code.

Definition at line 203 of file YieldRuleStruct.hpp.

Referenced by AIRRAC::YieldParserHelper::storeAirlineCode::operator()(), and AIRRAC::YieldParserHelper::storeYieldId::operator()().

24.46.3.35 void AIRRAC::YieldRuleStruct::setClassCode (const stdair::ClassCode_T & *iClassCode*) [inline]

Set the class code.

Definition at line 208 of file YieldRuleStruct.hpp.

Referenced by AIRRAC::YieldParserHelper::storeYieldId::operator()().

24.46.3.36 void AIRRAC::YieldRuleStruct::clearAirlineCodeList () [inline]

Empty the airline code list.

Definition at line 213 of file YieldRuleStruct.hpp.

Referenced by AIRRAC::YieldParserHelper::storeYieldId::operator()().

24.46.3.37 void AIRRAC::YieldRuleStruct::clearClassCodeList () [inline]

Empty the class code list.

Definition at line 218 of file YieldRuleStruct.hpp.

Referenced by AIRRAC::YieldParserHelper::storeYieldId::operator()().

24.46.3.38 void AIRRAC::YieldRuleStruct::addAirlineCode (const stdair::AirlineCode_T & *iAirlineCode*) [inline]

Add an airline code to the list.

Definition at line 223 of file YieldRuleStruct.hpp.

Referenced by AIRRAC::YieldParserHelper::storeAirlineCode::operator()().

24.46.3.39 void AIRRAC::YieldRuleStruct::addClassCode (const stdair::ClassCode_T & *iClassCode*) [inline]

Add a class code to the list.

Definition at line 228 of file YieldRuleStruct.hpp.

Referenced by AIRRAC::YieldParserHelper::storeClass::operator()().

24.46.4 Member Data Documentation

24.46.4.1 stdair::year_t AIRRAC::YieldRuleStruct::_itYear

Staging Date.

Definition at line 235 of file YieldRuleStruct.hpp.

Referenced by calculateDate(), and AIRRAC::YieldParserHelper::YieldRuleParser::YieldRuleParser().

24.46.4.2 stdair::month_t AIRRAC::YieldRuleStruct::_itMonth

Definition at line 236 of file YieldRuleStruct.hpp.

Referenced by calculateDate(), and AIRRAC::YieldParserHelper::YieldRuleParser::YieldRuleParser().

24.46.4.3 stdair::day_t AIRRAC::YieldRuleStruct::_itDay

Definition at line 237 of file YieldRuleStruct.hpp.

Referenced by calculateDate(), and AIRRAC::YieldParserHelper::YieldRuleParser::YieldRuleParser().

24.46.4.4 stdair::hour_t AIRRAC::YieldRuleStruct::_itHours

Staging Time.

Definition at line 241 of file YieldRuleStruct.hpp.

Referenced by calculateTime(), and AIRRAC::YieldParserHelper::YieldRuleParser::YieldRuleParser().

24.46.4.5 stdair::minute_t AIRRAC::YieldRuleStruct::_itMinutes

Definition at line 242 of file YieldRuleStruct.hpp.

Referenced by calculateTime(), and AIRRAC::YieldParserHelper::YieldRuleParser::YieldRuleParser().

24.46.4.6 stdair::second_t AIRRAC::YieldRuleStruct::_itSeconds

Definition at line 243 of file YieldRuleStruct.hpp.

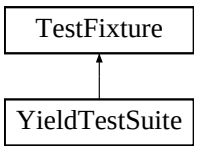
Referenced by calculateTime(), AIRRAC::YieldParserHelper::storeEndRangeTime::operator()(), AIRRAC::YieldParserHelper::storeStartRangeTime::operator()(), AIRRAC::YieldParserHelper::storeYieldId::operator()(), and AIRRAC::YieldParserHelper::YieldRuleParser::YieldRuleParser().

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

- airrac/bom/YieldRuleStruct.hpp
- airrac/bom/YieldRuleStruct.cpp

24.47 YieldTestSuite Class Reference

`#include <test/airrac/YieldTestSuite.hpp>`Inheritance diagram for YieldTestSuite::



Public Member Functions

- void simpleYield ()
- YieldTestSuite ()

Protected Attributes

- std::stringstream _describeKey

24.47.1 Detailed Description

Utility class for CPPUnit-based testing.

Definition at line 7 of file YieldTestSuite.hpp.

24.47.2 Constructor & Destructor Documentation

24.47.2.1 YieldTestSuite::YieldTestSuite ()

Test some error detection functionalities. Constructor.

24.47.3 Member Function Documentation

24.47.3.1 void YieldTestSuite::simpleYield ()

Test a simple yield functionality.

24.47.4 Member Data Documentation

24.47.4.1 std::stringstream YieldTestSuite::_describeKey `[protected]`

Definition at line 28 of file YieldTestSuite.hpp.

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

- test/airrac/YieldTestSuite.hpp

25 File Documentation

25.1 airrac/AIRRAC_Service.hpp File Reference

```
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_service_types.hpp>
#include <stdair/bom/TravelSolutionTypes.hpp>
#include <airrac/AIRRAC_Types.hpp>
```

Classes

- class AIRRAC::AIRRAC_Service
  Interface for the AIRRAC Services.

Namespaces

- namespace stdair
  Forward declarations.
- namespace AIRRAC

25.2 AIRRAC_Service.hpp

```
00001 #ifndef __AIRRAC_SVC_AIRRAC_SERVICE_HPP
00002 #define __AIRRAC_SVC_AIRRAC_SERVICE_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009 #include <stdair/stdair_service_types.hpp>
00010 #include <stdair/bom/TravelSolutionTypes.hpp>
00011 // AirRAC
00012 #include <airrac/AIRRAC_Types.hpp>
00013
00014 // Forward declarations.
00015 namespace stdair {
00016   class STDAIR_Service;
00017   class BomRoot;
00018   struct BasLogParams;
00019   struct BasDBParams;
00020 }
00021
00022 namespace AIRRAC {
00023
00025   class AIRRAC_ServiceContext;
00026
00030   class AIRRAC_Service {
00031   public:
00032     // ////////////////// Constructors and Destructors /////////////////
00033
00045     AIRRAC_Service (const stdair::BasLogParams&);
00046
00059     AIRRAC_Service (const stdair::BasLogParams&, const stdair::BasDBParams&);
00060
00076     AIRRAC_Service (stdair::STDAIR_ServicePtr_T ioSTDAIR_ServicePtr);
00077
00086     void parseAndLoad (const YieldFilePath& iYieldFilename);
00087
00088
00092     ~AIRRAC_Service();
00093
00094
00095   public:
00096     // /////////// Business Methods /////////////
00100     void calculateYields (stdair::TravelSolutionList_T&);
00101
00105     void updateYields(stdair::BomRoot&);
00106
00113     void buildSampleBom();
00114
00118     void clonePersistentBom();
00119
00124     void buildComplementaryLinks (stdair::BomRoot&);
00125
00143     void buildSampleTravelSolutions (stdair::TravelSolutionList_T&);
00144
00145
00146   public:
00147     // ///////////////// Display support methods /////////////////
00155     std::string csvDisplay() const;
00156
00164     std::string csvDisplay (const stdair::TravelSolutionList_T&) const;
00165
00166
00167   private:
00168     // /////// Construction and Destruction helper methods ///////
```

```
00172     AIRRAC_Service();
00173
00177     AIRRAC_Service (const AIRRAC_Service&);
00178
00183     void initServiceContext();
00184
00194     stdair::STDAIR_ServicePtr_T initStdAirService (const stdair::BasLogParams&,
00195                                                    const stdair::BasDBParams&);
00196
00205     stdair::STDAIR_ServicePtr_T initStdAirService (const stdair::BasLogParams&);
00206
00215     void addStdAirService (stdair::STDAIR_ServicePtr_T,
00216                            const bool iOwnStdairService);
00217
00224     void initAirracService();
00225
00234     void initAirracService (const YieldFilePath& iYieldFilename);
00235
00239     void finalise();
00240
00241
00242   private:
00243     // ///////// Service Context /////////
00247     AIRRAC_ServiceContext* _airracServiceContext;
00248   };
00249 }
00250 #endif // __AIRRAC_SVC_AIRRAC_SERVICE_HPP
```

25.3 airrac/AIRRAC_Types.hpp File Reference

```
#include <vector>
#include <string>
#include <boost/shared_ptr.hpp>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/stdair_file.hpp>
```

Classes

- class AIRRAC::AirportPairNotFoundException
- class AIRRAC::PosOrChannelNotFoundException
- class AIRRAC::FlightDateNotFoundException
- class AIRRAC::FlightTimeNotFoundException
- class AIRRAC::FeaturesNotFoundException
- class AIRRAC::AirlineNotFoundException
- class AIRRAC::YieldInputFileNotFoundException
- class AIRRAC::YieldFileParsingFailedException
- class AIRRAC::QuotingException
- class AIRRAC::YieldFilePath

Namespaces

- namespace AIRRAC

Typedefs

- typedef boost::shared_ptr< AIRRAC_Service > AIRRAC::AIRRAC_ServicePtr_T
- typedef unsigned int AIRRAC::YieldID_T

25.4 AIRRAC_Types.hpp

```
00001 #ifndef __AIRRAC_AIRRAC_TYPES_HPP
00002 #define __AIRRAC_AIRRAC_TYPES_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <vector>
00009 #include <string>
00010 // Boost
00011 #include <boost/shared_ptr.hpp>
00012 // StdAir
00013 #include <stdair/stdair_exceptions.hpp>
00014 #include <stdair/stdair_file.hpp>
00015
00016 namespace AIRRAC {
00017
00018   // ********* Exceptions ***********
00019   class AirportPairNotFoundException : public stdair::ObjectNotFoundException {
00020   public:
00022     AirportPairNotFoundException (const std::string& iWhat)
00023       : stdair::ObjectNotFoundException (iWhat) {}
00024   };
00025
00026   class PosOrChannelNotFoundException : public stdair::ObjectNotFoundException {
00027   public:
00029     PosOrChannelNotFoundException (const std::string& iWhat)
00030       : stdair::ObjectNotFoundException (iWhat) {}
00031   };
00032
00033   class FlightDateNotFoundException : public stdair::ObjectNotFoundException {
00034   public:
00036     FlightDateNotFoundException (const std::string& iWhat)
00037       : stdair::ObjectNotFoundException (iWhat) {}
00038   };
00039
00040   class FlightTimeNotFoundException : public stdair::ObjectNotFoundException {
00041   public:
00043     FlightTimeNotFoundException (const std::string& iWhat)
00044       : stdair::ObjectNotFoundException (iWhat) {}
00045   };
00046
00047   class FeaturesNotFoundException : public stdair::ObjectNotFoundException {
00048   public:
00050     FeaturesNotFoundException (const std::string& iWhat)
00051       : stdair::ObjectNotFoundException (iWhat) {}
00052   };
00053
00054   class AirlineNotFoundException : public stdair::ObjectNotFoundException {
00055   public:
00057     AirlineNotFoundException (const std::string& iWhat)
00058       : stdair::ObjectNotFoundException (iWhat) {}
00059   };
00060
00061   class YieldInputFileNotFoundException : public stdair::FileNotFoundException {
00062   public:
00064     YieldInputFileNotFoundException (const std::string& iWhat)
00065       : stdair::FileNotFoundException (iWhat) {}
00066   };
00067
00068   class YieldFileParsingFailedException : public stdair::
      ParsingFileFailedException {
00069   public:
00071     YieldFileParsingFailedException (const std::string& iWhat)
00072       : stdair::ParsingFileFailedException (iWhat) {}
```

```
00073   };
00074
00075   class QuotingException : public stdair::RootException {
00076   };
00077
00078   // ///////// Files ///////////
00082   class YieldFilePath : public stdair::InputFilePath {
00083   public:
00087     explicit YieldFilePath (const stdair::Filename_T& iFilename)
00088       : stdair::InputFilePath (iFilename) {}
00089   };
00090
00091   // //////// Type definitions specific to AirRAC /////////
00095   class AIRRAC_Service;
00096   typedef boost::shared_ptr<AIRRAC_Service> AIRRAC_ServicePtr_T;
00097
00098
00102   typedef unsigned int YieldID_T;
00103 }
00104 #endif // __AIRRAC_AIRRAC_TYPES_HPP
00105
```

25.5 airrac/basic/BasConst.cpp File Reference

```
#include <airrac/basic/BasConst_General.hpp>
#include <airrac/basic/BasConst_AIRRAC_Service.hpp>
```

Namespaces

- namespace AIRRAC

Variables

- const std::string AIRRAC::DEFAULT_AIRLINE_CODE = "BA"

25.6 BasConst.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 #include <airrac/basic/BasConst_General.hpp>
00005 #include <airrac/basic/BasConst_AIRRAC_Service.hpp>
00006
00007 namespace AIRRAC {
00008
00010   const std::string DEFAULT_AIRLINE_CODE = "BA";
00011
00012 }
```

25.7 airrac/basic/BasConst_AIRRAC_Service.hpp File Reference

```
#include <string>
```

Namespaces

- namespace AIRRAC

25.8 BasConst_AIRRAC_Service.hpp

```
00001 #ifndef __AIRRAC_BAS_BASCONST_AIRRAC_SERVICE_HPP
00002 #define __AIRRAC_BAS_BASCONST_AIRRAC_SERVICE_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 #include <string>
00008
00009 namespace AIRRAC {
00010
00012   extern const std::string DEFAULT_AIRLINE_CODE;
00013
00014 }
00015 #endif // __AIRRAC_BAS_BASCONST_AIRRAC_SERVICE_HPP
```

25.9 airrac/basic/BasConst_General.hpp File Reference

Namespaces

- namespace AIRRAC

25.10 BasConst_General.hpp

```
00001 #ifndef __AIRRAC_BAS_BASCONST_GENERAL_HPP
00002 #define __AIRRAC_BAS_BASCONST_GENERAL_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007
00008 namespace AIRRAC {
00009
00010 }
00011 #endif // __AIRRAC_BAS_BASCONST_GENERAL_HPP
```

25.11 airrac/batches/airrac.cpp File Reference

```
#include <cassert>
#include <iostream>
#include <sstream>
#include <fstream>
#include <vector>
#include <list>
#include <string>
#include <boost/date_time/posix_time/posix_time.hpp>
#include <boost/date_time/gregorian/gregorian.hpp>
#include <boost/tokenizer.hpp>
#include <boost/program_options.hpp>
#include <stdair/STDAIR_Service.hpp>
#include <stdair/bom/TravelSolutionStruct.hpp>
#include <stdair/service/Logger.hpp>
#include <airrac/AIRRAC_Service.hpp>
#include <airrac/config/airrac-paths.hpp>
```

Typedefs

- typedef std::vector< std::string > WordList_T

Functions

- const std::string K_AIRRAC_DEFAULT_LOG_FILENAME ("airrac.log")
- const std::string K_AIRRAC_DEFAULT_YIELD_INPUT_FILENAME (STDAIR_SAMPLE_DIR"/yieldstore01.csv")
- template<class T >
  std::ostream & operator<< (std::ostream &os, const std::vector< T > &v)
- int readConfiguration (int argc, char *argv[ ], bool &ioIsBuiltin, stdair::Filename_T &ioYieldInputFilename, std::string &ioLogFilename)
- int main (int argc, char *argv[ ])

Variables

- const bool K_AIRRAC_DEFAULT_BUILT_IN_INPUT = false
- const int K_AIRRAC_EARLY_RETURN_STATUS = 99

25.11.1 Typedef Documentation

25.11.1.1 typedef std::vector<std::string> WordList_T

Definition at line 23 of file airrac.cpp.

25.11.2 Function Documentation

25.11.2.1 const std::string K_AIRRAC_DEFAULT_LOG_FILENAME ("airrac.log")

Default name and location for the log file.

Referenced by readConfiguration().

25.11.2.2 const std::string K_AIRRAC_DEFAULT_YIELD_INPUT_FILENAME (STDAIR_SAMPLE_DIR"/yieldstore01.csv")

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

Referenced by readConfiguration().

25.11.2.3 template<class T > std::ostream& operator<< (std::ostream & *os*, const std::vector< T > & *v*) [inline]

Definition at line 43 of file airrac.cpp.

25.11.2.4 int readConfiguration (int *argc*, char * *argv*[ ], bool & *ioIsBuiltin*, stdair::Filename_T & *ioYieldInputFilename*, std::string & *ioLogFilename*)

Read and parse the command line options.

Definition at line 50 of file airrac.cpp.

References K_AIRRAC_DEFAULT_BUILT_IN_INPUT, K_AIRRAC_DEFAULT_LOG_FILENAME(), K_AIRRAC_DEFAULT_YIELD_INPUT_FILENAME(), K_AIRRAC_EARLY_RETURN_STATUS, PACKAGE_NAME, PACKAGE_VERSION, and PREFIXDIR.

Referenced by main().

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

Definition at line 153 of file airrac.cpp.

References AIRRAC::AIRRAC_Service::buildSampleBom(), AIRRAC::AIRRAC_Service::buildSampleTravelSolutions(), AIRRAC::AIRRAC_Service::csvDisplay(), K_AIRRAC_EARLY_RETURN_STATUS, AIRRAC::AIRRAC_Service::parseAndLoad(), and readConfiguration().

25.11.3 Variable Documentation

25.11.3.1 const bool K_AIRRAC_DEFAULT_BUILT_IN_INPUT = false

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

Definition at line 36 of file airrac.cpp.

Referenced by readConfiguration().

25.11.3.2 const int K_AIRRAC_EARLY_RETURN_STATUS = 99

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

Definition at line 39 of file airrac.cpp.

Referenced by main(), and readConfiguration().

25.12 airrac.cpp

```
00001 // STL
00002 #include <cassert>
00003 #include <iostream>
00004 #include <sstream>
00005 #include <fstream>
00006 #include <vector>
00007 #include <list>
00008 #include <string>
00009 // Boost (Extended STL)
00010 #include <boost/date_time/posix_time/posix_time.hpp>
00011 #include <boost/date_time/gregorian/gregorian.hpp>
00012 #include <boost/tokenizer.hpp>
00013 #include <boost/program_options.hpp>
00014 // StdAir
00015 #include <stdair/STDAIR_Service.hpp>
00016 #include <stdair/bom/TravelSolutionStruct.hpp>
00017 #include <stdair/service/Logger.hpp>
00018 // Airrac
00019 #include <airrac/AIRRAC_Service.hpp>
00020 #include <airrac/config/airrac-paths.hpp>
00021
00022 // //////// Type definitions ///////
00023 typedef std::vector<std::string> WordList_T;
00024
00025
00026 // //////// Constants //////
00028 const std::string K_AIRRAC_DEFAULT_LOG_FILENAME ("airrac.log");
00029
00031 const std::string K_AIRRAC_DEFAULT_YIELD_INPUT_FILENAME (STDAIR_SAMPLE_DIR
00032                                                          "/yieldstore01.csv");
00033
00036 const bool K_AIRRAC_DEFAULT_BUILT_IN_INPUT = false;
00037
00039 const int K_AIRRAC_EARLY_RETURN_STATUS = 99;
00040
00041 // ///////// Parsing of Options & Configuration /////////
00042 // A helper function to simplify the main part.
00043 template<class T> std::ostream& operator<< (std::ostream& os,
00044                                             const std::vector<T>& v) {
00045   std::copy (v.begin(), v.end(), std::ostream_iterator<T> (std::cout, " "));
00046   return os;
00047 }
00048
00050 int readConfiguration (int argc, char* argv[], bool& ioIsBuiltin,
00051                        stdair::Filename_T& ioYieldInputFilename,
00052                        std::string& ioLogFilename) {
00053
00054   // Default for the built-in input
00055   ioIsBuiltin = K_AIRRAC_DEFAULT_BUILT_IN_INPUT;
00056
00057   // Declare a group of options that will be allowed only on command line
00058   boost::program_options::options_description generic ("Generic options");
00059   generic.add_options()
00060     ("prefix", "print installation prefix")
00061     ("version,v", "print version string")
00062     ("help,h", "produce help message");
00063
00064   // Declare a group of options that will be allowed both on command
00065   // line and in config file
00066   boost::program_options::options_description config ("Configuration");
00067   config.add_options()
00068     ("builtin,b",
00069      "The sample BOM tree can be either built-in or parsed from an input file. Th
      at latter must then be given with the -y/--yield option")
00070     ("yield,y",
```

```
00071      boost::program_options::value< std::string >(&ioYieldInputFilename)->default
     _value(K_AIRRAC_DEFAULT_YIELD_INPUT_FILENAME),
00072      "(CSV) input file for the yield rules")
00073     ("log,l",
00074      boost::program_options::value< std::string >(&ioLogFilename)->default_value(
     K_AIRRAC_DEFAULT_LOG_FILENAME),
00075      "Filename for the logs")
00076     ;
00077
00078   // Hidden options, will be allowed both on command line and
00079   // in config file, but will not be shown to the user.
00080   boost::program_options::options_description hidden ("Hidden options");
00081   hidden.add_options()
00082     ("copyright",
00083      boost::program_options::value< std::vector<std::string> >(),
00084      "Show the copyright (license)");
00085
00086   boost::program_options::options_description cmdline_options;
00087   cmdline_options.add(generic).add(config).add(hidden);
00088
00089   boost::program_options::options_description config_file_options;
00090   config_file_options.add(config).add(hidden);
00091
00092   boost::program_options::options_description visible ("Allowed options");
00093   visible.add(generic).add(config);
00094
00095   boost::program_options::positional_options_description p;
00096   p.add ("copyright", -1);
00097
00098   boost::program_options::variables_map vm;
00099   boost::program_options::
00100     store (boost::program_options::command_line_parser (argc, argv).
00101            options (cmdline_options).positional(p).run(), vm);
00102
00103   std::ifstream ifs ("airrac.cfg");
00104   boost::program_options::store (parse_config_file (ifs, config_file_options),
00105                                  vm);
00106   boost::program_options::notify (vm); if (vm.count ("help")) {
00107     std::cout << visible << std::endl;
00108     return K_AIRRAC_EARLY_RETURN_STATUS;
00109   }
00110
00111   if (vm.count ("version")) {
00112     std::cout << PACKAGE_NAME << ", version " << PACKAGE_VERSION << std::endl;
00113     return K_AIRRAC_EARLY_RETURN_STATUS;
00114   }
00115
00116   if (vm.count ("prefix")) {
00117     std::cout << "Installation prefix: " << PREFIXDIR << std::endl;
00118     return K_AIRRAC_EARLY_RETURN_STATUS;
00119   }
00120
00121   if (vm.count ("builtin")) {
00122     ioIsBuiltin = true;
00123   }
00124   const std::string isBuiltinStr = (ioIsBuiltin == true)?"yes":"no";
00125   std::cout << "The BOM should be built-in? " << isBuiltinStr << std::endl;
00126
00127   if (ioIsBuiltin == false) {
00128
00129     // The BOM tree should be built from parsing a yield (and O&D) file
00130     if (vm.count ("yield")) {
00131       ioYieldInputFilename = vm["yield"].as< std::string >();
00132       std::cout << "Input yield filename is: " << ioYieldInputFilename
00133                 << std::endl;
00134
00135     } else {
```

```
00136       // The built-in option is not selected. However, no yield file
00137       // is specified
00138       std::cerr << "Either one among the -b/--builtin and -y/--yield "
00139                 << "options must be specified" << std::endl;
00140     }
00141   }
00142
00143   if (vm.count ("log")) {
00144     ioLogFilename = vm["log"].as< std::string >();
00145     std::cout << "Log filename is: " << ioLogFilename << std::endl;
00146   }
00147
00148   return 0;
00149 }
00150
00151
00152 // /////////////// M A I N /////////////////
00153 int main (int argc, char* argv[]) {
00154
00155   // State whether the BOM tree should be built-in or parsed from an input file
00156   bool isBuiltin;
00157
00158   // Yield input filename
00159   stdair::Filename_T lYieldInputFilename;
00160
00161   // Output log File
00162   stdair::Filename_T lLogFilename;
00163
00164   // Call the command-line option parser
00165   const int lOptionParserStatus =
00166     readConfiguration (argc, argv, isBuiltin, lYieldInputFilename, lLogFilename);

00167
00168   if (lOptionParserStatus == K_AIRRAC_EARLY_RETURN_STATUS) {
00169     return 0;
00170   }
00171
00172   // Set the log parameters
00173   std::ofstream logOutputFile;
00174   // Open and clean the log outputfile
00175   logOutputFile.open (lLogFilename.c_str());
00176   logOutputFile.clear();
00177
00178   // Initialise the AirRAC service object
00179   const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
00180
00181   AIRRAC::AIRRAC_Service airracService (lLogParams);
00182
00183   // DEBUG
00184   STDAIR_LOG_DEBUG ("Welcome to AirRAC");
00185
00186   // Build a sample list of travel solutions
00187   stdair::TravelSolutionList_T lTravelSolutionList;
00188   airracService.buildSampleTravelSolutions (lTravelSolutionList);
00189
00190   // Check wether or not a (CSV) input file should be read
00191   if (isBuiltin == true) {
00192
00193     // Build the sample BOM tree (filled with yields) for AirRAC
00194     airracService.buildSampleBom();
00195
00196   } else {
00197
00198     // Build the BOM tree from parsing a yield file
00199     AIRRAC::YieldFilePath lYieldFilePath (lYieldInputFilename);
00200     airracService.parseAndLoad (lYieldFilePath);
00201
```

```
00202   }
00203
00204   // DEBUG: Display the whole BOM tree
00205   const std::string& lBOMCSVDump = airracService.csvDisplay();
00206   STDAIR_LOG_DEBUG ("BOM tree: " << lBOMCSVDump);
00207
00208   // DEBUG: Display the travel solutions
00209   const std::string& lTSCSVDump =
00210     airracService.csvDisplay (lTravelSolutionList);
00211   STDAIR_LOG_DEBUG (lTSCSVDump);
00212
00213   // Close the Log outputFile
00214   logOutputFile.close();
00215
00216   /*
00217     Note: as that program is not intended to be run on a server in
00218     production, it is better not to catch the exceptions. When it
00219     happens (that an exception is throwned), that way we get the
00220     call stack.
00221   */
00222
00223   return 0;
00224 }
```

25.13 airrac/bom/YieldRuleStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/service/Logger.hpp>
#include <airrac/AIRRAC_Types.hpp>
#include <airrac/bom/YieldRuleStruct.hpp>
```

Namespaces

- namespace AIRRAC

25.14 YieldRuleStruct.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 // AIRRAC
00011 #include <airrac/AIRRAC_Types.hpp>
00012 #include <airrac/bom/YieldRuleStruct.hpp>
00013
00014 namespace AIRRAC {
00015
00016   // ////////////////////////////////////////////////////////////////////
00017   YieldRuleStruct::YieldRuleStruct()
00018     : _yieldId(0),
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       _yield(0),
00026       _cabinCode(""),
00027       _pos(""),
00028       _channel(""),
00029       _airlineCode(""),
00030       _classCode("") {
00031   }
00032
00033   // ////////////////////////////////////////////////////////////////////
00034   YieldRuleStruct::~YieldRuleStruct() {
00035   }
00036
00037   // ////////////////////////////////////////////////////////////////////
00038   stdair::Date_T YieldRuleStruct::calculateDate() 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 YieldRuleStruct::calculateTime() 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   const std::string YieldRuleStruct::describe() const {
00053     std::ostringstream oStr;
00054     oStr << "YieldRule: " << _yieldId << ", ";
00055
00056     oStr << _origin << "-" << _destination << " ("
00057          << _pos << "), " << _channel << ", [";
00058     oStr << _dateRangeStart << "/" << _dateRangeEnd << "] - ["
00059          << boost::posix_time::to_simple_string (_timeRangeStart) << "/"
00060          << boost::posix_time::to_simple_string (_timeRangeEnd) << "], ";
00061
00062     oStr << _cabinCode << ", " << _yield << " EUR, ";
00063
00064     // Sanity check
00065     assert (_airlineCodeList.size() == _classCodeList.size());
```

```
00066
00067     // Browse the class-pathes
00068     unsigned short idx = 0;
00069     stdair::ClassList_StringList_T::const_iterator itClass =
00070       _classCodeList.begin();
00071     for (stdair::AirlineCodeList_T::const_iterator itAirline =
00072            _airlineCodeList.begin();
00073          itAirline != _airlineCodeList.end(); ++itAirline, ++itClass, ++idx) {
00074       if (idx != 0) {
00075         oStr << " - ";
00076       }
00077       const stdair::AirlineCode_T lAirlineCode = *itAirline;
00078       const stdair::ClassCode_T lClassCode = *itClass;
00079       oStr << lAirlineCode << " / " << lClassCode;
00080     }
00081
00082     return oStr.str();
00083   }
00084 }
```

25.15 airrac/bom/YieldRuleStruct.hpp File Reference

```
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/basic/BasParserHelperTypes.hpp>
#include <airrac/AIRRAC_Types.hpp>
```

Classes

- struct AIRRAC::YieldRuleStruct

  Utility Structure for the parsing of Flight-Date structures.

Namespaces

- namespace AIRRAC

25.16 YieldRuleStruct.hpp

```
00001 #ifndef __AIRRAC_BOM_YIELDRULESTRUCT_HPP
00002 #define __AIRRAC_BOM_YIELDRULESTRUCT_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/stdair_basic_types.hpp>
00011 #include <stdair/stdair_date_time_types.hpp>
00012 #include <stdair/stdair_demand_types.hpp>
00013 #include <stdair/stdair_inventory_types.hpp>
00014 #include <stdair/basic/StructAbstract.hpp>
00015 #include <stdair/basic/BasParserHelperTypes.hpp>
00016 // AirRAC
00017 #include <airrac/AIRRAC_Types.hpp>
00018
00019 namespace AIRRAC {
00020
00024   struct YieldRuleStruct : public stdair::StructAbstract {
00025   public:
00026     // ////////////////// Initialisation / Destruction //////////////////////
00030     YieldRuleStruct();
00031
00035     ~YieldRuleStruct();
00036
00037   public:
00038     // ////////////////////// Getters ///////////////////////////
00040     AIRRAC::YieldID_T getYieldID () const {
00041       return _yieldId;
00042     }
00043
00045     stdair::AirportCode_T getOrigin () const {
00046       return _origin;
00047     }
00048
00050     stdair::AirportCode_T getDestination () const {
00051       return _destination;
00052     }
00053
00055     stdair::TripType_T getTripType () const {
00056       return _tripType;
00057     }
00058
00060     stdair::Date_T getDateRangeStart () const {
00061       return _dateRangeStart;
00062     }
00063
00065     stdair::Date_T getDateRangeEnd () const {
00066       return _dateRangeEnd;
00067     }
00068
00070     stdair::Duration_T getTimeRangeStart () const {
00071       return _timeRangeStart;
00072     }
00073
00075     stdair::Duration_T getTimeRangeEnd () const {
00076       return _timeRangeEnd;
00077     }
00078
00080     stdair::CabinCode_T getCabinCode () const {
00081       return _cabinCode;
00082     }
00083
```

```
00085     const stdair::CityCode_T getPOS () const {
00086       return _pos;
00087     }
00088
00090     stdair::ChannelLabel_T getChannel () const {
00091       return _channel;
00092     }
00093
00095     stdair::YieldValue_T getYield () const {
00096       return _yield;
00097     }
00098
00100     stdair::AirlineCode_T getAirlineCode () const {
00101       return _airlineCode;
00102     }
00103
00105     stdair::ClassCode_T getClassCode () const {
00106       return _classCode;
00107     }
00108
00110     const unsigned int getAirlineListSize () const {
00111       return _airlineCodeList.size();
00112     }
00113
00115     const unsigned int getClassCodeListSize () const {
00116       return _classCodeList.size();
00117     }
00118
00120     stdair::AirlineCodeList_T getAirlineList () const {
00121       return _airlineCodeList;
00122     }
00123
00125     stdair::ClassList_StringList_T getClassCodeList () const {
00126       return _classCodeList;
00127     }
00128
00129   public:
00130     // ////////////////////// Display support methods //////////////////////////
00132     stdair::Date_T calculateDate() const;
00133
00135     stdair::Duration_T calculateTime() const;
00136
00138     const std::string describe() const;
00139
00140   public:
00141     // ///////// Setters //////////
00143     void setYieldID (const AIRRAC::YieldID_T iYieldID) {
00144       _yieldId = iYieldID;
00145     }
00146
00148     void setOrigin (const stdair::AirportCode_T& iOrigin) {
00149       _origin = iOrigin;
00150     }
00151
00153     void setDestination (const stdair::AirportCode_T& iDestination) {
00154       _destination = iDestination;
00155     }
00156
00158     void setTripType (const stdair::TripType_T& iTripType) {
00159       _tripType = iTripType;
00160     }
00161
00163     void setDateRangeStart (const stdair::Date_T& iDateRangeStart) {
00164       _dateRangeStart = iDateRangeStart;
00165     }
00166
00168     void setDateRangeEnd (const stdair::Date_T& iDateRangeEnd) {
```

```
00169       _dateRangeEnd = iDateRangeEnd;
00170     }
00171
00173     void setTimeRangeStart (const stdair::Duration_T& iTimeRangeStart) {
00174       _timeRangeStart = iTimeRangeStart;
00175     }
00176
00178     void setTimeRangeEnd (const stdair::Duration_T& iTimeRangeEnd) {
00179       _timeRangeEnd = iTimeRangeEnd;
00180     }
00181
00183     void setCabinCode (const stdair::CabinCode_T& iCabinCode) {
00184       _cabinCode = iCabinCode;
00185     }
00186
00188     void setPOS (const stdair::CityCode_T& iPOS) {
00189       _pos = iPOS;
00190     }
00191
00193     void setChannel (const stdair::ChannelLabel_T& iChannel) {
00194       _channel = iChannel;
00195     }
00196
00198     void setYield(const stdair::YieldValue_T& iYield) {
00199       _yield = iYield;
00200     }
00201
00203     void setAirlineCode (const stdair::AirlineCode_T&  iAirlineCode) {
00204       _airlineCode = iAirlineCode;
00205     }
00206
00208     void setClassCode (const stdair::ClassCode_T& iClassCode) {
00209       _classCode = iClassCode;
00210     }
00211
00213     void clearAirlineCodeList () {
00214       _airlineCodeList.clear();
00215     }
00216
00218     void clearClassCodeList () {
00219       _classCodeList.clear();
00220     }
00221
00223     void addAirlineCode (const stdair::AirlineCode_T& iAirlineCode)  {
00224       _airlineCodeList.push_back (iAirlineCode);
00225     }
00226
00228     void addClassCode (const stdair::ClassCode_T& iClassCode) {
00229       _classCodeList.push_back (iClassCode);
00230     }
00231
00232   public:
00233     // ////////////////// Attributes /////////////////
00235     stdair::year_t _itYear;
00236     stdair::month_t _itMonth;
00237     stdair::day_t _itDay;
00238
00240     //long _itHours;
00241     stdair::hour_t _itHours;
00242     stdair::minute_t _itMinutes;
00243     stdair::second_t _itSeconds;
00244
00245   private:
00246     // ////////////////// Attributes /////////////////
00247
00249     YieldID_T _yieldId;
00250
```

```
00252     stdair::AirportCode_T _origin;
00253
00255     stdair::AirportCode_T _destination;
00256
00258     stdair::TripType_T _tripType;
00259
00261     stdair::Date_T _dateRangeStart;
00262
00264     stdair::Date_T _dateRangeEnd;
00265
00267     stdair::Duration_T _timeRangeStart;
00268
00270     stdair::Duration_T _timeRangeEnd;
00271
00273     stdair::YieldValue_T _yield;
00274
00276     stdair::CabinCode_T _cabinCode;
00277
00279     stdair::CityCode_T _pos;
00280
00282     stdair::ChannelLabel_T _channel;
00283
00285     stdair::AirlineCode_T _airlineCode;
00286
00288     stdair::ClassCode_T _classCode;
00289
00291     stdair::AirlineCodeList_T _airlineCodeList;
00292
00294     stdair::ClassList_StringList_T _classCodeList;
00295   };
00296
00297 }
00298 #endif // __AIRRAC_BOM_YIELDRULESTRUCT_HPP
```

25.17 airrac/command/YieldManager.cpp File Reference

```
#include <cassert>
#include <stdair/basic/BasConst_Request.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/TravelSolutionStruct.hpp>
#include <stdair/bom/AirportPair.hpp>
#include <stdair/bom/PosChannel.hpp>
#include <stdair/bom/DatePeriod.hpp>
#include <stdair/bom/TimePeriod.hpp>
#include <stdair/bom/YieldFeatures.hpp>
#include <stdair/bom/AirlineClassList.hpp>
#include <stdair/factory/FacBomManager.hpp>
#include <stdair/service/Logger.hpp>
#include <airrac/AIRRAC_Types.hpp>
#include <airrac/command/YieldManager.hpp>
```

Namespaces

- namespace AIRRAC

25.18 YieldManager.cpp

```
00001 // //////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // StdAir
00007 #include <stdair/basic/BasConst_Request.hpp>
00008 #include <stdair/bom/BomManager.hpp>
00009 #include <stdair/bom/BomRoot.hpp>
00010 #include <stdair/bom/Inventory.hpp>
00011 #include <stdair/bom/FlightDate.hpp>
00012 #include <stdair/bom/SegmentDate.hpp>
00013 #include <stdair/bom/SegmentCabin.hpp>
00014 #include <stdair/bom/FareFamily.hpp>
00015 #include <stdair/bom/BookingClass.hpp>
00016 #include <stdair/bom/TravelSolutionStruct.hpp>
00017 #include <stdair/bom/AirportPair.hpp>
00018 #include <stdair/bom/PosChannel.hpp>
00019 #include <stdair/bom/DatePeriod.hpp>
00020 #include <stdair/bom/TimePeriod.hpp>
00021 #include <stdair/bom/YieldFeatures.hpp>
00022 #include <stdair/bom/AirlineClassList.hpp>
00023 #include <stdair/factory/FacBomManager.hpp>
00024 #include <stdair/service/Logger.hpp>
00025 // Airrac
00026 #include <airrac/AIRRAC_Types.hpp>
00027 #include <airrac/command/YieldManager.hpp>
00028
00029 namespace AIRRAC {
00030
00031   // //////////////////////////////////////////////////////////////////////
00032   YieldManager::YieldManager() {
00033     assert (false);
00034   }
00035
00036   // //////////////////////////////////////////////////////////////////////
00037   YieldManager::YieldManager (const YieldManager&) {
00038     assert (false);
00039   }
00040
00041   // //////////////////////////////////////////////////////////////////////
00042   YieldManager::~YieldManager() {
00043   }
00044
00045   // //////////////////////////////////////////////////////////////////////
00046   void YieldManager::
00047   calculateYield (stdair::TravelSolutionList_T& ioTravelSolutionList,
00048                   const stdair::BomRoot& iBomRoot) {
00049
00050     // Browse the list of TravelSolution structures
00051     for (stdair::TravelSolutionList_T::iterator itTravelSolution =
00052            ioTravelSolutionList.begin();
00053          itTravelSolution != ioTravelSolutionList.end(); ++itTravelSolution) {
00054       stdair::TravelSolutionStruct& lTravelSolution = *itTravelSolution;
00055
00056       //
00057       YieldManager::calculateYield (lTravelSolution, iBomRoot);
00058     }
00059   }
00060
00061   // //////////////////////////////////////////////////////////////////////
00062   void YieldManager::
00063   calculateYield (stdair::TravelSolutionStruct& ioTravelSolution,
00064                   const stdair::BomRoot& iBomRoot) {
00065
```

```
00066     // Calculate/retrieve the yield for the given travel solution
00067     //YieldStore::calculateYield (ioYield, ioTravelSolution);
00068
00069     // TODO: update the statistical attributes of the yield.
00070   }
00071
00072   // ////////////////////////////////////////////////////////////////////
00073   void YieldManager::updateYields (const stdair::BomRoot& iBomRoot) {
00074     // Browse the list of booking classes and update yield for each one.
00075     const stdair::InventoryList_T lInvList =
00076       stdair::BomManager::getList<stdair::Inventory> (iBomRoot);
00077     for (stdair::InventoryList_T::const_iterator itInv = lInvList.begin();
00078          itInv != lInvList.end(); ++itInv) {
00079       const stdair::Inventory* lInv_ptr = *itInv;
00080       assert (lInv_ptr != NULL);
00081
00082       // Retrieve the airline code.
00083       const stdair::AirlineCode_T& lAirlineCode = lInv_ptr->getAirlineCode();
00084
00085       //
00086       const stdair::FlightDateList_T& lFDList =
00087         stdair::BomManager::getList<stdair::FlightDate> (*lInv_ptr);
00088       for (stdair::FlightDateList_T::const_iterator itFD = lFDList.begin();
00089            itFD != lFDList.end(); ++itFD) {
00090         const stdair::FlightDate* lFD_ptr = *itFD;
00091         assert (lFD_ptr != NULL);
00092
00093         //
00094         const stdair::SegmentDateList_T& lSDList =
00095           stdair::BomManager::getList<stdair::SegmentDate> (*lFD_ptr);
00096         for (stdair::SegmentDateList_T::const_iterator itSD = lSDList.begin();
00097              itSD != lSDList.end(); ++itSD) {
00098           const stdair::SegmentDate* lSD_ptr = *itSD;
00099           assert (lSD_ptr != NULL);
00100
00101           // Retrieve the origin and the destination
00102           const stdair::AirportCode_T& lOrigin = lSD_ptr->getBoardingPoint();
00103           const stdair::AirportCode_T& lDestination = lSD_ptr->getOffPoint();
00104
00105           // Retrieve the airport pair in the yield structure.
00106           const stdair::AirportPairKey lAirportPairKey (lOrigin, lDestination);
00107           stdair::AirportPair* lAirportPair_ptr = stdair::BomManager::
00108             getObjectPtr<stdair::AirportPair> (iBomRoot,
00109                                                lAirportPairKey.toString());
00110           if (lAirportPair_ptr == NULL) {
00111             STDAIR_LOG_ERROR ("Cannot find yield corresponding to the airport "
00112                               << "pair: " << lAirportPairKey.toString());
00113             assert (false);
00114           }
00115
00116           // Retrieve the boarding date and time
00117           const stdair::Date_T& lDate = lSD_ptr->getBoardingDate();
00118           const stdair::Duration_T& lTime = lSD_ptr->getBoardingTime();
00119
00120           // Retrieve the corresponding date period.
00121           const stdair::DatePeriodList_T& lDatePeriodList =
00122             stdair::BomManager::getList<stdair::DatePeriod> (*lAirportPair_ptr);
00123           for (stdair::DatePeriodList_T::const_iterator itDatePeriod =
00124                  lDatePeriodList.begin();
00125                itDatePeriod != lDatePeriodList.end(); ++itDatePeriod) {
00126             const stdair::DatePeriod* lDatePeriod_ptr = *itDatePeriod;
00127             assert (lDatePeriod_ptr != NULL);
00128
00129             const bool isDepartureDateValid =
00130               lDatePeriod_ptr->isDepartureDateValid (lDate);
00131
00132             if (isDepartureDateValid == true) {
```

```
00133
00134               // Retrieve the PoS-Channel.
00135               const stdair::PosChannelKey lPosChannelKey (stdair::DEFAULT_POS,
00136                                                           stdair::DEFAULT_CHANNEL
      );
00137               stdair::PosChannel* lPosChannel_ptr = stdair::BomManager::
00138                 getObjectPtr<stdair::PosChannel> (*lDatePeriod_ptr,
00139                                                   lPosChannelKey.toString());
00140               if (lPosChannel_ptr == NULL) {
00141                 STDAIR_LOG_ERROR ("Cannot find yield corresponding to the PoS-"
00142                                   << "Channel: " << lPosChannelKey.toString());
00143                 assert (false);
00144               }
00145
00146               // Retrieve the corresponding time period.
00147               const stdair::TimePeriodList_T& lTimePeriodList = stdair::
00148                 BomManager::getList<stdair::TimePeriod> (*lPosChannel_ptr);
00149               for (stdair::TimePeriodList_T::const_iterator itTimePeriod =
00150                      lTimePeriodList.begin();
00151                    itTimePeriod != lTimePeriodList.end(); ++itTimePeriod) {
00152                 const stdair::TimePeriod* lTimePeriod_ptr = *itTimePeriod;
00153                 assert (lTimePeriod_ptr != NULL);
00154
00155                 const bool isDepartureTimeValid =
00156                   lTimePeriod_ptr->isDepartureTimeValid (lTime);
00157
00158                 if (isDepartureTimeValid == true) {
00159                   updateYields (*lSD_ptr, *lTimePeriod_ptr, lAirlineCode);
00160                 }
00161               }
00162             }
00163           }
00164         }
00165       }
00166     }
00167   }
00168
00169   // ////////////////////////////////////////////////////////////////////
00170   void YieldManager::updateYields (const stdair::SegmentDate& iSegmentDate,
00171                                    const stdair::TimePeriod& iTimePeriod,
00172                                    const stdair::AirlineCode_T& iAirlineCode) {
00173     // Browse the segment-cabin list and retrieve the corresponding
00174     // yield features.
00175     const stdair::SegmentCabinList_T& lSegmentCabinList =
00176       stdair::BomManager::getList<stdair::SegmentCabin> (iSegmentDate);
00177     for (stdair::SegmentCabinList_T::const_iterator itSC =
00178            lSegmentCabinList.begin(); itSC != lSegmentCabinList.end(); ++itSC) {
00179       const stdair::SegmentCabin* lSegmentCabin_ptr = *itSC;
00180       assert (lSegmentCabin_ptr != NULL);
00181
00182       const stdair::CabinCode_T& lCabinCode = lSegmentCabin_ptr->getCabinCode();
00183       const stdair::TripType_T lTripType (stdair::TRIP_TYPE_ONE_WAY);
00184       const stdair::YieldFeaturesKey lYieldFeaturesKey (lTripType,
00185                                                         lCabinCode);
00186       const stdair::YieldFeatures* lYieldFeatures_ptr = stdair::BomManager::
00187         getObjectPtr<stdair::YieldFeatures> (iTimePeriod,
00188                                              lYieldFeaturesKey.toString());
00189       if (lYieldFeatures_ptr == NULL) {
00190         STDAIR_LOG_ERROR ("Cannot find the yield features corresponding to "
00191                           << iTimePeriod.describeKey() << ", "
00192                           << lCabinCode << " and " << lTripType);
00193         assert (false);
00194       }
00195
00196       // Browse the list of booking class and update the yield for each one.
00197       const stdair::FareFamilyList_T& lFFlist = stdair::BomManager::
00198         getList<stdair::FareFamily> (*lSegmentCabin_ptr);
```

```
00199       for (stdair::FareFamilyList_T::const_iterator itFF = lFFlist.begin();
00200            itFF != lFFlist.end(); ++itFF) {
00201         const stdair::FareFamily* lFF_ptr = *itFF;
00202         assert (lFF_ptr != NULL);
00203
00204         const stdair::BookingClassList_T& lBCList = stdair::BomManager::
00205           getList<stdair::BookingClass> (*lFF_ptr);
00206         for (stdair::BookingClassList_T::const_iterator itBC = lBCList.begin();
00207              itBC != lBCList.end(); ++itBC) {
00208           stdair::BookingClass* lBookingClass_ptr = *itBC;
00209           assert (lBookingClass_ptr != NULL);
00210
00211           const stdair::ClassCode_T& lClassCode =
00212             lBookingClass_ptr->getClassCode();
00213           stdair::AirlineCodeList_T lAirlineCodeList;
00214           lAirlineCodeList.push_back (iAirlineCode);
00215           stdair::ClassList_StringList_T lClassList;
00216           lClassList.push_back (lClassCode);
00217           const stdair::AirlineClassListKey lACListKey (lAirlineCodeList,
00218                                                         lClassList);
00219
00220           const stdair::AirlineClassList* lAirlineClassList_ptr = stdair::
00221             BomManager::getObjectPtr<stdair::AirlineClassList> (*lYieldFeatures_p
      tr, lACListKey.toString());
00222           if (lAirlineClassList_ptr != NULL) {
00223             const stdair::Yield_T& lYield = lAirlineClassList_ptr->getYield();
00224             lBookingClass_ptr->setYield (lYield);
00225
00226             //DEBUG
00227             STDAIR_LOG_DEBUG ("Update yield of " << lYield << " for "
00228                               << iAirlineCode << ", "
00229                               << iSegmentDate.describeKey() << ", "
00230                               << lBookingClass_ptr->describeKey());
00231           }
00232         }
00233       }
00234     }
00235   }
00236 }
```

25.19 airrac/command/YieldManager.hpp File Reference

```
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/TravelSolutionTypes.hpp>
```

Classes

- class AIRRAC::YieldManager

  Command wrapping the travel request process.

Namespaces

- namespace stdair

  Forward declarations.

- namespace AIRRAC

25.20 YieldManager.hpp

```
00001 #ifndef __AIRRAC_CMD_YIELDMANAGER_HPP
00002 #define __AIRRAC_CMD_YIELDMANAGER_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009 #include <stdair/bom/TravelSolutionTypes.hpp>
00010
00012 namespace stdair {
00013   class BomRoot;
00014   class SegmentDate;
00015   class TimePeriod;
00016 }
00017
00018 namespace AIRRAC {
00019
00023   class YieldManager {
00027     friend class AIRRAC_Service;
00028
00029   private:
00033     static void calculateYield (stdair::TravelSolutionList_T&,
00034                                 const stdair::BomRoot&);
00035
00039     static void calculateYield (stdair::TravelSolutionStruct&,
00040                                 const stdair::BomRoot&);
00041
00045     static void updateYields (const stdair::BomRoot&);
00046     static void updateYields (const stdair::SegmentDate&,
00047                               const stdair::TimePeriod&,
00048                               const stdair::AirlineCode_T&);
00049
00050   private:
00054     YieldManager();
00055
00059     YieldManager(const YieldManager&);
00060
00064     ~YieldManager();
00065   };
00066
00067 }
00068 #endif // __AIRRAC_CMD_YIELDMANAGER_HPP
```

25.21 airrac/command/YieldParser.cpp File Reference

```
#include <cassert>
#include <string>
#include <stdair/basic/BasFileMgr.hpp>
#include <airrac/command/YieldParserHelper.hpp>
#include <airrac/command/YieldParser.hpp>
```

Namespaces

- namespace AIRRAC

25.22 YieldParser.cpp

```
00001 // //////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <string>
00007 // StdAir
00008 #include <stdair/basic/BasFileMgr.hpp>
00009 // AirRAC
00010 #include <airrac/command/YieldParserHelper.hpp>
00011 #include <airrac/command/YieldParser.hpp>
00012
00013 namespace AIRRAC {
00014
00015   // //////////////////////////////////////////////////////////////////////
00016   void YieldParser::generateYieldStore (const YieldFilePath& iYieldFilename,
00017                                         stdair::BomRoot& ioBomRoot) {
00018
00019     const stdair::Filename_T lFilename = iYieldFilename.name();
00020
00021     // Check that the file path given as input corresponds to an actual file
00022     const bool doesExistAndIsReadable =
00023       stdair::BasFileMgr::doesExistAndIsReadable (lFilename);
00024     if (doesExistAndIsReadable == false) {
00025       STDAIR_LOG_ERROR ("The yield input file, '" << lFilename
00026                         << "', can not be retrieved on the file-system");
00027       throw YieldInputFileNotFoundException ("The yield file '" + lFilename
00028                                              + "' does not exist or can not "
00029                                              "be read");
00030     }
00031
00032     // Initialise the yield file parser.
00033     YieldFileParser lYieldParser (ioBomRoot, lFilename);
00034
00035     // Parse the CSV-formatted yield store input file, and generate the
00036     // corresponding Yield-related objects.
00037     lYieldParser.generateYieldStore();
00038   }
00039 }
```

25.23 airrac/command/YieldParser.hpp File Reference

```
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/command/CmdAbstract.hpp>
#include <airrac/AIRRAC_Types.hpp>
```

Classes

- class AIRRAC::YieldParser

  Class wrapping the parser entry point.

Namespaces

- namespace stdair

  Forward declarations.

- namespace AIRRAC

25.24 YieldParser.hpp

```
00001 #ifndef __AIRRAC_CMD_YIELDPARSER_HPP
00002 #define __AIRRAC_CMD_YIELDPARSER_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/stdair_basic_types.hpp>
00011 #include <stdair/command/CmdAbstract.hpp>
00012 //AirRAC
00013 #include <airrac/AIRRAC_Types.hpp>
00014
00016 namespace stdair {
00017   class BomRoot;
00018 }
00019
00020 namespace AIRRAC {
00021
00025   class YieldParser : public stdair::CmdAbstract {
00026   public:
00036     static void generateYieldStore (const YieldFilePath&, stdair::BomRoot&);
00037   };
00038 }
00039 #endif // __AIRRAC_CMD_YIELDPARSER_HPP
```

25.25 airrac/command/YieldParserHelper.cpp File Reference

```
#include <cassert>
#include <fstream>
#include <vector>
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/basic/BasConst_Request.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/basic/BasParserTypes.hpp>
#include <airrac/command/YieldParserHelper.hpp>
#include <airrac/command/YieldRuleGenerator.hpp>
```

Classes

- struct AIRRAC::YieldParserHelper::YieldRuleParser

Namespaces

- namespace AIRRAC
- namespace AIRRAC::YieldParserHelper

Variables

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

25.26 YieldParserHelper.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <fstream>
00007 #include <vector>
00008 // StdAir
00009 #include <stdair/basic/BasFileMgr.hpp>
00010 #include <stdair/basic/BasConst_Request.hpp>
00011 #include <stdair/bom/BomRoot.hpp>
00012 #include <stdair/service/Logger.hpp>
00013 //#define BOOST_SPIRIT_DEBUG
00014 #include <stdair/basic/BasParserTypes.hpp>
00015 // Airrac
00016 #include <airrac/command/YieldParserHelper.hpp>
00017 #include <airrac/command/YieldRuleGenerator.hpp>
00018
00019 namespace AIRRAC {
00020
00021   namespace YieldParserHelper {
00022
00023     // ////////////////////////////////////////////////////////////////////
00024     //  Semantic actions
00025     // ////////////////////////////////////////////////////////////////////
00026
00027     ParserSemanticAction::
00028     ParserSemanticAction (YieldRuleStruct& ioYieldRule)
00029       : _yieldRule (ioYieldRule) {
00030     }
00031
00032     // ////////////////////////////////////////////////////////////////////
00033     storeYieldId::
00034     storeYieldId (YieldRuleStruct& ioYieldRule)
00035       : ParserSemanticAction (ioYieldRule) {
00036     }
00037
00038     // ////////////////////////////////////////////////////////////////////
00039     void storeYieldId::operator() (unsigned int iYieldId,
00040                                    boost::spirit::qi::unused_type,
00041                                    boost::spirit::qi::unused_type) const {
00042       _yieldRule.setYieldID (iYieldId);
00043
00044       // DEBUG
00045       //STDAIR_LOG_DEBUG ( "Yield Id: " << _yieldRule.getYieldID ());
00046
00047       const stdair::AirlineCode_T lEmptyAirlineCode ("");
00048       _yieldRule.setAirlineCode(lEmptyAirlineCode);
00049       _yieldRule.clearAirlineCodeList();
00050       const stdair::ClassCode_T lEmptyClassCode ("");
00051       _yieldRule.setClassCode(lEmptyClassCode);
00052       _yieldRule.clearClassCodeList();
00053       _yieldRule._itSeconds = 0;
00054
00055     }
00056
00057     // ////////////////////////////////////////////////////////////////////
00058     storeOrigin ::
00059     storeOrigin (YieldRuleStruct& ioYieldRule)
00060       : ParserSemanticAction (ioYieldRule) {
00061     }
00062
00063     // ////////////////////////////////////////////////////////////////////
00064     void storeOrigin::operator() (std::vector<char> iChar,
00065                                   boost::spirit::qi::unused_type,
```

```
00066                                   boost::spirit::qi::unused_type) const {
00067       const stdair::AirportCode_T lOrigin (iChar.begin(), iChar.end());
00068       _yieldRule.setOrigin (lOrigin);
00069       // DEBUG
00070       //STDAIR_LOG_DEBUG ( "Origin: " << _yieldRule.getOrigin ());
00071     }
00072
00073     // ////////////////////////////////////////////////////////////////////
00074     storeDestination ::
00075     storeDestination (YieldRuleStruct& ioYieldRule)
00076       : ParserSemanticAction (ioYieldRule) {
00077     }
00078
00079     // ////////////////////////////////////////////////////////////////////
00080     void storeDestination::operator() (std::vector<char> iChar,
00081                                        boost::spirit::qi::unused_type,
00082                                        boost::spirit::qi::unused_type) const {
00083       const stdair::AirportCode_T lDestination (iChar.begin(), iChar.end());
00084       _yieldRule.setDestination (lDestination);
00085       // DEBUG
00086       //STDAIR_LOG_DEBUG ( "Destination: " << _yieldRule.getDestination ());
00087     }
00088
00089     // ////////////////////////////////////////////////////////////////////
00090     storeTripType ::
00091     storeTripType (YieldRuleStruct& ioYieldRule)
00092       : ParserSemanticAction (ioYieldRule) {
00093     }
00094
00095     // ////////////////////////////////////////////////////////////////////
00096     void storeTripType::operator() (std::vector<char> iChar,
00097                                     boost::spirit::qi::unused_type,
00098                                     boost::spirit::qi::unused_type) const {
00099       const stdair::TripType_T lTripType (iChar.begin(), iChar.end());
00100       if (lTripType == "OW" || lTripType == "RT") {
00101         _yieldRule.setTripType (lTripType);
00102       } else {
00103         // ERROR
00104         STDAIR_LOG_ERROR ("Invalid trip type  " << lTripType);
00105       }
00106       // DEBUG
00107       //STDAIR_LOG_DEBUG ("TripType: " << _yieldRule.getTripType ());
00108     }
00109
00110
00111     // ////////////////////////////////////////////////////////////////////
00112     storeDateRangeStart::
00113     storeDateRangeStart (YieldRuleStruct& ioYieldRule)
00114       : ParserSemanticAction (ioYieldRule) {
00115     }
00116
00117     // ////////////////////////////////////////////////////////////////////
00118     void storeDateRangeStart::operator() (boost::spirit::qi::unused_type,
00119                                           boost::spirit::qi::unused_type,
00120                                           boost::spirit::qi::unused_type) const {

00121       const stdair::Date_T& lDateStart = _yieldRule.calculateDate ();
00122       _yieldRule.setDateRangeStart (lDateStart);
00123       // DEBUG
00124       //STDAIR_LOG_DEBUG ("Date Range Start: "<< _yieldRule.getDateRangeStart ())
    ;
00125     }
00126
00127     // ////////////////////////////////////////////////////////////////////
00128     storeDateRangeEnd::
00129     storeDateRangeEnd(YieldRuleStruct& ioYieldRule)
00130       : ParserSemanticAction (ioYieldRule) {
```

```
00131     }
00132
00133     // ////////////////////////////////////////////////////////////////////
00134     void storeDateRangeEnd::operator() (boost::spirit::qi::unused_type,
00135                                         boost::spirit::qi::unused_type,
00136                                         boost::spirit::qi::unused_type) const {
00137       const stdair::Date_T& lDateEnd = _yieldRule.calculateDate ();
00138       // As a Boost date period (DatePeriod_T) defines the last day of
00139       // the period to be end-date - one day, we have to add one day to that
00140       // end date before.
00141       const stdair::DateOffset_T oneDay (1);
00142       const stdair::Date_T lBoostDateEnd = lDateEnd + oneDay;
00143       _yieldRule.setDateRangeEnd (lBoostDateEnd);
00144        // DEBUG
00145        //STDAIR_LOG_DEBUG ("Date Range End: " << _yieldRule.getDateRangeEnd ());
00146     }
00147
00148     // ////////////////////////////////////////////////////////////////////
00149     storeStartRangeTime::
00150     storeStartRangeTime (YieldRuleStruct& ioYieldRule)
00151       : ParserSemanticAction (ioYieldRule) {
00152     }
00153
00154     // ////////////////////////////////////////////////////////////////////
00155     void storeStartRangeTime::operator() (boost::spirit::qi::unused_type,
00156                                           boost::spirit::qi::unused_type,
00157                                           boost::spirit::qi::unused_type) const {

00158       const stdair::Duration_T& lTimeStart = _yieldRule.calculateTime ();
00159       _yieldRule.setTimeRangeStart (lTimeStart);
00160       // DEBUG
00161       //STDAIR_LOG_DEBUG ("Time Range Start: " << _yieldRule.getTimeRangeStart ()
      );
00162       // Reset the number of seconds
00163       _yieldRule._itSeconds = 0;
00164     }
00165
00166     // ////////////////////////////////////////////////////////////////////
00167     storeEndRangeTime::
00168     storeEndRangeTime (YieldRuleStruct& ioYieldRule)
00169       : ParserSemanticAction (ioYieldRule) {
00170     }
00171
00172     // ////////////////////////////////////////////////////////////////////
00173     void storeEndRangeTime::operator() (boost::spirit::qi::unused_type,
00174                                         boost::spirit::qi::unused_type,
00175                                         boost::spirit::qi::unused_type) const {
00176       const stdair::Duration_T& lTimeEnd = _yieldRule.calculateTime ();
00177       _yieldRule.setTimeRangeEnd (lTimeEnd);
00178       // DEBUG
00179       //STDAIR_LOG_DEBUG ("Time Range End: " << _yieldRule.getTimeRangeEnd ());
00180       // Reset the number of seconds
00181       _yieldRule._itSeconds = 0;
00182     }
00183
00184     // ////////////////////////////////////////////////////////////////////
00185     storePOS ::
00186     storePOS (YieldRuleStruct& ioYieldRule)
00187       : ParserSemanticAction (ioYieldRule) {
00188     }
00189
00190     // ////////////////////////////////////////////////////////////////////
00191     void storePOS::operator() (std::vector<char> iChar,
00192                                boost::spirit::qi::unused_type,
00193                                boost::spirit::qi::unused_type) const {
00194       const stdair::CityCode_T lPOS (iChar.begin(), iChar.end());
00195       if (lPOS == _yieldRule.getOrigin() || lPOS == _yieldRule.getDestination() )
```

```
    {
00196         _yieldRule.setPOS (lPOS);
00197       } else if (lPOS == "ROW") {
00198         const stdair::CityCode_T lPOSROW ("ROW");
00199         _yieldRule.setPOS (lPOSROW);
00200       } else if (lPOS == stdair::DEFAULT_POS) {
00201         _yieldRule.setPOS (stdair::DEFAULT_POS);
00202       } else {
00203         // ERROR
00204         STDAIR_LOG_ERROR ("Invalid point of sale " << lPOS);
00205       }
00206       // DEBUG
00207       //STDAIR_LOG_DEBUG ("POS: " << _yieldRule.getPOS ());
00208     }
00209
00210     // ////////////////////////////////////////////////////////////////////
00211     storeCabinCode ::
00212     storeCabinCode (YieldRuleStruct& ioYieldRule)
00213       : ParserSemanticAction (ioYieldRule) {
00214     }
00215
00216     // ////////////////////////////////////////////////////////////////////
00217     void storeCabinCode::operator() (char iChar,
00218                                      boost::spirit::qi::unused_type,
00219                                      boost::spirit::qi::unused_type) const {
00220       std::ostringstream ostr;
00221       ostr << iChar;
00222       const std::string& cabinCodeStr = ostr.str();
00223       const stdair::CabinCode_T lCabinCode (cabinCodeStr);
00224       _yieldRule.setCabinCode (lCabinCode);
00225
00226       // DEBUG
00227       //STDAIR_LOG_DEBUG ("Cabin Code: " << _yieldRule.getCabinCode ());

00228
00229     }
00230
00231     // ////////////////////////////////////////////////////////////////////
00232     storeChannel ::
00233     storeChannel (YieldRuleStruct& ioYieldRule)
00234       : ParserSemanticAction (ioYieldRule) {
00235     }
00236
00237     // ////////////////////////////////////////////////////////////////////
00238     void storeChannel::operator() (std::vector<char> iChar,
00239                                    boost::spirit::qi::unused_type,
00240                                    boost::spirit::qi::unused_type) const {
00241       const stdair::ChannelLabel_T lChannel (iChar.begin(), iChar.end());
00242       if (lChannel != "IN" && lChannel != "IF" && lChannel != "DN"
00243           && lChannel != "DF" && lChannel != stdair::DEFAULT_CHANNEL) {
00244         // ERROR
00245         STDAIR_LOG_ERROR ("Invalid channel " << lChannel);
00246       }
00247       _yieldRule.setChannel (lChannel);
00248       // DEBUG
00249       //STDAIR_LOG_DEBUG ("Channel: " << _yieldRule.getChannel ());
00250     }
00251
00252     // ////////////////////////////////////////////////////////////////////
00253     storeYield::
00254     storeYield (YieldRuleStruct& ioYieldRule)
00255       : ParserSemanticAction (ioYieldRule) {
00256     }
00257
00258     // ////////////////////////////////////////////////////////////////////
00259     void storeYield::operator() (double iYield,
00260                                  boost::spirit::qi::unused_type,
```

```
00261                                boost::spirit::qi::unused_type) const {
00262       const stdair::YieldValue_T lYield= iYield;
00263       _yieldRule.setYield (lYield);
00264       // DEBUG
00265       //STDAIR_LOG_DEBUG ("Yield: " << _yieldRule.getYield ());
00266     }
00267
00268     // ////////////////////////////////////////////////////////////////////
00269     storeAirlineCode ::
00270     storeAirlineCode (YieldRuleStruct& ioYieldRule)
00271       : ParserSemanticAction (ioYieldRule) {
00272     }
00273
00274     // ////////////////////////////////////////////////////////////////////
00275     void storeAirlineCode::operator() (std::vector<char> iChar,
00276                                        boost::spirit::qi::unused_type,
00277                                        boost::spirit::qi::unused_type) const {
00278
00279       const stdair::AirlineCode_T lAirlineCode (iChar.begin(), iChar.end());
00280       // Update the airline code
00281       _yieldRule.setAirlineCode (lAirlineCode);
00282       // Insertion of this airline Code list in the whole AirlineCode name
00283       _yieldRule.addAirlineCode (lAirlineCode);
00284       // DEBUG
00285       //STDAIR_LOG_DEBUG ( "Airline code: " << lAirlineCode);
00286     }
00287
00288     // ////////////////////////////////////////////////////////////////////
00289     storeClass ::
00290     storeClass (YieldRuleStruct& ioYieldRule)
00291       : ParserSemanticAction (ioYieldRule) {
00292     }
00293
00294     // ////////////////////////////////////////////////////////////////////
00295     void storeClass::operator() (std::vector<char> iChar,
00296                                  boost::spirit::qi::unused_type,
00297                                  boost::spirit::qi::unused_type) const {
00298       std::ostringstream ostr;
00299       for (std::vector<char>::const_iterator lItVector = iChar.begin();
00300          lItVector != iChar.end();
00301          lItVector++) {
00302        ostr << *lItVector;
00303       }
00304       const std::string& classCodeStr = ostr.str();
00305       const stdair::ClassCode_T lClassCode (classCodeStr);
00306       // Insertion of this class Code list in the whole classCode name
00307       _yieldRule.addClassCode (lClassCode);
00308       // DEBUG
00309       //STDAIR_LOG_DEBUG ("Class Code: " << classCodeStr);
00310     }
00311
00312     // ////////////////////////////////////////////////////////////////////
00313     doEndYield::
00314     doEndYield (stdair::BomRoot& ioBomRoot,
00315                 YieldRuleStruct& ioYieldRule)
00316       : ParserSemanticAction (ioYieldRule),
00317         _bomRoot (ioBomRoot) {
00318     }
00319
00320     // ////////////////////////////////////////////////////////////////////
00321     void doEndYield::operator() (boost::spirit::qi::unused_type,
00322                                  boost::spirit::qi::unused_type,
00323                                  boost::spirit::qi::unused_type) const {
00324       // DEBUG
00325       // STDAIR_LOG_DEBUG ("Do End");
00326       // Generation of the yield rule object.
00327       YieldRuleGenerator::createAirportPair (_bomRoot, _yieldRule);
```

```
00328       STDAIR_LOG_DEBUG (_yieldRule.describe());
00329     }
00330
00331     // ////////////////////////////////////////////////////////////////////
00332     //
00333     //  Utility Parsers
00334     //
00335     // ////////////////////////////////////////////////////////////////////
00337     namespace bsq = boost::spirit::qi;
00338     namespace bsa = boost::spirit::ascii;
00339
00341     stdair::int1_p_t int1_p;
00342
00344     stdair::uint2_p_t uint2_p;
00345
00347     stdair::uint4_p_t uint4_p;
00348
00350     stdair::uint1_4_p_t uint1_4_p;
00351
00353     stdair::hour_p_t hour_p;
00354     stdair::minute_p_t minute_p;
00355     stdair::second_p_t second_p;
00356
00358     stdair::year_p_t year_p;
00359     stdair::month_p_t month_p;
00360     stdair::day_p_t day_p;
00361
00363     //
00364     //  (Boost Spirit) Grammar Definition
00365     //
00367
00387     struct YieldRuleParser :
00388       public boost::spirit::qi::grammar<stdair::iterator_t,
00389                                         boost::spirit::ascii::space_type> {
00390
00391       YieldRuleParser (stdair::BomRoot& ioBomRoot,
00392                        YieldRuleStruct& ioYieldRule) :
00393       YieldRuleParser::base_type(start),
00394       _bomRoot(ioBomRoot), _yieldRule(ioYieldRule) {
00395
00396         start = *(comments | yield_rule);
00397
00398         comments = (bsq::lexeme[bsq::repeat(2)[bsa::char_('//')]
00399                                 >> +(bsa::char_ - bsq::eol)
00400                                 >> bsq::eol]
00401                     | bsq::lexeme[bsa::char_('//') >>bsa::char_('*')
00402                                   >> +(bsa::char_ - bsa::char_('*'))
00403                                   >> bsa::char_('*') >> bsa::char_('//')]);
00404
00405         yield_rule =  yield_id
00406           >> ';' >> origin >> ';' >> destination
00407           >> ';' >> tripType
00408           >> ';' >> dateRangeStart >> ';' >> dateRangeEnd
00409           >> ';' >> timeRangeStart >> ';' >> timeRangeEnd
00410           >> ';' >> point_of_sale >>  ';' >> cabinCode
00411           >> ';' >> channel >> ';' >> yield
00412           >> +( ';' >> segment )
00413           >> yield_rule_end[doEndYield(_bomRoot, _yieldRule)];
00414         ;
00415
00416         yield_id = uint1_4_p[storeYieldId(_yieldRule)];
00417
00418         origin = bsq::repeat(3)[bsa::char_("A-Z")][storeOrigin(_yieldRule)];
00419
00420         destination =
00421           bsq::repeat(3)[bsa::char_("A-Z")][storeDestination(_yieldRule)];
00422
```

```
00423         tripType =
00424           bsq::repeat(2)[bsa::char_("A-Z")][storeTripType(_yieldRule)];
00425
00426         dateRangeStart = date[storeDateRangeStart(_yieldRule)];
00427
00428         dateRangeEnd = date[storeDateRangeEnd(_yieldRule)];
00429
00430         date = bsq::lexeme
00431           [year_p[boost::phoenix::ref(_yieldRule._itYear) = bsq::labels::_1]
00432            >> '-'
00433            >> month_p[boost::phoenix::ref(_yieldRule._itMonth) = bsq::labels::_1]
00434            >> '-'
00435            >> day_p[boost::phoenix::ref(_yieldRule._itDay) = bsq::labels::_1] ];
00436
00437         timeRangeStart = time[storeStartRangeTime(_yieldRule)];
00438
00439         timeRangeEnd = time[storeEndRangeTime(_yieldRule)];
00440
00441         time =  bsq::lexeme
00442           [hour_p[boost::phoenix::ref(_yieldRule._itHours) = bsq::labels::_1]
00443            >> ':'
00444            >> minute_p[boost::phoenix::ref(_yieldRule._itMinutes) = bsq::labels::
      _1]
00445            >> - (':' >> second_p[boost::phoenix::ref(_yieldRule._itSeconds) = bsq
      ::labels::_1]) ];
00446
00447         point_of_sale = bsq::repeat(3)[bsa::char_("A-Z")][storePOS(_yieldRule)];
00448
00449         cabinCode = bsa::char_("A-Z")[storeCabinCode(_yieldRule)];
00450
00451         channel = bsq::repeat(2)[bsa::char_("A-Z")][storeChannel(_yieldRule)];
00452
00453         yield = bsq::double_[storeYield(_yieldRule)];
00454
00455         segment = bsq::repeat(2)[bsa::char_("A-Z")][storeAirlineCode(_yieldRule)]
00456           >> ';'
00457           >> bsq::repeat(1,bsq::inf)[bsa::char_("A-Z")][storeClass(_yieldRule)];
00458
00459         yield_rule_end = bsa::char_(';');
00460
00461         // BOOST_SPIRIT_DEBUG_NODE (YieldParser);
00462         BOOST_SPIRIT_DEBUG_NODE (start);
00463         BOOST_SPIRIT_DEBUG_NODE (comments);
00464         BOOST_SPIRIT_DEBUG_NODE (yield_rule);
00465         BOOST_SPIRIT_DEBUG_NODE (yield_id);
00466         BOOST_SPIRIT_DEBUG_NODE (origin);
00467         BOOST_SPIRIT_DEBUG_NODE (destination);
00468         BOOST_SPIRIT_DEBUG_NODE (tripType);
00469         BOOST_SPIRIT_DEBUG_NODE (dateRangeStart);
00470         BOOST_SPIRIT_DEBUG_NODE (dateRangeEnd);
00471         BOOST_SPIRIT_DEBUG_NODE (date);
00472         BOOST_SPIRIT_DEBUG_NODE (timeRangeStart);
00473         BOOST_SPIRIT_DEBUG_NODE (timeRangeEnd);
00474         BOOST_SPIRIT_DEBUG_NODE (time);
00475         BOOST_SPIRIT_DEBUG_NODE (point_of_sale);
00476         BOOST_SPIRIT_DEBUG_NODE (cabinCode);
00477         BOOST_SPIRIT_DEBUG_NODE (channel);
00478         BOOST_SPIRIT_DEBUG_NODE (yield);
00479         BOOST_SPIRIT_DEBUG_NODE (segment);
00480         BOOST_SPIRIT_DEBUG_NODE (yield_rule_end);
00481
00482       }
00483
00484       // Instantiation of rules
00485       boost::spirit::qi::rule<stdair::iterator_t,
```

```
00486                                boost::spirit::ascii::space_type>
00487       start, comments, yield_rule, yield_id, origin, destination, tripType,
00488         dateRangeStart, dateRangeEnd, date, timeRangeStart, timeRangeEnd,
00489         time, point_of_sale, cabinCode, channel, yield, segment,
00490         yield_rule_end;
00491
00492       // Parser Context
00493       stdair::BomRoot& _bomRoot;
00494       YieldRuleStruct& _yieldRule;
00495     };
00496
00497   }
00498
00499
00501   //
00502   //  Entry class for the file parser
00503   //
00505
00506   // ////////////////////////////////////////////////////////////////////
00507   YieldFileParser::YieldFileParser (stdair::BomRoot& ioBomRoot,
00508                                     const std::string& iFilename)
00509     : _filename (iFilename), _bomRoot (ioBomRoot) {
00510     init();
00511   }
00512
00513   // ////////////////////////////////////////////////////////////////////
00514   void YieldFileParser::init() {
00515
00516     // Check that the file exists and is readable
00517     const bool doesExistAndIsReadable =
00518       stdair::BasFileMgr::doesExistAndIsReadable (_filename);
00519
00520     if (doesExistAndIsReadable == false) {
00521       STDAIR_LOG_ERROR ("The yield schedule file " << _filename
00522                         << " does not exist or can not be  read.");
00523
00524       throw YieldInputFileNotFoundException ("The yield file " + _filename + " do
      es not exist or can not be read");
00525     }
00526   }
00527
00528   // ////////////////////////////////////////////////////////////////////
00529   void YieldFileParser::generateYieldStore () {
00530
00531     STDAIR_LOG_DEBUG ("Parsing yield input file: " << _filename);
00532
00533     // File to be parsed
00534     std::ifstream fileToBeParsed (_filename.c_str(), std::ios_base::in);
00535
00536     // Check the filename exists and can be open
00537     if (fileToBeParsed.is_open() == false) {
00538       STDAIR_LOG_ERROR ("The yield store file " << _filename
00539                         << " can not be open."
00540                         << std::endl);
00541
00542       throw YieldInputFileNotFoundException ("The file " + _filename
00543                                              + " does not exist or can not be rea
      d");
00544     }
00545
00546     // Create an input iterator
00547     stdair::base_iterator_t inputBegin (fileToBeParsed);
00548
00549     // Convert input iterator to an iterator usable by spirit parser
00550     stdair::iterator_t
00551       start (boost::spirit::make_default_multi_pass (inputBegin));
00552     stdair::iterator_t end;
```

```
00553
00554     // Initialise the parser (grammar) with the helper/staging structure.
00555     YieldParserHelper::YieldRuleParser lYParser(_bomRoot, _yieldRule);
00556
00557     // Launch the parsing of the file and, thanks to the doEndYield
00558     // call-back structure, the building of the whole BomRoot BOM
00559     const bool hasParsingBeenSuccesful =
00560        boost::spirit::qi::phrase_parse (start, end, lYParser,
00561                                         boost::spirit::ascii::space);
00562
00563     if (hasParsingBeenSuccesful == false) {
00564       // TODO: decide whether to throw an exceqption
00565       STDAIR_LOG_ERROR ("Parsing of yield input file: " << _filename
00566                         << " failed");
00567       throw YieldFileParsingFailedException ("Parsing of yield input file: "
00568                                              + _filename + " failed");
00569     }
00570     if  (start != end) {
00571       // TODO: decide whether to throw an exception
00572       STDAIR_LOG_ERROR ("Parsing of yield input file: " << _filename
00573                         << " failed");
00574       throw YieldFileParsingFailedException ("Parsing of yield input file: "
00575                                              + _filename + " failed");
00576     }
00577     if (hasParsingBeenSuccesful == true && start == end) {
00578       STDAIR_LOG_DEBUG ("Parsing of yield input file: " << _filename
00579       << " succeeded");
00580     }
00581
00582   }
00583
00584 }
```

25.27 airrac/command/YieldParserHelper.hpp File Reference

```
#include <string>
#include <boost/spirit/include/qi.hpp>
#include <stdair/command/CmdAbstract.hpp>
#include <airrac/AIRRAC_Types.hpp>
#include <airrac/bom/YieldRuleStruct.hpp>
```

Classes

- struct AIRRAC::YieldParserHelper::ParserSemanticAction
- struct AIRRAC::YieldParserHelper::storeYieldId
- struct AIRRAC::YieldParserHelper::storeOrigin
- struct AIRRAC::YieldParserHelper::storeDestination
- struct AIRRAC::YieldParserHelper::storeTripType
- struct AIRRAC::YieldParserHelper::storeDateRangeStart
- struct AIRRAC::YieldParserHelper::storeDateRangeEnd
- struct AIRRAC::YieldParserHelper::storeStartRangeTime
- struct AIRRAC::YieldParserHelper::storeEndRangeTime
- struct AIRRAC::YieldParserHelper::storePOS
- struct AIRRAC::YieldParserHelper::storeCabinCode
- struct AIRRAC::YieldParserHelper::storeChannel
- struct AIRRAC::YieldParserHelper::storeYield
- struct AIRRAC::YieldParserHelper::storeAirlineCode
- struct AIRRAC::YieldParserHelper::storeClass
- struct AIRRAC::YieldParserHelper::doEndYield
- class AIRRAC::YieldFileParser

Namespaces

- namespace stdair

  Forward declarations.

- namespace AIRRAC
- namespace AIRRAC::YieldParserHelper

25.28 YieldParserHelper.hpp

```
00001 #ifndef __AIRRAC_CMD_YIELDPARSERHELPER_HPP
00002 #define __AIRRAC_CMD_YIELDPARSERHELPER_HPP
00003
00004 // //////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // Boost
00010 #include <boost/spirit/include/qi.hpp>
00011 // StdAir
00012 #include <stdair/command/CmdAbstract.hpp>
00013 // Airrac
00014 #include <airrac/AIRRAC_Types.hpp>
00015 #include <airrac/bom/YieldRuleStruct.hpp>
00016
00017 // Forward declarations
00018 namespace stdair {
00019   class BomRoot;
00020 }
00021
00022 namespace AIRRAC {
00023
00024   namespace YieldParserHelper {
00025
00026     // //////////////////////////////////////////////////////////////////
00027     //  Semantic actions
00028     // //////////////////////////////////////////////////////////////////
00030     struct ParserSemanticAction {
00032       ParserSemanticAction (YieldRuleStruct&);
00034       YieldRuleStruct& _yieldRule;
00035     };
00036
00038     struct storeYieldId : public ParserSemanticAction {
00040       storeYieldId (YieldRuleStruct&);
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 (YieldRuleStruct&);
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 (YieldRuleStruct&);
00062       void operator() (std::vector<char>,
00063                        boost::spirit::qi::unused_type,
00064                        boost::spirit::qi::unused_type) const;
00065     };
00066
00068     struct storeTripType : public ParserSemanticAction {
00070       storeTripType (YieldRuleStruct&);
00072       void operator() (std::vector<char>,
00073                        boost::spirit::qi::unused_type,
00074                        boost::spirit::qi::unused_type) const;
00075     };
00076
00078     struct storeDateRangeStart : public ParserSemanticAction {
00080       storeDateRangeStart (YieldRuleStruct&);
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 storeDateRangeEnd : public ParserSemanticAction {
00090       storeDateRangeEnd (YieldRuleStruct&);
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 storeStartRangeTime : public ParserSemanticAction {
00100       storeStartRangeTime (YieldRuleStruct&);
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 storeEndRangeTime : public ParserSemanticAction {
00110       storeEndRangeTime (YieldRuleStruct&);
00112       void operator() (boost::spirit::qi::unused_type,
00113                        boost::spirit::qi::unused_type,
00114                        boost::spirit::qi::unused_type) const;
00115     };
00116
00118     struct storePOS : public ParserSemanticAction {
00120       storePOS (YieldRuleStruct&);
00122       void operator() (std::vector<char>,
00123                        boost::spirit::qi::unused_type,
00124                        boost::spirit::qi::unused_type) const;
00125     };
00126
00128     struct storeCabinCode : public ParserSemanticAction {
00130       storeCabinCode  (YieldRuleStruct&);
00132       void operator() (char,
00133                        boost::spirit::qi::unused_type,
00134                        boost::spirit::qi::unused_type) const;
00135     };
00136
00138     struct storeChannel : public ParserSemanticAction {
00140       storeChannel (YieldRuleStruct&);
00142       void operator() (std::vector<char>,
00143                        boost::spirit::qi::unused_type,
00144                        boost::spirit::qi::unused_type) const;
00145     };
00146
00148     struct storeYield : public ParserSemanticAction {
00150       storeYield (YieldRuleStruct&);
00152       void operator() (double,
00153                        boost::spirit::qi::unused_type,
00154                        boost::spirit::qi::unused_type) const;
00155     };
00156
00158     struct storeAirlineCode : public ParserSemanticAction {
00160       storeAirlineCode (YieldRuleStruct&);
00162       void operator() (std::vector<char>,
00163                        boost::spirit::qi::unused_type,
00164                        boost::spirit::qi::unused_type) const;
00165     };
00166
00168     struct storeClass : public ParserSemanticAction {
00170       storeClass (YieldRuleStruct&);
00172       void operator() (std::vector<char>,
00173                        boost::spirit::qi::unused_type,
00174                        boost::spirit::qi::unused_type) const;
00175     };
00176
00178     struct doEndYield : public ParserSemanticAction {
```

```
00180       doEndYield (stdair::BomRoot&, YieldRuleStruct&);
00182       void operator() (boost::spirit::qi::unused_type,
00183                        boost::spirit::qi::unused_type,
00184                        boost::spirit::qi::unused_type) const;
00186       stdair::BomRoot& _bomRoot;
00187     };
00188
00189   }
00190
00192   //
00193   //  Entry class for the file parser
00194   //
00196
00202   class YieldFileParser : public stdair::CmdAbstract {
00203   public:
00205     YieldFileParser (stdair::BomRoot&,
00206                      const stdair::Filename_T& iYieldInputFilename);
00207
00209     void generateYieldStore ();
00210
00211   private:
00213     void init();
00214
00215   private:
00216
00217     // Attributes
00219     stdair::Filename_T _filename;
00220
00222     stdair::BomRoot& _bomRoot;
00223
00225     YieldRuleStruct _yieldRule;
00226   };
00227
00228 }
00229 #endif // __AIRRAC_CMD_YIELDPARSERHELPER_HPP
```

25.29 airrac/command/YieldRuleGenerator.cpp File Reference

```
#include <cassert>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/AirportPair.hpp>
#include <stdair/bom/PosChannel.hpp>
#include <stdair/bom/DatePeriod.hpp>
#include <stdair/bom/TimePeriod.hpp>
#include <stdair/bom/YieldFeatures.hpp>
#include <stdair/bom/AirlineClassList.hpp>
#include <stdair/factory/FacBomManager.hpp>
#include <stdair/service/Logger.hpp>
#include <airrac/bom/YieldRuleStruct.hpp>
#include <airrac/command/YieldRuleGenerator.hpp>
```

Namespaces

- namespace AIRRAC

25.30 YieldRuleGenerator.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/AirportPair.hpp>
00010 #include <stdair/bom/PosChannel.hpp>
00011 #include <stdair/bom/DatePeriod.hpp>
00012 #include <stdair/bom/TimePeriod.hpp>
00013 #include <stdair/bom/YieldFeatures.hpp>
00014 #include <stdair/bom/AirlineClassList.hpp>
00015 #include <stdair/factory/FacBomManager.hpp>
00016 #include <stdair/service/Logger.hpp>
00017 // AirRAC
00018 #include <airrac/bom/YieldRuleStruct.hpp>
00019 #include <airrac/command/YieldRuleGenerator.hpp>
00020
00021 namespace AIRRAC {
00022
00023   // //////////////////////////////////////////////////////////////////////
00024   void YieldRuleGenerator::
00025   createAirportPair (stdair::BomRoot& ioBomRoot,
00026                      const YieldRuleStruct& iYieldRuleStruct) {
00027
00028     // Set the airport-pair primary key.
00029     const stdair::AirportCode_T& lBoardPoint = iYieldRuleStruct.getOrigin ();
00030     const stdair::AirportCode_T& lOffPoint = iYieldRuleStruct.getDestination ();
00031     const stdair::AirportPairKey lAirportPairKey (lBoardPoint, lOffPoint);
00032
00033     // Check that the airport-pair object is not already existing. If an
00034     // airport-pair object with the same key has not already been created,
00035     // create it and link it to the ioBomRoot object.
00036     stdair::AirportPair* lAirportPair_ptr = stdair::BomManager::
00037       getObjectPtr<stdair::AirportPair> (ioBomRoot, lAirportPairKey.toString());
00038     if (lAirportPair_ptr == NULL) {
00039       lAirportPair_ptr = &stdair::FacBom<stdair::AirportPair>::
00040         instance().create (lAirportPairKey);
00041       stdair::FacBomManager::addToListAndMap (ioBomRoot, *lAirportPair_ptr);
00042       stdair::FacBomManager::linkWithParent (ioBomRoot, *lAirportPair_ptr);
00043     }
00044     // Sanity check.
00045     assert (lAirportPair_ptr != NULL);
00046
00047     stdair::AirportPair& lAirportPair = *lAirportPair_ptr;
00048     // Generate the date-period object corresponding to the given
00049     // yieldRule.
00050     createDateRange (lAirportPair, iYieldRuleStruct);
00051
00052   }
00053
00054
00055   // //////////////////////////////////////////////////////////////////////
00056   void YieldRuleGenerator::
00057   createDateRange (stdair::AirportPair& iAirportPair,
00058                    const YieldRuleStruct& iYieldRuleStruct) {
00059
00060     // Create the yield date-period primary key.
00061     const stdair::Date_T& lDateRangeStart =
00062       iYieldRuleStruct.getDateRangeStart ();
00063     const stdair::Date_T& lDateRangeEnd =
00064       iYieldRuleStruct.getDateRangeEnd ();
00065     const stdair::DatePeriod_T lDatePeriod (lDateRangeStart, lDateRangeEnd);
```

```
00066     const stdair::DatePeriodKey lYieldDatePeriodKey (lDatePeriod);
00067
00068     // Check that the date-period object is not already existing.
00069     // If a date-period object with the same key has not already been
00070     // created, create it and link it to the airport-pair object.
00071     stdair::DatePeriod* lYieldDatePeriod_ptr = stdair::BomManager::
00072       getObjectPtr<stdair::DatePeriod> (iAirportPair,
00073                                         lYieldDatePeriodKey.toString());
00074     if (lYieldDatePeriod_ptr == NULL) {
00075       lYieldDatePeriod_ptr = &stdair::FacBom<stdair::DatePeriod>::
00076         instance().create (lYieldDatePeriodKey);
00077       stdair::FacBomManager::
00078         addToListAndMap (iAirportPair, *lYieldDatePeriod_ptr);
00079       stdair::FacBomManager::
00080         linkWithParent (iAirportPair, *lYieldDatePeriod_ptr);
00081     }
00082     // Sanity check.
00083     assert (lYieldDatePeriod_ptr != NULL);
00084
00085     stdair::DatePeriod& lDateRange = *lYieldDatePeriod_ptr;
00086     // Generate the point_of_sale-channel object corresponding to
00087     // the given yieldRule.
00088     createPOSChannel (lDateRange, iYieldRuleStruct);
00089
00090   }
00091
00092   // ////////////////////////////////////////////////////////////////////
00093   void YieldRuleGenerator::
00094   createPOSChannel (stdair::DatePeriod& iDatePeriod,
00095                     const YieldRuleStruct& iYieldRuleStruct) {
00096
00097     // Create the point-of-sale-channel primary key.
00098     const stdair::CityCode_T& lPoS = iYieldRuleStruct.getPOS ();
00099     const stdair::ChannelLabel_T& lChannel = iYieldRuleStruct.getChannel ();
00100     const stdair::PosChannelKey lYieldPosChannelKey (lPoS, lChannel);
00101
00102     // Check that the point_of_sale-channel object is not already existing.
00103     // If a point_of_sale-channel object with the same key has not already
00104     // been created, create it and link it to the date-period object.
00105     stdair::PosChannel* lYieldPosChannel_ptr = stdair::BomManager::
00106       getObjectPtr<stdair::PosChannel> (iDatePeriod,
00107                                         lYieldPosChannelKey.toString());
00108     if (lYieldPosChannel_ptr == NULL) {
00109       lYieldPosChannel_ptr = &stdair::FacBom<stdair::PosChannel>::
00110         instance().create (lYieldPosChannelKey);
00111       stdair::FacBomManager::
00112         addToListAndMap (iDatePeriod, *lYieldPosChannel_ptr);
00113       stdair::FacBomManager::
00114         linkWithParent (iDatePeriod, *lYieldPosChannel_ptr);
00115     }
00116     // Sanity check.
00117     assert (lYieldPosChannel_ptr != NULL);
00118
00119     stdair::PosChannel& lPosChannel = *lYieldPosChannel_ptr;
00120     // Generate the time-period object corresponding to the given
00121     // yieldRule.
00122     createTimeRange (lPosChannel, iYieldRuleStruct);
00123
00124   }
00125
00126   // ////////////////////////////////////////////////////////////////////
00127   void YieldRuleGenerator::
00128   createTimeRange (stdair::PosChannel& iPosChannel,
00129                    const YieldRuleStruct& iYieldRuleStruct) {
00130
00131     // Create the yield time-period primary key.
00132     const stdair::Time_T& lTimeRangeStart
```

```
00133       = iYieldRuleStruct.getTimeRangeStart ();
00134     const stdair::Time_T& lTimeRangeEnd
00135       = iYieldRuleStruct.getTimeRangeEnd ();
00136     const stdair::TimePeriodKey lYieldTimePeriodKey (lTimeRangeStart,
00137                                                      lTimeRangeEnd);
00138
00139     // Check that the time-period object is not already existing.
00140     // If a time-period object with the same key has not already been
00141     // created, create it and link it to the point_of_sale-channel object.
00142     stdair::TimePeriod* lYieldTimePeriod_ptr = stdair::BomManager::
00143       getObjectPtr<stdair::TimePeriod> (iPosChannel,
00144                                         lYieldTimePeriodKey.toString());
00145     if (lYieldTimePeriod_ptr == NULL) {
00146       lYieldTimePeriod_ptr = &stdair::FacBom<stdair::TimePeriod>::
00147         instance().create (lYieldTimePeriodKey);
00148       stdair::FacBomManager::
00149         addToListAndMap (iPosChannel, *lYieldTimePeriod_ptr);
00150       stdair::FacBomManager::
00151         linkWithParent (iPosChannel, *lYieldTimePeriod_ptr);
00152     }
00153     // Sanity check.
00154     assert (lYieldTimePeriod_ptr != NULL);
00155
00156     stdair::TimePeriod& lTimeRange = *lYieldTimePeriod_ptr;
00157     // Generate the yield-features object corresponding to the given
00158     // yieldRule.
00159     createYieldFeatures (lTimeRange, iYieldRuleStruct);
00160
00161   }
00162
00163   // ////////////////////////////////////////////////////////////////////
00164   void YieldRuleGenerator::
00165   createYieldFeatures (stdair::TimePeriod& iTimePeriod,
00166                        const YieldRuleStruct& iYieldRuleStruct) {
00167
00168     // Create the yield-features primary key.
00169     const stdair::TripType_T& lTripType = iYieldRuleStruct.getTripType ();
00170     stdair::CabinCode_T lCabinCode = iYieldRuleStruct.getCabinCode ();
00171     const stdair::YieldFeaturesKey lYieldFeaturesKey (lTripType, lCabinCode);
00172
00173     // Check that the yield features object is not already existing.
00174     // If a yield features object with the same key has not already been
00175     // created, create it and link it to the time-period object.
00176     stdair::YieldFeatures* lYieldFeatures_ptr = stdair::BomManager::
00177       getObjectPtr<stdair::YieldFeatures > (iTimePeriod,
00178                                             lYieldFeaturesKey.toString());
00179     if (lYieldFeatures_ptr == NULL) {
00180       lYieldFeatures_ptr = &stdair::FacBom<stdair::YieldFeatures>::
00181         instance().create (lYieldFeaturesKey);
00182       stdair::FacBomManager::
00183         addToListAndMap (iTimePeriod, *lYieldFeatures_ptr);
00184       stdair::FacBomManager::
00185         linkWithParent (iTimePeriod, *lYieldFeatures_ptr);
00186     }
00187     // Sanity check.
00188     assert (lYieldFeatures_ptr != NULL);
00189
00190     stdair::YieldFeatures& lYieldFeatures = *lYieldFeatures_ptr;
00191     // Generate the airline-class list object corresponding to the
00192     // given yieldRule
00193     createAirlineClassList (lYieldFeatures, iYieldRuleStruct);
00194
00195   }
00196
00197   // ////////////////////////////////////////////////////////////////////
00198   void YieldRuleGenerator::
00199   createAirlineClassList (stdair::YieldFeatures& iYieldFeatures,
```

```
00200                           const YieldRuleStruct& iYieldRuleStruct) {
00201
00202     // Create the AirlineClassList primary key.
00203     const unsigned int lAirlineListSize =
00204       iYieldRuleStruct.getAirlineListSize();
00205     const unsigned int lClassCodeListSize =
00206       iYieldRuleStruct.getClassCodeListSize();
00207     assert (lAirlineListSize == lClassCodeListSize);
00208     const stdair::AirlineClassListKey
00209       lAirlineClassListKey (iYieldRuleStruct.getAirlineList() ,
00210                             iYieldRuleStruct.getClassCodeList());
00211     const stdair::Yield_T& lYield = iYieldRuleStruct.getYield ();
00212
00213     // Create the airline class list object and link it to the yieldures
00214     // object.
00215     stdair::AirlineClassList* lAirlineClassList_ptr =
00216       &stdair::FacBom<stdair::AirlineClassList>::instance().
00217       create(lAirlineClassListKey);
00218     lAirlineClassList_ptr->setYield (lYield);
00219     stdair::FacBomManager::addToListAndMap (iYieldFeatures,
00220                                             *lAirlineClassList_ptr);
00221     stdair::FacBomManager::linkWithParent (iYieldFeatures,
00222                                            *lAirlineClassList_ptr);
00223   }
00224
00225 }
00226
```

25.31 airrac/command/YieldRuleGenerator.hpp File Reference

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

Classes

- class AIRRAC::YieldRuleGenerator

Namespaces

- namespace stdair

  Forward declarations.

- namespace AIRRAC
- namespace AIRRAC::YieldParserHelper

25.32 YieldRuleGenerator.hpp

```
00001 #ifndef __AIRRAC_CMD_YIELDRULEGENERATOR_HPP
00002 #define __AIRRAC_CMD_YIELDRULEGENERATOR_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // StdAir
00008 #include <stdair/command/CmdAbstract.hpp>
00009 // AirRAC
00010 #include <airrac/AIRRAC_Types.hpp>
00011
00012 namespace stdair {
00013   class BomRoot;
00014   class YieldRule;
00015   class AirportPair;
00016   class DatePeriod;
00017   class PosChannel;
00018   class TimePeriod;
00019   class YieldFeatures;
00020   class AirlineClassList;
00021 }
00022
00023 namespace AIRRAC {
00024
00025   // Forward declarations
00026   struct YieldRuleStruct;
00027   namespace YieldParserHelper {
00028     struct doEndYield;
00029   }
00030
00032   class YieldRuleGenerator : public stdair::CmdAbstract {
00033     // Only the following class may use methods of YieldGenerator.
00034     // Indeed, as those methods build the BOM, it is not good to expose
00035     // them public.
00036     friend class YieldFileParser;
00037     friend struct YieldParserHelper::doEndYield;
00038     friend class YieldParser;
00039
00040   private:
00041
00050     static void createAirportPair (stdair::BomRoot&,
00051                                    const YieldRuleStruct&);
00052
00061     static void createDateRange (stdair::AirportPair&,
00062                                  const YieldRuleStruct&);
00063
00072     static void createPOSChannel (stdair::DatePeriod&,
00073                                   const YieldRuleStruct&);
00074
00083     static void createTimeRange (stdair::PosChannel&,
00084                                  const YieldRuleStruct&);
00085
00094     static void createYieldFeatures (stdair::TimePeriod&,
00095                                      const YieldRuleStruct&);
00096
00105     static void createAirlineClassList (stdair::YieldFeatures&,
00106                                         const YieldRuleStruct&);
00107
00108
00109   };
00110
00111 }
00112 #endif // __AIRRAC_CMD_YIELDRULEGENERATOR_HPP
```

25.33 airrac/config/airrac-paths.hpp File Reference

Defines

- #define PACKAGE "airrac"
- #define PACKAGE_NAME "AIRRAC"
- #define PACKAGE_VERSION "1.00.0"
- #define PREFIXDIR "/usr"
- #define EXEC_PREFIX "/usr"
- #define BINDIR "/usr/bin"
- #define LIBDIR "/usr/lib64"
- #define LIBEXECDIR "/usr/libexec"
- #define SBINDIR "/usr/sbin"
- #define SYSCONFDIR "/usr/etc"
- #define INCLUDEDIR "/usr/include"
- #define DATAROOTDIR "/usr/share"
- #define DATADIR "/usr/share"
- #define DOCDIR "/usr/share/doc/airrac-1.00.0"
- #define MANDIR "/usr/share/man"
- #define INFODIR "/usr/share/info"
- #define HTMLDIR "/usr/share/doc/airrac-1.00.0/html"
- #define PDFDIR "/usr/share/doc/airrac-1.00.0/html"
- #define STDAIR_SAMPLE_DIR "/usr/share/stdair/samples"

25.33.1 Define Documentation

25.33.1.1 #define PACKAGE "airrac"

Definition at line 4 of file airrac-paths.hpp.

25.33.1.2 #define PACKAGE_NAME "AIRRAC"

Definition at line 5 of file airrac-paths.hpp.

Referenced by readConfiguration().

25.33.1.3 #define PACKAGE_VERSION "1.00.0"

Definition at line 6 of file airrac-paths.hpp.

Referenced by readConfiguration().

25.33.1.4 #define PREFIXDIR "/usr"

Definition at line 7 of file airrac-paths.hpp.

Referenced by readConfiguration().

25.33.1.5 #define EXEC_PREFIX "/usr"

Definition at line 8 of file airrac-paths.hpp.

25.33.1.6 #define BINDIR "/usr/bin"

Definition at line 9 of file airrac-paths.hpp.

25.33.1.7 #define LIBDIR "/usr/lib64"

Definition at line 10 of file airrac-paths.hpp.

25.33.1.8 #define LIBEXECDIR "/usr/libexec"

Definition at line 11 of file airrac-paths.hpp.

25.33.1.9 #define SBINDIR "/usr/sbin"

Definition at line 12 of file airrac-paths.hpp.

25.33.1.10 #define SYSCONFDIR "/usr/etc"

Definition at line 13 of file airrac-paths.hpp.

25.33.1.11 #define INCLUDEDIR "/usr/include"

Definition at line 14 of file airrac-paths.hpp.

25.33.1.12 #define DATAROOTDIR "/usr/share"

Definition at line 15 of file airrac-paths.hpp.

25.33.1.13 #define DATADIR "/usr/share"

Definition at line 16 of file airrac-paths.hpp.

25.33.1.14 #define DOCDIR "/usr/share/doc/airrac-1.00.0"

Definition at line 17 of file airrac-paths.hpp.

25.33.1.15 #define MANDIR "/usr/share/man"

Definition at line 18 of file airrac-paths.hpp.

25.33.1.16 #define INFODIR "/usr/share/info"

Definition at line 19 of file airrac-paths.hpp.

25.33.1.17 #define HTMLDIR "/usr/share/doc/airrac-1.00.0/html"

Definition at line 20 of file airrac-paths.hpp.

25.33.1.18 #define PDFDIR "/usr/share/doc/airrac-1.00.0/html"

Definition at line 21 of file airrac-paths.hpp.

25.33.1.19 #define STDAIR_SAMPLE_DIR "/usr/share/stdair/samples"

Definition at line 22 of file airrac-paths.hpp.

25.34 airrac-paths.hpp

```
00001 #ifndef __AIRRAC_PATHS_HPP__
00002 #define __AIRRAC_PATHS_HPP__
00003
00004 #define PACKAGE "airrac"
00005 #define PACKAGE_NAME "AIRRAC"
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/lib64"
00011 #define LIBEXECDIR "/usr/libexec"
00012 #define SBINDIR "/usr/sbin"
00013 #define SYSCONFDIR "/usr/etc"
00014 #define INCLUDEDIR "/usr/include"
00015 #define DATAROOTDIR "/usr/share"
00016 #define DATADIR "/usr/share"
00017 #define DOCDIR "/usr/share/doc/airrac-1.00.0"
00018 #define MANDIR "/usr/share/man"
00019 #define INFODIR "/usr/share/info"
00020 #define HTMLDIR "/usr/share/doc/airrac-1.00.0/html"
00021 #define PDFDIR "/usr/share/doc/airrac-1.00.0/html"
00022 #define STDAIR_SAMPLE_DIR "/usr/share/stdair/samples"
00023
00024 #endif // __AIRRAC_PATHS_HPP__
```

25.35 airrac/factory/FacAirracServiceContext.cpp File Reference

```
#include <cassert>
#include <stdair/service/FacSupervisor.hpp>
#include <airrac/factory/FacAirracServiceContext.hpp>
#include <airrac/service/AIRRAC_ServiceContext.hpp>
```

Namespaces

- namespace AIRRAC

25.36 FacAirracServiceContext.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // StdAir
00007 #include <stdair/service/FacSupervisor.hpp>
00008 // AIRRAC Common
00009 #include <airrac/factory/FacAirracServiceContext.hpp>
00010 #include <airrac/service/AIRRAC_ServiceContext.hpp>
00011
00012 namespace AIRRAC {
00013
00014   FacAirracServiceContext* FacAirracServiceContext::_instance = NULL;
00015
00016   // ////////////////////////////////////////////////////////////////////
00017   FacAirracServiceContext::~FacAirracServiceContext() {
00018     _instance = NULL;
00019   }
00020
00021   // ////////////////////////////////////////////////////////////////////
00022   FacAirracServiceContext& FacAirracServiceContext::instance() {
00023
00024     if (_instance == NULL) {
00025       _instance = new FacAirracServiceContext();
00026       assert (_instance != NULL);
00027
00028       stdair::FacSupervisor::instance().registerServiceFactory (_instance);
00029     }
00030     return *_instance;
00031   }
00032
00033   // ////////////////////////////////////////////////////////////////////
00034   AIRRAC_ServiceContext& FacAirracServiceContext::create() {
00035     AIRRAC_ServiceContext* aServiceContext_ptr = NULL;
00036
00037     aServiceContext_ptr = new AIRRAC_ServiceContext();
00038     assert (aServiceContext_ptr != NULL);
00039
00040     // The new object is added to the Bom pool
00041     _pool.push_back (aServiceContext_ptr);
00042
00043     return *aServiceContext_ptr;
00044   }
00045
00046 }
```

25.37 airrac/factory/FacAirracServiceContext.hpp File Reference

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

Classes

- class AIRRAC::FacAirracServiceContext

  Factory for the service context.

Namespaces

- namespace AIRRAC

25.38 FacAirracServiceContext.hpp

```
00001 #ifndef __AIRRAC_FAC_FACAIRRACSERVICECONTEXT_HPP
00002 #define __AIRRAC_FAC_FACAIRRACSERVICECONTEXT_HPP
00003 // ////////////////////////////////////////////////////////////////////
00004 // Import section
00005 // ////////////////////////////////////////////////////////////////////
00006 // STL
00007 #include <string>
00008 // StdAir
00009 #include <stdair/stdair_basic_types.hpp>
00010 #include <stdair/service/FacServiceAbstract.hpp>
00011
00012 namespace AIRRAC {
00013
00015   class AIRRAC_ServiceContext;
00016
00017
00021   class FacAirracServiceContext : public stdair::FacServiceAbstract {
00022   public:
00023
00030     static FacAirracServiceContext& instance();
00031
00038     ~FacAirracServiceContext();
00039
00047     AIRRAC_ServiceContext& create();
00048
00049
00050   protected:
00056     FacAirracServiceContext() {}
00057
00058
00059   private:
00063     static FacAirracServiceContext* _instance;
00064   };
00065
00066 }
00067 #endif // __AIRRAC_FAC_FACAIRRACSERVICECONTEXT_HPP
```

25.39 airrac/service/AIRRAC_Service.cpp File Reference

```
#include <cassert>
#include <boost/make_shared.hpp>
#include <stdair/basic/BasChronometer.hpp>
#include <stdair/bom/BomDisplay.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/STDAIR_Service.hpp>
#include <airrac/basic/BasConst_AIRRAC_Service.hpp>
#include <airrac/factory/FacAirracServiceContext.hpp>
#include <airrac/command/YieldParser.hpp>
#include <airrac/command/YieldManager.hpp>
#include <airrac/service/AIRRAC_ServiceContext.hpp>
#include <airrac/AIRRAC_Service.hpp>
```

Namespaces

- namespace AIRRAC

25.40 AIRRAC_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/basic/BasChronometer.hpp>
00010 #include <stdair/bom/BomDisplay.hpp>
00011 #include <stdair/service/Logger.hpp>
00012 #include <stdair/STDAIR_Service.hpp>
00013 // Airrac
00014 #include <airrac/basic/BasConst_AIRRAC_Service.hpp>
00015 #include <airrac/factory/FacAirracServiceContext.hpp>
00016 #include <airrac/command/YieldParser.hpp>
00017 #include <airrac/command/YieldManager.hpp>
00018 #include <airrac/service/AIRRAC_ServiceContext.hpp>
00019 #include <airrac/AIRRAC_Service.hpp>
00020
00021 namespace AIRRAC {
00022
00023   // ////////////////////////////////////////////////////////////////////
00024   AIRRAC_Service::AIRRAC_Service() : _airracServiceContext (NULL) {
00025     assert (false);
00026   }
00027
00028   // ////////////////////////////////////////////////////////////////////
00029   AIRRAC_Service::AIRRAC_Service (const AIRRAC_Service& iService) {
00030     assert (false);
00031   }
00032
00033   // ////////////////////////////////////////////////////////////////////
00034   AIRRAC_Service::AIRRAC_Service (const stdair::BasLogParams& iLogParams)
00035     : _airracServiceContext (NULL) {
00036
00037     // Initialise the STDAIR service handler
00038     stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00039       initStdAirService (iLogParams);
00040
00041     // Initialise the service context
00042     initServiceContext();
00043
00044     // Add the StdAir service context to the AIRRAC service context
00045     // \note AIRRAC owns the STDAIR service resources here.
00046     const bool ownStdairService = true;
00047     addStdAirService (lSTDAIR_Service_ptr, ownStdairService);
00048
00049     // Initialise the (remaining of the) context
00050     initAirracService();
00051   }
00052
00053   // ////////////////////////////////////////////////////////////////////
00054   AIRRAC_Service::AIRRAC_Service (const stdair::BasLogParams& iLogParams,
00055                                   const stdair::BasDBParams& iDBParams)
00056     : _airracServiceContext (NULL) {
00057
00058     // Initialise the STDAIR service handler
00059     stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00060       initStdAirService (iLogParams, iDBParams);
00061
00062     // Initialise the service context
00063     initServiceContext();
00064
00065     // Add the StdAir service context to the AIRRAC service context
```

```
00066     // \note AIRRAC owns the STDAIR service resources here.
00067     const bool ownStdairService = true;
00068     addStdAirService (lSTDAIR_Service_ptr, ownStdairService);
00069
00070     // Initialise the (remaining of the) context
00071     initAirracService();
00072   }
00073
00074   // ////////////////////////////////////////////////////////////////////
00075   AIRRAC_Service::
00076   AIRRAC_Service (stdair::STDAIR_ServicePtr_T ioSTDAIR_Service_ptr)
00077     : _airracServiceContext (NULL) {
00078
00079     // Initialise the service context
00080     initServiceContext();
00081
00082     // Store the STDAIR service object within the (AIRRAC) service context
00083     // \note Airrac does not own the STDAIR service resources here.
00084     const bool doesNotOwnStdairService = false;
00085     addStdAirService (ioSTDAIR_Service_ptr, doesNotOwnStdairService);
00086
00087     // Initialise the context
00088     initAirracService();
00089   }
00090
00091   // ////////////////////////////////////////////////////////////////////
00092   AIRRAC_Service::~AIRRAC_Service() {
00093     // Delete/Clean all the objects from memory
00094     finalise();
00095   }
00096
00097   // ////////////////////////////////////////////////////////////////////
00098   void AIRRAC_Service::finalise() {
00099     assert (_airracServiceContext != NULL);
00100     // Reset the (Boost.)Smart pointer pointing on the STDAIR_Service object.
00101     _airracServiceContext->reset();
00102   }
00103
00104   // ////////////////////////////////////////////////////////////////////
00105   void AIRRAC_Service::initServiceContext() {
00106     // Initialise the service context
00107     AIRRAC_ServiceContext& lAIRRAC_ServiceContext =
00108       FacAirracServiceContext::instance().create();
00109     _airracServiceContext = &lAIRRAC_ServiceContext;
00110   }
00111
00112   // ////////////////////////////////////////////////////////////////////
00113   stdair::STDAIR_ServicePtr_T AIRRAC_Service::
00114   initStdAirService (const stdair::BasLogParams& iLogParams,
00115                      const stdair::BasDBParams& iDBParams) {
00116
00124     stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00125       boost::make_shared<stdair::STDAIR_Service> (iLogParams, iDBParams);
00126
00127     return lSTDAIR_Service_ptr;
00128
00129   }
00130
00131   // ////////////////////////////////////////////////////////////////////
00132   stdair::STDAIR_ServicePtr_T AIRRAC_Service::
00133   initStdAirService (const stdair::BasLogParams& iLogParams) {
00134
00142     stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00143       boost::make_shared<stdair::STDAIR_Service> (iLogParams);
00144
00145     return lSTDAIR_Service_ptr;
00146   }
```

```
00147
00148   // ////////////////////////////////////////////////////////////////////
00149   void AIRRAC_Service::
00150   addStdAirService (stdair::STDAIR_ServicePtr_T ioSTDAIR_Service_ptr,
00151                     const bool iOwnStdairService) {
00152
00153     // Retrieve the Airrac service context
00154     assert (_airracServiceContext != NULL);
00155     AIRRAC_ServiceContext& lAIRRAC_ServiceContext = *_airracServiceContext;
00156
00157     // Store the STDAIR service object within the (AIRRAC) service context
00158     lAIRRAC_ServiceContext.setSTDAIR_Service (ioSTDAIR_Service_ptr,
00159                                               iOwnStdairService);
00160   }
00161
00162   // ////////////////////////////////////////////////////////////////////
00163   void AIRRAC_Service::initAirracService() {
00164     // Do nothing at this stage. A sample BOM tree may be built by
00165     // calling the buildSampleBom() method
00166   }
00167
00168   // ////////////////////////////////////////////////////////////////////
00169   void AIRRAC_Service::
00170   parseAndLoad (const YieldFilePath& iYieldFilename) {
00171
00172     // Retrieve the AirRAC service context
00173     if (_airracServiceContext == NULL) {
00174       throw stdair::NonInitialisedServiceException ("The AirRAC service has not"
00175                                                     " been initialised");
00176     }
00177     assert (_airracServiceContext != NULL);
00178
00179     // Retrieve the AirRAC service context and whether it owns the Stdair
00180     // service
00181     AIRRAC_ServiceContext& lAIRRAC_ServiceContext = *_airracServiceContext;
00182     const bool doesOwnStdairService =
00183       lAIRRAC_ServiceContext.getOwnStdairServiceFlag();
00184
00185     // Retrieve the StdAir service object from the (AirRAC) service context
00186     stdair::STDAIR_Service& lSTDAIR_Service =
00187       lAIRRAC_ServiceContext.getSTDAIR_Service();
00188
00189     // Retrieve the BOM root object.
00190     stdair::BomRoot& lPersistentBomRoot =
00191       lSTDAIR_Service.getPersistentBomRoot();
00192
00196     YieldParser::generateYieldStore (iYieldFilename, lPersistentBomRoot);
00197
00210     buildComplementaryLinks (lPersistentBomRoot);
00211
00216     if (doesOwnStdairService == true) {
00217       //
00218       clonePersistentBom ();
00219     }
00220   }
00221
00222   // ////////////////////////////////////////////////////////////////////
00223   void AIRRAC_Service::buildSampleBom() {
00224
00225     // Retrieve the AirRAC service context
00226     if (_airracServiceContext == NULL) {
00227       throw stdair::NonInitialisedServiceException ("The AirRAC service has not"
00228                                                     " been initialised");
00229     }
00230     assert (_airracServiceContext != NULL);
00231
00232     // Retrieve the AirRAC service context and whether it owns the Stdair
```

```
00233     // service
00234     AIRRAC_ServiceContext& lAIRRAC_ServiceContext = *_airracServiceContext;
00235     const bool doesOwnStdairService =
00236       lAIRRAC_ServiceContext.getOwnStdairServiceFlag();
00237
00238     // Retrieve the StdAir service object from the (AirRAC) service context
00239     stdair::STDAIR_Service& lSTDAIR_Service =
00240       lAIRRAC_ServiceContext.getSTDAIR_Service();
00241
00242     // Retrieve the persistent BOM root object.
00243     stdair::BomRoot& lPersistentBomRoot =
00244       lSTDAIR_Service.getPersistentBomRoot();
00245
00250     if (doesOwnStdairService == true) {
00251       //
00252       lSTDAIR_Service.buildSampleBom();
00253     }
00254
00267     buildComplementaryLinks (lPersistentBomRoot);
00268
00273     if (doesOwnStdairService == true) {
00274       //
00275       clonePersistentBom ();
00276     }
00277   }
00278
00279   // ////////////////////////////////////////////////////////////////////
00280   void AIRRAC_Service::clonePersistentBom () {
00281
00282     // Retrieve the AirRAC service context
00283     if (_airracServiceContext == NULL) {
00284       throw stdair::NonInitialisedServiceException ("The AirRAC service has not"
00285                                                     " been initialised");
00286     }
00287     assert (_airracServiceContext != NULL);
00288
00289     // Retrieve the AirRAC service context and whether it owns the Stdair
00290     // service
00291     AIRRAC_ServiceContext& lAIRRAC_ServiceContext = *_airracServiceContext;
00292     const bool doesOwnStdairService =
00293       lAIRRAC_ServiceContext.getOwnStdairServiceFlag();
00294
00295     // Retrieve the StdAir service object from the (AirRAC) service context
00296     stdair::STDAIR_Service& lSTDAIR_Service =
00297       lAIRRAC_ServiceContext.getSTDAIR_Service();
00298
00303     if (doesOwnStdairService == true) {
00304
00305       //
00306       lSTDAIR_Service.clonePersistentBom ();
00307     }
00308
00321     stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
00322     buildComplementaryLinks (lBomRoot);
00323   }
00324
00325   // ////////////////////////////////////////////////////////////////////
00326   void AIRRAC_Service::buildComplementaryLinks (stdair::BomRoot& ioBomRoot) {
00327     // Currently, no more things to do by AirRAC at that stage.
00328   }
00329
00330   // ////////////////////////////////////////////////////////////////////
00331   void AIRRAC_Service::
00332   buildSampleTravelSolutions(stdair::TravelSolutionList_T& ioTravelSolutionList){

00333
00334     // Retrieve the AIRRAC service context
```

```
00335     if (_airracServiceContext == NULL) {
00336       throw stdair::NonInitialisedServiceException ("The AirRAC service has not"
00337                                                     " been initialised");
00338     }
00339     assert (_airracServiceContext != NULL);
00340
00341     AIRRAC_ServiceContext& lAIRRAC_ServiceContext = *_airracServiceContext;
00342
00343     // Retrieve the STDAIR service object from the (AirRAC) service context
00344     stdair::STDAIR_Service& lSTDAIR_Service =
00345       lAIRRAC_ServiceContext.getSTDAIR_Service();
00346
00347     // Delegate the BOM building to the dedicated service
00348     lSTDAIR_Service.buildSampleTravelSolutions (ioTravelSolutionList);
00349   }
00350
00351   // ////////////////////////////////////////////////////////////////////
00352   std::string AIRRAC_Service::csvDisplay() const {
00353
00354     // Retrieve the AIRRAC service context
00355     if (_airracServiceContext == NULL) {
00356       throw stdair::NonInitialisedServiceException ("The Airrac service "
00357                                                     "has not been initialised");
00358     }
00359     assert (_airracServiceContext != NULL);
00360
00361     AIRRAC_ServiceContext& lAIRRAC_ServiceContext = *_airracServiceContext;
00362
00363     // Retrieve the STDAIR service object from the (Airrac) service context
00364     stdair::STDAIR_Service& lSTDAIR_Service =
00365       lAIRRAC_ServiceContext.getSTDAIR_Service();
00366
00367     // Get the root of the BOM tree, on which all of the other BOM objects
00368     // are attached
00369     stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
00370
00371     // Delegate the BOM display to the dedicated service
00372     std::ostringstream oCSVStr;
00373     stdair::BomDisplay::csvSimFQTAirRACDisplay (oCSVStr, lBomRoot);
00374     return oCSVStr.str();
00375
00376   }
00377
00378   // ////////////////////////////////////////////////////////////////////
00379   std::string AIRRAC_Service::
00380   csvDisplay (const stdair::TravelSolutionList_T& ioTravelSolutionList) const {
00381
00382     // Retrieve the AirRAC service context
00383     if (_airracServiceContext == NULL) {
00384       throw stdair::NonInitialisedServiceException ("The AirRAC service has not"
00385                                                     " been initialised");
00386     }
00387     assert (_airracServiceContext != NULL);
00388
00389     // Retrieve the AirRAC service context
00390     AIRRAC_ServiceContext& lAIRRAC_ServiceContext = *_airracServiceContext;
00391
00392     // Retrieve the STDAIR service object from the (AirRAC) service context
00393     stdair::STDAIR_Service& lSTDAIR_Service =
00394       lAIRRAC_ServiceContext.getSTDAIR_Service();
00395
00396     // Delegate the BOM building to the dedicated service
00397     return lSTDAIR_Service.csvDisplay (ioTravelSolutionList);
00398   }
00399
00400   // ////////////////////////////////////////////////////////////////////
00401   void AIRRAC_Service::
```

```
00402   calculateYields (stdair::TravelSolutionList_T& ioTravelSolutionList) {
00403
00404     // Retrieve the Airrac service context
00405     if (_airracServiceContext == NULL) {
00406       throw stdair::NonInitialisedServiceException ("The AirRAC service has not"
00407                                                     " been initialised");
00408     }
00409     assert (_airracServiceContext != NULL);
00410     AIRRAC_ServiceContext& lAIRRAC_ServiceContext = *_airracServiceContext;
00411
00412     // Retrieve the StdAir service context
00413     stdair::STDAIR_Service& lSTDAIR_Service =
00414       lAIRRAC_ServiceContext.getSTDAIR_Service();
00415
00416     // Get the root of the BOM tree, on which all of the other BOM objects
00417     // will be attached
00418     stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
00419
00420     // Delegate the booking to the dedicated command: set the yields
00421     // for each travel solution of the given list
00422     stdair::BasChronometer lYieldChronometer;
00423     lYieldChronometer.start();
00424     YieldManager::calculateYield (ioTravelSolutionList, lBomRoot);
00425     const double lYieldMeasure = lYieldChronometer.elapsed();
00426
00427     // DEBUG
00428     STDAIR_LOG_DEBUG ("Yield calculation: " << lYieldMeasure << " - "
00429                       << lAIRRAC_ServiceContext.display());
00430   }
00431
00432   // ////////////////////////////////////////////////////////////////////
00433   void AIRRAC_Service::updateYields (stdair::BomRoot& ioBomRoot) {
00434     // Retrieve the AirRAC service context
00435     assert (_airracServiceContext != NULL);
00436
00437     // Update the default yields to the booking classes.
00438     YieldManager::updateYields (ioBomRoot);
00439   }
00440 }
```

25.41 airrac/service/AIRRAC_ServiceContext.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <airrac/basic/BasConst_AIRRAC_Service.hpp>
#include <airrac/service/AIRRAC_ServiceContext.hpp>
```

Namespaces

- namespace AIRRAC

25.42 AIRRAC_ServiceContext.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Airrac
00008 #include <airrac/basic/BasConst_AIRRAC_Service.hpp>
00009 #include <airrac/service/AIRRAC_ServiceContext.hpp>
00010
00011 namespace AIRRAC {
00012
00013   // ////////////////////////////////////////////////////////////////////
00014   AIRRAC_ServiceContext::AIRRAC_ServiceContext() : _ownStdairService (false) {
00015   }
00016
00017   // ////////////////////////////////////////////////////////////////////
00018   AIRRAC_ServiceContext::AIRRAC_ServiceContext (const AIRRAC_ServiceContext&)
00019     : _ownStdairService (false) {
00020     assert (false);
00021   }
00022
00023   // ////////////////////////////////////////////////////////////////////
00024   AIRRAC_ServiceContext::~AIRRAC_ServiceContext() {
00025   }
00026
00027   // ////////////////////////////////////////////////////////////////////
00028   const std::string AIRRAC_ServiceContext::shortDisplay() const {
00029     std::ostringstream oStr;
00030     oStr << "AIRRAC_ServiceContext -- Owns StdAir service: "
00031          << _ownStdairService;
00032     return oStr.str();
00033   }
00034
00035   // ////////////////////////////////////////////////////////////////////
00036   const std::string AIRRAC_ServiceContext::display() const {
00037     std::ostringstream oStr;
00038     oStr << shortDisplay();
00039     return oStr.str();
00040   }
00041   // ////////////////////////////////////////////////////////////////////
00042   const std::string AIRRAC_ServiceContext::describe() const {
00043     return shortDisplay();
00044   }
00045
00046   // ////////////////////////////////////////////////////////////////////
00047   void AIRRAC_ServiceContext::reset() {
00048
00049     // The shared_ptr<>::reset() method drops the refcount by one.
00050     // If the count result is dropping to zero, the resource pointed to
00051     // by the shared_ptr<> will be freed.
00052
00053     // Reset the stdair shared pointer
00054     _stdairService.reset();
00055   }
00056
00057 }
```

25.43 airrac/service/AIRRAC_ServiceContext.hpp File Reference

```
#include <string>
#include <stdair/stdair_service_types.hpp>
#include <stdair/service/ServiceAbstract.hpp>
#include <airrac/AIRRAC_Types.hpp>
```

Classes

- class AIRRAC::AIRRAC_ServiceContext

  Inner class holding the context for the AIRRAC Service object.

Namespaces

- namespace stdair

  Forward declarations.

- namespace AIRRAC

25.44 AIRRAC_ServiceContext.hpp

```
00001 #ifndef __AIRRAC_SVC_AIRRACSERVICECONTEXT_HPP
00002 #define __AIRRAC_SVC_AIRRACSERVICECONTEXT_HPP
00003
00004 // //////////////////////////////////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/stdair_service_types.hpp>
00011 #include <stdair/service/ServiceAbstract.hpp>
00012 // Airrac
00013 #include <airrac/AIRRAC_Types.hpp>
00014
00016 namespace stdair {
00017   class STDAIR_Service;
00018 }
00019
00020 namespace AIRRAC {
00021
00025   class AIRRAC_ServiceContext : public stdair::ServiceAbstract {
00031     friend class AIRRAC_Service;
00032     friend class FacAirracServiceContext;
00033
00034   private:
00035     // ///////// Getters //////////
00039     stdair::STDAIR_ServicePtr_T getSTDAIR_ServicePtr() const {
00040       return _stdairService;
00041     }
00042
00046     stdair::STDAIR_Service& getSTDAIR_Service() const {
00047       assert (_stdairService != NULL);
00048       return *_stdairService;
00049     }
00050
00054     const bool getOwnStdairServiceFlag() const {
00055       return _ownStdairService;
00056     }
00057
00058
00059   private:
00060     // ////////////////// Setters ////////////////////
00064     void setSTDAIR_Service (stdair::STDAIR_ServicePtr_T ioSTDAIR_ServicePtr,
00065                             const bool iOwnStdairService) {
00066       _stdairService = ioSTDAIR_ServicePtr;
00067       _ownStdairService = iOwnStdairService;
00068     }
00069
00070
00071   private:
00072     // ///////// Display Methods //////////
00076     const std::string shortDisplay() const;
00077
00081     const std::string display() const;
00082
00086     const std::string describe() const;
00087
00088
00089   private:
00090     // /////// Construction / initialisation ////////
00094     AIRRAC_ServiceContext();
00095
00099     AIRRAC_ServiceContext (const AIRRAC_ServiceContext&);
00100
00104     ~AIRRAC_ServiceContext();
```

```
00105
00109     void reset();
00110
00111
00112   private:
00113     // ////////////// Attributes ///////////////////
00117     stdair::STDAIR_ServicePtr_T _stdairService;
00118
00122     bool _ownStdairService;
00123   };
00124
00125 }
00126 #endif // __AIRRAC_SVC_AIRRACSERVICECONTEXT_HPP
```

25.60 YieldTestSuite.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 YieldTestSuite
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/service/Logger.hpp>
00023 // Airrac
00024 #include <airrac/AIRRAC_Service.hpp>
00025 #include <airrac/config/airrac-paths.hpp>
00026
00027 namespace boost_utf = boost::unit_test;
00028
00029 // (Boost) Unit Test XML Report
00030 std::ofstream utfReportStream ("YieldTestSuite_utfresults.xml");
00031
00035 struct UnitTestConfig {
00037   UnitTestConfig() {
00038     boost_utf::unit_test_log.set_stream (utfReportStream);
00039     boost_utf::unit_test_log.set_format (boost_utf::XML);
00040     boost_utf::unit_test_log.set_threshold_level (boost_utf::log_test_units);
00041     //boost_utf::unit_test_log.set_threshold_level (boost_utf::log_successful_tes
      ts);
00042   }
00043
00045   ~UnitTestConfig() {
00046   }
00047 };
00048
00049 // ////////////////////////////////////////////////////////////////////
00050
00054 void testYieldQuoterHelper (const unsigned short iTestFlag,
00055                             const stdair::Filename_T iYieldInputFilename,
00056                             const bool isBuiltin) {
00057
00058   // Output log File
00059   std::ostringstream oStr;
00060   oStr << "FQTTestSuite_" << iTestFlag << ".log";
00061   const stdair::Filename_T lLogFilename (oStr.str());
00062
00063   // Set the log parameters
00064   std::ofstream logOutputFile;
00065   // Open and clean the log outputfile
00066   logOutputFile.open (lLogFilename.c_str());
00067   logOutputFile.clear();
00068
00069   // Initialise the AirRAC service object
00070   const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG,
00071                                          logOutputFile);
00072
00073   // Initialise the AirRAC service object
00074   AIRRAC::AIRRAC_Service airracService (lLogParams);
00075
```

```
00076   // Build a sample list of travel solutions
00077   stdair::TravelSolutionList_T lTravelSolutionList;
00078   airracService.buildSampleTravelSolutions (lTravelSolutionList);
00079
00080   // Check whether or not a (CSV) input file should be read
00081   if (isBuiltin == true) {
00082
00083     // Build the default sample BOM tree (filled with yields) for AirRAC
00084     airracService.buildSampleBom();
00085
00086   } else {
00087
00088     // Build the BOM tree from parsing the yield input file
00089     AIRRAC::YieldFilePath lYieldFilePath (iYieldInputFilename);
00090     airracService.parseAndLoad (lYieldFilePath);
00091   }
00092
00093   // Calculate the yields for the given travel solution
00094   airracService.calculateYields (lTravelSolutionList);
00095
00096   // Close the log file
00097   logOutputFile.close();
00098
00099 }
00100
00101
00102 // /////////////// Main: Unit Test Suite //////////////
00103
00104 // Set the UTF configuration (re-direct the output to a specific file)
00105 BOOST_GLOBAL_FIXTURE (UnitTestConfig);
00106
00107 // Start the test suite
00108 BOOST_AUTO_TEST_SUITE (master_test_suite)
00109
00110
00113 BOOST_AUTO_TEST_CASE (airrac_simple_yield) {
00114
00115   // Input file name
00116   const stdair::Filename_T lYieldInputFilename (STDAIR_SAMPLE_DIR "/yieldstore01.
      csv");
00117
00118   // State whether the BOM tree should be built-in or parsed from an input file
00119   const bool isBuiltin = false;
00120
00121   // Try to yieldQuote the sample default list of travel solutions
00122   BOOST_CHECK_NO_THROW (testYieldQuoterHelper (0, lYieldInputFilename, isBuiltin)
      );
00123
00124 }
00125
00130 BOOST_AUTO_TEST_CASE (airrac_error_parsing_input_file) {
00131
00132   // Input file name
00133   const stdair::Filename_T lYieldInputFilename (STDAIR_SAMPLE_DIR "/yieldstoreErr
      or01.csv");
00134
00135   // State whether the BOM tree should be built-in or parsed from an input file
00136   const bool isBuiltin = false;
00137
00138   // Try to yield quote the sample default list of travel solutions
00139   BOOST_CHECK_THROW (testYieldQuoterHelper (1, lYieldInputFilename, isBuiltin),
00140                      AIRRAC::YieldFileParsingFailedException);
00141 }
00142
00147 BOOST_AUTO_TEST_CASE (airrac_error_missing_input_file) {
00148
00149   // Input file name
```

```
00150   const stdair::Filename_T lYieldInputFilename (STDAIR_SAMPLE_DIR "/missingFile.c
      sv");
00151
00152   // State whether the BOM tree should be built-in or parsed from an input file
00153   const bool isBuiltin = false;
00154
00155   // Try to yield quote the sample default list of travel solutions
00156   BOOST_CHECK_THROW (testYieldQuoterHelper (2, lYieldInputFilename, isBuiltin),
00157                      AIRRAC::YieldInputFileNotFoundException);
00158 }
00159
00163 BOOST_AUTO_TEST_CASE (airrac_simple_yield_built_in) {
00164
00165   // State whether the BOM tree should be built-in or parsed from an input file
00166   const bool isBuiltin = true;
00167
00168   // Try to yield quote the sample default list of travel solutions
00169   BOOST_CHECK_NO_THROW (testYieldQuoterHelper (3, " ", isBuiltin));
00170
00171 }
00172
00173 // End the test suite
00174 BOOST_AUTO_TEST_SUITE_END()
00175
00176
```

25.61 test/airrac/YieldTestSuite.hpp File Reference

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

Classes

- class YieldTestSuite

Functions

- CPPUNIT_TEST_SUITE_REGISTRATION (YieldTestSuite)

25.61.1 Function Documentation

25.61.1.1 CPPUNIT_TEST_SUITE_REGISTRATION (YieldTestSuite)

25.62 YieldTestSuite.hpp

```
00001 // STL
00002 #include <iosfwd>
00003 // CPPUNIT
00004 #include <cppunit/extensions/HelperMacros.h>
00005
00007 class YieldTestSuite : public CppUnit::TestFixture {
00008   CPPUNIT_TEST_SUITE (YieldTestSuite);
00009   CPPUNIT_TEST (simpleYield);
00010   //  CPPUNIT_TEST (errorCase);
00011   CPPUNIT_TEST_SUITE_END ();
00012 public:
00013
00015   void simpleYield();
00016
00018   //  void errorCase ();
00019
00021   YieldTestSuite ();
00022
00023 private:
00025   void simpleYieldHelper();
00026
00027 protected:
00028   std::stringstream _describeKey;
00029 };
00030
00031 CPPUNIT_TEST_SUITE_REGISTRATION (YieldTestSuite);
```